PARETOGS: HIGH-QUALITY CONTINUOUS-BUDGET 3D GAUSSIAN SPLATTING WITH SPARSE ADAPTED ANCHORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scalable 3D Gaussian Splatting allows one scene representation to serve different Gaussian-count budgets, yet its quality–budget curve may lag behind models optimized separately for each budget. Reducing the number of Gaussians changes both the selected members and the reconstruction tasks carried by the survivors; parameters inherited from a high-budget model may fit a low-budget representation poorly. We propose ParetoGS, which describes member changes and parameter evolution with a small set of adapted budget anchors and decodes intermediate budgets without separate optimization at every requested count. ParetoGS alternates pruning and fixed-member adaptation to build nested anchors with persistent identities, and encodes their parameter changes relative to the full model in a compact residual atlas. At decoding, it interpolates the states of shared members and gradually introduces the rendering contributions of new members, connecting sparse anchors into a continuous-budget representation. Across 13 scenes, ParetoGS attains better average PSNR, SSIM, and LPIPS with fewer Gaussians than the compared prefix methods at 10% of each method's own full model. From a shared full model, it also outperforms FlexGS-imp at its trained 1%, 5%, and 10% budgets and NanoGS-imp at the 1% and 0.1% budgets. On Garden, its non-anchor quality approaches an empirical reference from independently optimized GNS models at higher retention ratios and exceeds it at some lower-budget points. Across 13 scenes, encoding reduces mean additional state from 390.018 to 85.653 MB with an average PSNR change of 0.012 dB.

1 INTRODUCTION

The explicit primitives of 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) make Gaussian count a direct control on rendering cost. Selecting a smaller subset from a full model, however, need not yield a high-quality representation at that count. Pruning and re-optimizing for each budget gives strong individual models but requires repeated training and multiple assets. Scalable methods instead use member orderings, hierarchies, or budget-conditioned transformations to serve many counts from one asset. Their quality–budget curves can still lag behind models optimized separately at each count. We ask whether one representation can decode any Gaussian-count budget while approaching this empirical per-budget quality reference.

Count reduction changes both membership and the reconstruction tasks of the survivors. After other Gaussians are removed, retained members must adjust their geometry, opacity, and appearance to fit the reduced capacity. Even a suitable subset can perform poorly with parameters inherited from the full model. Its adapted state also provides a new starting point for the next pruning decision. ParetoGS therefore contracts from previously adapted anchors and describes both member changes and parameter evolution.

Persistent identities retained through progressive contraction align the same Gaussian across neighboring budgets. Budget evolution then has two parts: discrete member entries and exits, described by a nested order, and changes in the parameters of surviving members. A few optimized budget anchors provide endpoints for the latter. Aligned states at adjacent anchors can reconstruct a piecewise

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.

parameter trajectory at unoptimized counts, turning continuous-budget representation into sparse anchoring and decoding.

We propose ParetoGS, which alternates structural contraction and fixed-member adaptation to build strictly nested anchors. It preserves identities and deletion order, encodes each anchor's changes relative to the full model in a layered residual atlas, and decodes intermediate budgets by interpolating shared-member states and gradually activating entering members. An approximately doubling schedule places anchors from 0.1% of Full to the full count, yielding a single asset that serves every integer count in this range without further optimization.

Across 13 scenes, ParetoGS outperforms CLOD-3DGS and MGS at 10% of each method's own full model while using fewer Gaussians. From a shared full model, it improves mean quality over FlexGS-imp at its trained 1%, 5%, and 10% budgets and over NanoGS-imp at 1% and 0.1%. On Garden, non-anchor quality approaches independently optimized GNS results at higher retention and exceeds them at some lower budgets; qualitative results also show better low-budget structure. Across 13 scenes, encoding reduces mean additional state from 390.018 to 85.653 MB.

Our contributions are:

- A representation of continuous-budget 3DGS as a joint trajectory of nested membership and budget-specific parameter states, organized by progressively adapted anchors.
- ParetoGS, which connects those anchors through persistent identities, aligned residual encoding, and piecewise decoding to generate any intermediate count without separate optimization.
- Thirteen-scene multi-budget comparisons, non-anchor evaluation, and ablations showing the value of adapted states and interval decoding, together with compact storage of the anchor family.

2 Related Work

3DGS representation and reconstruction. 3DGS (Kerbl et al., 2023) combines explicit Gaussians with differentiable rasterization. Subsequent work extends primitives and dynamic scenes (Huang et al., 2024; Wu et al., 2024), improves surface reconstruction (Yu et al., 2024b; Chen et al., 2025), visibility and filtering (Radl et al., 2024; Yu et al., 2024a), rendering and training efficiency (Hanson et al., 2025a; Mallick et al., 2024), and initialization and capacity allocation (Kotovenko et al., 2026; Deng et al., 2026; Kheradmand et al., 2024). Wild-GS, PPISP, and Deblurring 3D-GS address appearance changes, camera photometry, and blur in real captures (Xu et al., 2024; Deutsch et al., 2026; Lee et al., 2024a). We focus on organizing an already trained scene into an asset decodable at arbitrary Gaussian budgets.

Compact and multilevel representations. Pruning reduces membership, whereas compression encodes attributes more compactly. HAC, SOGS, CompGS, and Compact-3DGS explore context coding, regular grids, vector quantization, and compact parameterization (Chen et al., 2024; Morgenstern et al., 2025; Navaneet et al., 2025; Lee et al., 2024b). LightGaussian, PUP 3D-GS, and Trimming the Fat use contribution or sensitivity for pruning (Fan et al., 2024; Hanson et al., 2025b; Ali et al., 2024); Mini-Splatting and GHAP use sampling or mixture reduction (Fang & Wang, 2024; Wang et al., 2025). MaskGaussian and GNS integrate member selection with adaptation (Liu et al., 2025c; Deng et al., 2025). Scale and spatial hierarchies organize models at different resolutions or capacities (Yan et al., 2024; Ren et al., 2025; Liu et al., 2025b; Kerbl et al., 2024; Kulhanek et al., 2025; Seo et al., 2025; Shen et al., 2025; Cheng et al., 2026). H3DGS also interpolates optimized parent–child attributes (Kerbl et al., 2024). Contribution orderings and layered decompositions yield cumulative representations (Zoomers et al., 2025; Tsai et al., 2025; Lavi et al., 2026), sometimes with progressive compression or attribute precision (Di Sario et al., 2025; Chen et al., 2026). ParetoGS uses GNS to build budget anchors and connects their aligned parameter states.

Arbitrary-budget decoding and adaptation. CLOD-3DGS fine-tunes random prefixes, MGS jointly trains a prefix and the full set on each view, and Differentiable Hierarchies learns a count-truncatable ordering (Milef et al., 2025; Guo et al., 2026; Pan et al., 2026). These approaches reuse the same Gaussian parameters across prefix budgets. RAVE reorders incremental members between

neighboring bitrate levels to connect intermediate rates and counts (Tran et al., 2026). LapisGS arranges content into cumulative resolution layers and interpolates readapted old-layer opacity while retaining old members' geometry and appearance (Shi et al., 2025). SCAR-GS progressively refines neural anchor features with residual vector quantization (Revilla et al., 2026); its residuals approximate features rather than record optimized states at different budgets. NanoGS progressively merges local pairs to produce a compact model at a target count without image-supervised optimization (Xiong et al., 2026). FlexGS learns both budget-conditioned selection and geometry transformation; its paper optimizes these modules after 15K pretraining (Liu et al., 2025a). ParetoGS instead stores a few explicitly optimized nested anchors, continues pruning from each adapted state, and connects the geometry, opacity, and appearance of corresponding members. Experiments assess the resulting quality at equal own-Full fractions, from a shared Full, and at non-anchor budgets.

3 METHOD

ParetoGS progressively contracts a trained full model into nested, adapted budget anchors. Persistent identities align their members; a layered residual atlas stores anchor states relative to the full model.

3.1 BUDGET-CONDITIONED SCENE REPRESENTATION

3DGS projects Gaussians carrying position, shape, opacity, and appearance into a view and composites them in depth order. Their opacity and preceding transmittance determine the color at pixel u:

$$I(u) = \sum_i T_i(u)\widetilde{a}_i(u)c_i(u) + T_{\mathrm{bg}}(u)c_{\mathrm{bg}}, \qquad T_i(u) = \prod_{k \prec_u i} \big(1 - \widetilde{a}_k(u)\big). \tag{1}$$

Here $\widetilde{a}_i(u)$ combines activated opacity and projected coverage, $c_i(u)$ is view-dependent SH color, and $k \prec_u i$ denotes a Gaussian preceding i. $T_i(u)$ is the transmittance before i; T_{bg} and c_{bg} describe the background. Removal also changes later members' transmittance weights, so full-model parameters need not fit a contracted model.

Let the full model contain N Gaussians, each with a persistent identity z_i and parameter vector $\theta_i = (\mathbf{x}_i, \mathbf{s}_i, \mathbf{r}_i, o_i, \mathbf{c}_i)$ for position, log scale, raw rotation quaternion, opacity logit, and spherical-harmonic coefficients. At budget K, let $\mathcal{S}(K)$ be the active identity set and $\theta_i(K)$ the parameters of member i. The budget-conditioned model has exactly K active members:

$$\mathcal{G}(K) = \{(z_i, \theta_i(K)) \mid z_i \in \mathcal{S}(K)\}, \qquad |\mathcal{S}(K)| = K. \tag{2}$$

We impose $\mathcal{S}(K_a) \subset \mathcal{S}(K_b)$ for $K_a < K_b$: larger budgets add members while allowing retained members' parameters to change. States are connected in optimization coordinates; scales are exponentiated, opacity is passed through a sigmoid, and quaternions are normalized for rendering. The asset decodes exactly K Gaussians for every integer $K \in [K_{\min}, N]$.

3.2 SPARSE ADAPTED ANCHORS AND MEMBER CORRESPONDENCE

We explicitly optimize a few budgets, adapting survivors at each anchor before starting the next contraction.

Let M anchors have budgets $K_{\min} = K_1 < K_2 < \cdots < K_M = N$. From Full, each stage contracts to the next target while optimizing parameters, then holds membership fixed and continues optimizing pixel L_1 and SSIM reconstruction losses. The adapted state is a decoding endpoint and the next stage's starting point. The chain deletes existing Gaussians without further densification; persistent IDs maintain correspondence as parameters change.

When contracting upper anchor U to lower anchor L, we record deletions and reverse them to obtain the entry order $\pi_{L\to U}$ as budget grows. For $L < K < U$, decoding retains every member of the lower anchor and admits the first $K - L$ entries:

$$\mathcal{S}(K) = \mathcal{S}(L) \cup \pi_{L\to U}[1 : K - L], \qquad L < K < U. \tag{3}$$

Concatenating the smallest anchor with these entry sequences yields a global nested order, indexed so budget K uses the first K rows.

Our approximately doubling schedule starts at 0.1% of the actual full count and ends at the full budget, giving 11 anchors for every scene. The target counts are $K_j = \mathrm{round}(N\rho_j)$ for $\rho_j = 0.001, 0.002, \ldots, 0.512, 1$. When building downward from Full, each stage inherits the previous anchor's Gaussian parameters, optimizer state, and persistent IDs. It resets the iteration label to 14,999, uses GNS (Deng et al., 2025) to reach K_j during 15K–18K, and continues fixed-member adaptation through 25K. GNS adjusts regularization in response to a prescribed linear opacity-decay trajectory at the budget boundary, advances removal within the selection window, and uses its budget safeguard to retain exactly K_j members. Densification, parameter learning, and pruning control follow the existing ImprovedGS (Deng et al., 2026) and GNS procedures.

3.3 Piecewise Budget Decoding

The nested order fixes membership. Shared members have aligned states at both neighboring anchors, whereas entering members have only an upper-anchor state.

Let $L < K < U$ denote lower and upper anchors and the requested count. The first L members connect their endpoint parameters with a weight that grows toward U. For normalized progress $t \in (0, 1)$,

$$t = \frac{K - L}{U - L}, \qquad \theta_i(K) = (1 - t^p)\theta_i(L) + t^p \theta_i(U), \quad i \le L. \tag{4}$$

$\theta_i(L)$ and $\theta_i(U)$ describe the same member. The exponent $p > 0$ maps budget progress onto their parameter-space segment: $p = 1$ is linear, while $p < 1$ and $p > 1$ approach the upper state earlier and later.

Entering members take upper-anchor geometry and appearance; an opacity gate controls their rendering contribution. With $q > 0$ and upper-anchor opacity $a_i(U) = \sigma(o_i(U))$,

$$a_i(K) = \mathrm{clip}\big(t^q a_i(U), \epsilon, 1 - \epsilon\big), \qquad L < i \le K. \tag{5}$$

σ is the sigmoid, and a small $\epsilon > 0$ bounds opacity within $[\epsilon, 1 - \epsilon]$ before conversion back to a finite logit. For example, decoding 150K between 100K and 200K anchors interpolates the first 100K states and admits 50K upper-anchor members with opacity scaled by 0.5^q.

We calibrate p, q per interval on training views while fixing anchors and member order. At $t \in \{0.25, 0.5, 0.75\}$, candidate exponents minimize reconstruction loss plus penalties for loss regressions and slope deviations; Appendix A gives the objective and search grid. Only two scalars are stored per interval. Calibration uses FP16 residual anchor states; deployment restores compressed endpoints with the same exponents. An anchor-budget request returns its endpoint directly.

3.4 Layered Residual Atlas

The layered residual atlas encodes all anchor changes by member layer and parameter field, with Full as a common reference; decoding a request restores only its neighboring endpoints.

Let Θ_{K_j} be the parameter matrix at anchor K_j, one Gaussian per row, and order the full matrix Θ_N by the global member sequence. For an anchor, we store its changes relative to the corresponding rows of the full model:

$$\Theta_{K_j} = \Theta_N[1 : K_j] + \Delta_j. \tag{6}$$

The full anchor has zero residual and needs no additional state.

Members first present at anchor K_s form layer s: $\mathcal{B}_s = \mathcal{S}(K_s) \setminus \mathcal{S}(K_{s-1})$, with $K_0 = 0$ and $\mathcal{S}(K_0) = \varnothing$. For each field f and $s < M$, flattening and stacking residuals from K_s through K_{M-1} gives D_s^f, with anchors as rows and member components as columns. Members present only at Full require no residual.

An orthogonal transform along the state dimension gives shared residual bases:

$$D_s^f = C_s^f B_s^f. \tag{7}$$

Rows of B_s^f are bases, and rows of C_s^f are anchor coefficients. We retain all state components, applying later compression to the bases.

Higher-order SH coefficients dominate storage. For each residual basis component, we approximate its Gaussian-by-channel matrix along the channel axis, saving per-Gaussian coefficients and a shared channel basis that reconstructs every channel. These coefficients and other fields' bases are block-quantized; anchor coefficients, channel bases, and quantization scales use half precision. Appendix A gives the ranks and quantization rule.

The anchor schedule also bounds the amount of state to encode: $\sum_{j<M} K_j \approx 1.023N$, so the valid raw parameters of all non-full anchors approach one full parameter tensor before encoding. The final asset comprises the ordered full model, residual atlas, interval exponents, and member-order metadata. At runtime, an anchor residual $\widehat{\Delta}_j$ restored from the atlas is added to the corresponding reference rows: $\widehat{\Theta}_{K_j} = \Theta_N[1:K_j] + \widehat{\Delta}_j$. The hat denotes a state approximated by lossy encoding. The resulting endpoints follow the rules of Section 3.3 to produce a standard K-Gaussian model for an existing rasterizer. An anchor request incurs encoding error but no interval state-estimation error. With the full model and required endpoints resident, parameter generation takes $O(K)$ work for fixed parameter dimension.

4 Experiments

We evaluate cross-budget reconstruction quality, budget-specific parameter states, interval-decoding components, and additional storage to test whether sparse adapted anchors form a high-quality family of budgeted models.

4.1 Experimental Setup

Datasets and metrics. We report PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) on nine Mip-NeRF 360 (Barron et al., 2022) scenes, Train and Truck from Tanks and Temples (Knapitsch et al., 2017), and Playroom and DrJohnson from Deep Blending (Hedman et al., 2018). The outdoor and indoor Mip-NeRF 360 scenes use the $4\times$ and $2\times$ downsampled images provided by 3DGS (Kerbl et al., 2023); the other datasets use its provided versions. After sorting image filenames, every eighth view is held out for testing and the rest are used for training. All methods use the same test views and image sizes, with the three metrics computed by the 3DGS evaluation implementation. The main tables and component and encoding ablations weight the 13 scenes equally; ablations first average ten interval midpoints within each scene.

Implementation and comparison protocol. We train a full model for 30K iterations with ImprovedGS (Deng et al., 2026), then progressively contract and adapt it with GNS (Deng et al., 2025), retaining 11 budget anchors. The anchors span 0.1% to 100% of each scene's full count; each interval uses ten training views to calibrate p, q. Appendix A gives the member-order, state-encoding, and interval-calibration procedures. All experiments use an RTX 4090 GPU with `data_device` set to GPU.

We use the comparison methods' official source code in two protocols. First, CLOD-3DGS (Milef et al., 2025), MGS (Guo et al., 2026), and ParetoGS each use their own full model; we compare equal retention ratios and report actual Gaussian counts. CLOD-3DGS trains down to 100K Gaussians. Second, FlexGS-imp and NanoGS-imp receive the same ImprovedGS full anchor as ParetoGS, and we compare equal requested budgets. FlexGS-imp runs the public FlexGS (Liu et al., 2025a) code's 35K-iteration budget-adaptation stage, replacing the output of its standard 3DGS coarse stage with the shared full model; this stage samples 1%, 5%, 10%, and 15%. Table 2 also queries 0.1% and 30% outside that sampled range. NanoGS-imp uses its official implementation to simplify the same full model offline (Xiong et al., 2026). Table headers report 13-scene mean offline time: ParetoGS includes full-model and anchor training, encoding, calibration, and packaging, while FlexGS-imp includes the shared full-model construction and its 35K adaptation stage. Appendix B provides per-scene quality, counts, and time.

4.2 Reconstruction Quality and Offline Cost

Quality at equal retention ratios. Table 1 shows the largest ParetoGS gains at low budgets. At 10%, 5%, 1%, and 0.1%, all three mean quality metrics improve over CLOD-3DGS and MGS. At

Table 1: Equal retention ratios from each method's own full model. Metrics and actual Gaussian counts are equally weighted 13-scene means; Num gives the actual count in thousands, and full-model sizes differ. Parentheses after method names give mean offline time in minutes; Ours denotes ParetoGS. Bold marks the best mean quality at each ratio.

Ratio	CLOD-3DGS (∼35 min)				MGS (∼74 min)				Ours (∼10.3 min)			
	PSNR↑	SSIM↑	LPIPS↓	Num	PSNR↑	SSIM↑	LPIPS↓	Num	PSNR↑	SSIM↑	LPIPS↓	Num
100%	27.54	0.847	0.197	3029	27.75	0.853	**0.180**	4923	**27.89**	**0.854**	0.185	1615
30%	27.24	0.834	0.222	909	27.32	0.835	**0.206**	1477	**27.88**	**0.850**	0.207	485
10%	25.59	0.779	0.294	303	25.74	0.757	0.301	492	**27.17**	**0.821**	**0.269**	162
5%	22.38	0.691	0.376	151	24.32	0.694	0.372	246	**26.48**	**0.791**	**0.315**	81
1%	12.86	0.383	0.581	30	20.35	0.565	0.507	49	**24.40**	**0.697**	**0.422**	16
0.1%	8.15	0.077	0.712	3	14.79	0.424	0.628	5	**21.15**	**0.580**	**0.539**	2

Table 2: Shared full initialization at equal requested budgets. Metrics and requested Gaussian counts are equally weighted 13-scene means; Num is in thousands, and Conv. is NanoGS-imp's offline conversion time from Full (seconds). Parentheses after method names give mean offline time in minutes; Ours denotes ParetoGS.

Ratio	Num	FlexGS-imp (∼81 min)			NanoGS-imp				Ours (∼10.3 min)		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	Conv.	PSNR↑	SSIM↑	LPIPS↓
30%	485	21.98	0.705	0.355	23.86	0.708	0.329	48	**27.88**	**0.850**	**0.207**
10%	162	25.21	0.760	0.327	21.29	0.603	0.436	54	**27.17**	**0.821**	**0.269**
5%	81	24.03	0.712	0.386	20.34	0.564	0.486	55	**26.48**	**0.791**	**0.315**
1%	16	21.75	0.621	0.488	18.84	0.511	0.579	56	**24.40**	**0.697**	**0.422**
0.1%	2	13.54	0.392	0.641	16.37	0.470	0.637	56	**21.15**	**0.580**	**0.539**

10%, ParetoGS reaches 27.17 dB with about 162K Gaussians; CLOD-3DGS and MGS use about 303K and 492K for 25.59 and 25.74 dB. Across those four low ratios, ParetoGS gains about 3.50 dB in PSNR over MGS on average. CLOD-3DGS fine-tunes random prefixes after full pretraining, and MGS jointly optimizes the full set and a sampled prefix on each step. Their mean offline costs are about 35 and 74 minutes, versus 10.3 minutes for ParetoGS's full model, anchor chain, encoding, calibration, and packaging. At high budgets the differences narrow; MGS has slightly better mean LPIPS at Full and 30%.

Quality from a shared full model. At the sampled 1%, 5%, and 10% budgets in Table 2, ParetoGS has better 13-scene mean PSNR, SSIM, and LPIPS than FlexGS-imp. At 5%, these are 26.48 versus 24.03 dB, 0.791 versus 0.712, and 0.315 versus 0.386. FlexGS-imp spends about 81 minutes including construction of the shared full model and its 35K-step adaptation stage, whereas the ParetoGS asset takes about 10.3 minutes to construct. The three trained-budget comparisons retain a clear quality gap. NanoGS studies 10%, 1%, and 0.1% as major compression budgets (Xiong et al., 2026). At 1%, ParetoGS increases its mean PSNR from 18.84 to 24.40 dB. At 0.1%, PSNR rises from 16.37 to 21.15 dB, SSIM from 0.470 to 0.580, and LPIPS decreases from 0.637 to 0.539. NanoGS-imp takes 48–56 seconds to convert the shared full model into each target model offline (Table 2). The default FlexGS-imp budget module has a fixed size of about 30 MB; Section 4.5 compares the additional ParetoGS states and total assets.

Figure 1 shows the same Garden test view across budgets. At 30% and 10%, CLOD-3DGS and MGS reconstruct the table, flowerpots, and background well. At 1%, their main structures fragment, and at 0.1% only scattered content remains. FlexGS-imp retains the table and flowerpots at its trained 1%, 5%, and 10% ratios, but much of the background becomes blurred. NanoGS-imp retains more content around 1% and 5%, with dense jagged and fragmented artifacts. At 0.1%, CLOD-3DGS



Figure 1: Cross-budget reconstruction of the same Garden test view. The five columns show CLOD-3DGS, MGS, FlexGS-imp, NanoGS-imp, and ParetoGS; the rows show 30%, 10%, 5%, 1%, and 0.1%. The first two columns use a fraction of each method's own full model; the last three use a shared full model. Each panel labels its actual Gaussian count.

is below its 100K training-count floor and FlexGS-imp is below its smallest sampled training ratio; NanoGS-imp still preserves the scene outline. ParetoGS produces clearer and more coherent table, flowerpots, and background at the decoded 30%, 10%, 5%, and 1% budgets. Its smallest 0.1% model comes from a progressively adapted anchor and retains the main objects and overall layout. The visual results complement the quality comparisons in Table 2.

4.3 Budget-Specific States and Continuous Decoding

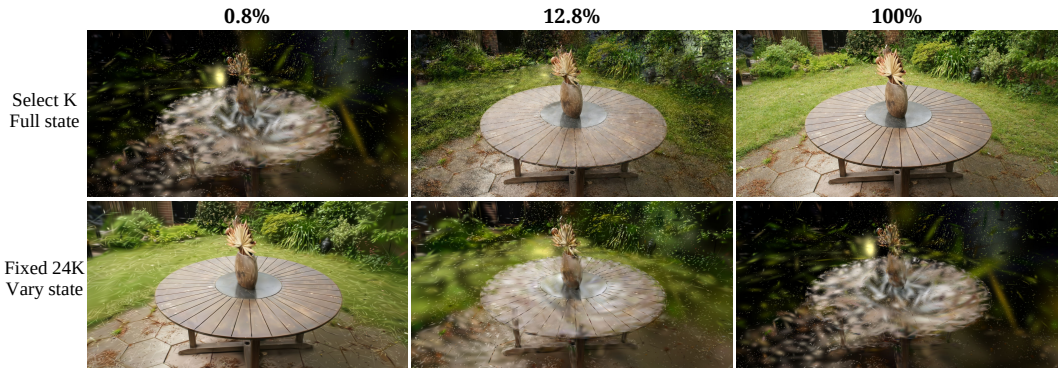


Figure 2: Parameter states of the same members across budgets. The Garden columns correspond to 0.8%, 12.8%, and 100% of Full. The first row selects 24K, 384K, and 3M members, respectively, while retaining Full parameters. The second row fixes the same 24K members from the upper-left panel; column headings now indicate the budget from which their parameters were taken. All panels use the same view and crop.

Why the same members need different states. The first row of Figure 2 selects members at each budget while retaining their full-model parameters. At 0.8%, much of the table and background is missing, showing that member selection alone does not preserve reconstruction quality. The second row fixes the same 24K Gaussians selected in the upper-left panel and changes only their budget-specific parameters. Viewed from right to left, these members by themselves cover only a small part of the scene in their Full state; as budget decreases, they take on the table and background, eventually reconstructing the whole view at 0.8%. The large change in task for the same identities motivates adapted parameter states at each anchor and their connection between anchors.

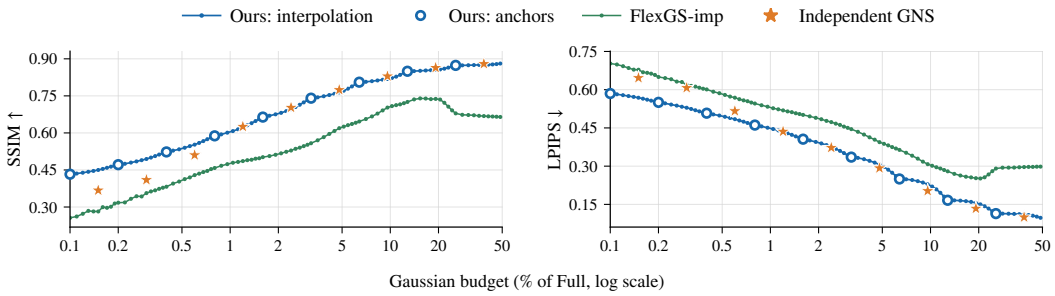


Figure 3: Continuous-budget quality on Garden. The full model has 3M Gaussians. SSIM and LPIPS are averaged over 24 test views against retention percentage on a log axis. Blue open circles are ParetoGS anchors and small blue dots are decoded intervals; green denotes FlexGS-imp, and orange stars are independently optimized GNS references. Line segments connect measured points over 0.1%–50%.

Quality between sparse anchors. Figure 3 evaluates non-anchor budgets on Garden. Independent GNS contracts the same Full directly to each target during 15K–18K and adapts fixed members through 25K; its measured points form an empirical per-budget reference. ParetoGS uses actual decoded test-view renderings. From 1% to 50% retention, its decoded SSIM and LPIPS closely track the independent GNS points. Below about 1%, the curves separate in favor of ParetoGS, with the largest gap at the smallest tested budgets. The other 12 scenes in Appendix E show a similar budget-dependent pattern.

4.4 Decoder Components and Anchor Ablations

We evaluate the arithmetic midpoint of each neighboring pair among the 11 anchors, giving ten budgets with identical member IDs across configurations. Table 3 compares full-model parameters for those members, a fixed lower-anchor shared state, no opacity gate for entering members, and linear progress $p = q = 1$. Without the gate, entering members use their upper-anchor opacity directly.

Table 3: Decoder components: mean quality at ten interval midpoints across 13 scenes. The Full-parameter row uses the same member IDs.

Variant	PSNR↑	SSIM↑	LPIPS↓
Full method	**25.473**	**0.74978**	0.34578
Fixed shared state	25.346	0.74795	0.34772
No opacity gate	25.261	0.74779	**0.34563**
Linear progress	25.329	0.74588	0.34905
Full parameters	17.073	0.54587	0.47175

Table 4: State encoding: quality at the same midpoints and mean additional storage across 13 scenes.

Encoding	PSNR↑	SSIM↑	LPIPS↓	MB↓
FP32	25.485	0.75031	0.34524	390.018
FP16	25.485	0.75031	0.34524	195.019
Atlas	25.473	0.74978	0.34578	85.653

Fixing common members at the lower-anchor state reduces mean PSNR by 0.127 dB and SSIM by 0.00183, while increasing LPIPS by 0.00194. Parameter changes matter even within a single anchor interval. Without the entering-member gate, PSNR and SSIM fall while LPIPS stays similar. Calibrated p, q improve all three metrics over $p = q = 1$.

Finally, assigning Full-model parameters to the same IDs lowers PSNR from 25.473 to 17.073 dB, SSIM from 0.74978 to 0.54587, and raises LPIPS from 0.34578 to 0.47175. This quantifies the budget-specific state change illustrated in Figure 2; all 13 scenes show the same decline (Appendix Table 15).

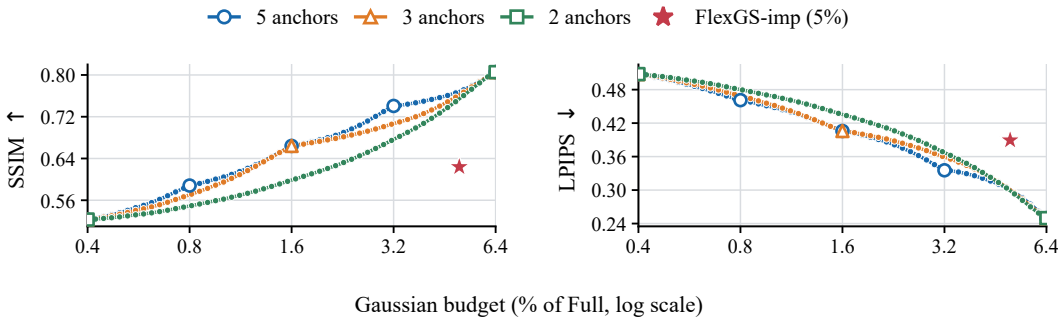


Figure 4: Effect of intermediate anchors on connection quality. From the same Garden state library and common 0.4% and 6.4% endpoints, we retain five, three, or two anchors. Open circles, triangles, and squares mark their anchors; small solid points show real evaluations at 50 shared non-anchor budgets. The red star is FlexGS-imp at the 5% Garden budget from the shared full model. Each configuration calibrates its own interval exponents, and metrics average all 24 test views.

Figure 4 remains smooth when anchor spacing doubles or quadruples, with mean SSIM changing from 0.65687 to 0.64666 and 0.61795. Directly adapted anchors exceed interpolation from sparse endpoints at 0.8% and 3.2%. Even the two-endpoint interpolation at 5% exceeds FlexGS-imp trained at that ratio (SSIM 0.62390, LPIPS 0.38939). Intermediate anchors raise the connecting curve's average quality.

4.5 State Encoding and Storage

Across 13 scenes, Table 4 reduces mean additional state from 390.018 MB (FP32) to 85.653 MB (atlas), or 78.04%, with only a 0.012 dB PSNR drop, 0.00053 SSIM drop, and 0.00054 LPIPS increase. FP16 residuals use 195.019 MB with the same quality at shown precision. For Bicycle, Garden, and Stump, the atlas averages 158.666 MB (Appendix Table 16); with their shared 744 MB Full, ParetoGS totals about 903 MB, versus about 774 MB for Full plus FlexGS-imp's 30 MB module. This roughly 17% asset increase accompanies better multi-budget quality and shorter offline construction. With anchors resident on the GPU, ParetoGS parameter construction has a 2.773 ms P95 (Appendix C). A deployment capped at 51.2% of Full can use that anchor as the reference and re-encode smaller states to reduce asset storage.

5 Conclusion

ParetoGS describes both Gaussian membership and budget-specific parameter states with sparse adapted anchors. Nested identities establish a cross-budget member order, while shared-state interpolation and an entering-member gate connect discrete anchors into a family of models decodable at requested counts. A residual atlas compresses the additional adapted states.

Across 13 scenes, ParetoGS achieves better mean quality than the compared prefix methods with fewer Gaussians at 10% retention. From a shared full model, it also improves over FlexGS-imp at its sampled 1%, 5%, and 10% budgets and over NanoGS-imp at 1% and 0.1%. Continuous-budget curves and component ablations across 13 scenes further demonstrate the value of adapted parameters and interval decoding. The compressed atlas reduces mean additional state from 390.018 to 85.653 MB with little average quality change. Organizing budget-specific states into one asset thus provides higher-quality multi-budget decoding from a single scene representation. In a three-scene asset comparison, ParetoGS uses about 17% more total storage than FlexGS-imp; its mean offline construction time across 13 scenes is shorter than those of CLOD-3DGS, MGS, and FlexGS-imp.

AI USE STATEMENT

Generative AI tools assisted with code writing, figure and table preparation, manuscript revision and English translation, and provided feedback on the method, ablation design, and interpretation of results. The authors reviewed all AI-assisted work, checked the algorithm, experimental data, and manuscript claims, and take responsibility for the final content.

ETHICS STATEMENT

This work studies 3D reconstruction on established benchmark datasets. We are not aware of ethical issues specific to the proposed method or its evaluation.

REPRODUCIBILITY STATEMENT

Section 4 specifies the datasets and comparison protocol. Appendix A provides the ordering, encoding, calibration, and decoding procedures; Appendix B reports per-scene results. Appendix C describes the budget-switch timing protocol and results.

REFERENCES

Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3D Gaussian Splats through pruning. In *British Machine Vision Conference (BMVC)*, 2024. URL https://bmvc2024.org/proceedings/358/.

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)*, 2022. URL https://jonbarron.info/mipnerf360/.

Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. PGSR: Planar-Based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 31(9):6100–6111, 2025. doi: 10.1109/TVCG.2024.3494046.

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)*, 2024.

Yihang Chen, Mengyao Li, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. PCGS: Progressive compression of 3D Gaussian Splatting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(4):3111–3119, 2026. doi: 10.1609/aaai.v40i4.37304.

Zhigang Cheng, Mingchao Sun, Yu Liu, Zengye Ge, Luyang Tang, Mu Xu, Yangyan Li, and Peng Pan. CLoD-GS: Continuous level-of-detail via 3D Gaussian Splatting. In *International Conference on Learning Representations (ICLR)*, 2026.

Xiaobin Deng, Qiuli Yu, Changyu Diao, Min Li, and Duanqing Xu. Gradient-Driven Natural Selection for Compact 3D Gaussian Splatting. *arXiv preprint arXiv:2511.16980*, 2025. URL https://arxiv.org/abs/2511.16980v1.

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.

Isaac Deutsch, Nicolas Moënne-Loccoz, Gavriel State, and Zan Gojcic. PPISP: Physically-plausible compensation and control of photometric variations in radiance field reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7289–7298, June 2026.

Francesco Di Sario, Riccardo Renzulli, Marco Grangetto, Akihiro Sugimoto, and Enzo Tartaglione. GoDe: Gaussians on demand for progressive level of detail and scalable compression. *arXiv preprint arXiv:2501.13558*, 2025. URL https://arxiv.org/abs/2501.13558v3.

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, 2024.

Guangchi Fang and Bing Wang. Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians. In *European Conference on Computer Vision*, 2024.

Zhilin Guo, Boqiao Zhang, Hakan Aktas, Kyle Fogarty, Jeffrey Hu, Nursena Koprucu Aslan, Wenzhao Li, Canberk Baykal, Albert Miao, Josef Bengtson, Chenliang Zhou, Weihao Xia, Cristina Nader Vasconcelos, and Cengiz Oztireli. Matryoshka Gaussian Splatting. *arXiv preprint arXiv:2603.19234*, 2026. URL https://arxiv.org/abs/2603.19234v2.

Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-Splat: Fast 3D Gaussian Splatting with Sparse Pixels and Sparse Primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21537–21546, June 2025a.

Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. PUP 3D-GS: Principled Uncertainty Pruning for 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5949–5958, June 2025b.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep Blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics*, 37(6), 2018. URL https://www-sop.inria.fr/reves/Basilic/2018/HPPFDB18/.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*, 2024. doi: 10.1145/3641519.3657428.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM Transactions on Graphics*, 43(4), 2024. doi: 10.1145/3658160.

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.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. URL https://www.tanksandtemples.org/.

Dmytro Kotovenko, Olga Grebenkova, and Björn Ommer. EDGS: Eliminating Densification for Efficient Convergence of 3DGS. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41065–41076, 2026.

Jonas Kulhanek, Marie-Julie Rakotosaona, Fabian Manhardt, Christina Tsalicoglou, Michael Niemeyer, Torsten Sattler, Songyou Peng, and Federico Tombari. LODGE: Level-of-detail large-scale Gaussian Splatting with efficient rendering. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yishai Lavi, Leo Segre, and Shai Avidan. Frequency-aware Gaussian Splatting decomposition. In *International Conference on 3D Vision (3DV)*, pp. 1133–1142, 2026. doi: 10.1109/3DV69130.2026.00112.

Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3D Gaussian Splatting. In *European Conference on Computer Vision (ECCV)*, 2024a. doi: 10.1007/978-3-031-73636-0_8.

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, June 2024b.

Hengyu Liu, Yuehao Wang, Chenxin Li, Ruisi Cai, Kevin Wang, Wuyang Li, Pavlo Molchanov, Peihao Wang, and Zhangyang Wang. FlexGS: Train once, deploy everywhere with many-in-one flexible 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16336–16345, 2025a.

Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. CityGaussian: Real-time High-quality Large-Scale Scene Rendering with Gaussians. In *Computer Vision – ECCV 2024*, pp. 265–282, 2025b.

Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. MaskGaussian: Adaptive 3D Gaussian Representation from Probabilistic Masks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 681–690, June 2025c.

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

Nicholas Milef, Dario Seyb, Todd Keeler, Thu Nguyen-Phuoc, Aljaz Bozic, Sushant Kondguli, and Carl Marshall. Learning fast 3D Gaussian Splatting rendering using continuous level of detail. *Computer Graphics Forum*, 44(2):e70069, 2025. doi: 10.1111/cgf.70069.

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. In *Computer Vision – ECCV 2024*, pp. 18–34, 2025. doi: 10.1007/978-3-031-73013-9_2.

K L Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. CompGS: Smaller and Faster Gaussian Splatting with Vector Quantization. In *Computer Vision – ECCV 2024*, pp. 330–349, 2025. doi: 10.1007/978-3-031-73411-3_19.

Youqi Pan, Wugen Zhou, and Hongbin Zha. Learning Differentiable Hierarchies in 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 40939–40948, June 2026.

Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Transactions on Graphics*, 43(4), 2024.

Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-GS: Towards consistent real-time rendering with LOD-structured 3D Gaussians. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. doi: 10.1109/TPAMI.2025.3568201.

Diego Revilla, Pooja Suresh, Wei Tsang Ooi, and Anand Bhojan. SCAR-GS: Spatial context attention for residuals in progressive Gaussian Splatting. *arXiv preprint arXiv:2601.04348*, 2026. URL https://arxiv.org/abs/2601.04348v2.

Yunji Seo, Young Sun Choi, Hyun Seung Son, and Youngjung Uh. FLoD: Integrating flexible level of detail into 3D Gaussian Splatting for customizable rendering. *ACM Transactions on Graphics*, 44(4), 2025. doi: 10.1145/3731430.

Jianxiong Shen, Yue Qian, and Xiaohang Zhan. LOD-GS: Achieving Levels of Detail using Scalable Gaussian Soup. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 671–680, June 2025.

Yuang Shi, Géraldine Morin, Simone Gasparini, and Wei Tsang Ooi. LapisGS: Layered progressive 3D Gaussian Splatting for adaptive streaming. In *International Conference on 3D Vision (3DV)*, pp. 991–1000, 2025.

Hoang-Nhat Tran, Francesco Di Sario, Gabriele Spadaro, Giuseppe Valenzise, and Enzo Tartaglione. RAVE: Rate-adaptive visual encoding for 3D Gaussian Splatting. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 11727–11731, 2026. doi: 10.1109/ICASSP55912.2026.11463333.

Yi-Zhen Tsai, Xuechen Zhang, Zheng Li, and Jiasi Chen. L3GS: Layered 3D Gaussian Splats for efficient 3D scene delivery. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*, 2025. doi: 10.1145/3680207.3723485.

Tao Wang, Mengyu Li, Geduo Zeng, Cheng Meng, and Qiong Zhang. Gaussian herding across pens: An optimal transport perspective on global gaussian reduction for 3DGS. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5277.

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), 2004. URL https://ece.uwaterloo.ca/~z70wang/publications/ssim.html.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20310–20320, 2024.

Butian Xiong, Rong Liu, Tiantian Zhou, Meida Chen, Zhiwen Fan, and Andrew Feng. NanoGS: Training-Free Gaussian Splat Simplification. *arXiv preprint arXiv:2603.16103*, 2026. URL https://arxiv.org/abs/2603.16103v3.

Jiacong Xu, Yiqun Mei, and Vishal M. Patel. Wild-GS: Real-time novel view synthesis from unconstrained photo collections. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

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

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19447–19456, June 2024a.

Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian Opacity Fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024b.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. URL https://richzhang.github.io/PerceptualSimilarity/.

Brent Zoomers, Maarten Wijnants, Ivan Molenaers, Joni Vanherck, Jeroen Put, Lode Jorissen, and Nick Michiels. PRoGS: Progressive rendering of Gaussian Splats. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pp. 3118–3127, 2025.

A State Encoding, Calibration, and Budget Decoding

This section specifies the member ordering, residual encoding, interval calibration, and decoding rules used by the main method. Given constructed nested adapted anchors and the full model, the runtime procedure returns parameters for exactly K Gaussians at any valid integer budget.

A.1 Member Ordering and Residual Atlas

When contracting an upper anchor U to a lower anchor L, we record deleted IDs. Reversing each deletion trace yields the order in which members enter as budget increases. Starting with the smallest anchor and concatenating these entry sequences gives the global nested ordering used in the paper; full-model parameters are reordered accordingly. Anchors are aligned to this ordering by ID, and their residuals relative to Full are computed in raw parameter coordinates. The six fields are position, log scale, quaternion, opacity logit, DC color, and higher-order SH coefficients, giving 59 scalar components per Gaussian at SH degree three.

Non-full anchor residuals are converted to FP16 before stacking by member layer and parameter field. Each layer undergoes an orthogonal decomposition along the anchor-state dimension. The maximum supplied state-axis rank is ten; all $11 - s$ available components of layer s are retained, with no truncation along that dimension. The 45 higher-order SH channels use a rank-36 channel subspace, while decoding still outputs all 45 channels. For non-SH state bases and the per-Gaussian SH-channel coefficients, we apply symmetric INT8 quantization independently to each channel within blocks of 64 Gaussian rows:

$$s_{bc} = \frac{\max_{i \in b} |x_{ic}|}{127}, \qquad q_{ic} = \mathrm{clip}\left(\mathrm{round}\left(\frac{x_{ic}}{s_{bc}}\right), -127, 127\right), \qquad \widehat{x}_{ic} = q_{ic}\widehat{s}_{bc}. \tag{8}$$

An all-zero block uses scale one. Quantization scales, anchor combination coefficients, and higher-order SH channel bases use FP16. Atlas construction consists of decomposition, quantization, and storage; the compressed states receive no further image optimization.

A.2 Interval Calibration and Budget Decoding

For each neighboring anchor pair, we enumerate the shared-state exponent p and entering-member gate exponent q over

$$\mathcal{P} = \{0.25, 0.5, 0.75, 1, 1.5, 2, 3, 4, 6\}, \qquad \mathcal{Q} = \{0.25, 0.5, 0.75, 1, 1.25, 1.5, 2, 3\}. \tag{9}$$

At rounded budgets corresponding to $t \in \{0.25, 0.5, 0.75\}$, we compare all 72 pairs on the same set of up to ten training views. The views are midpoint selections from equally partitioned intervals of the training-camera list. Calibration uses FP16 anchor states, does not update Gaussian parameters, and uses no test views.

Let $\ell_{v,h}$ be the reconstruction loss for view v at the lower endpoint, three intermediate budgets, and upper endpoint, indexed by $h = 0, \ldots, 4$. Let t_h be the actual normalized progress of each rounded count. For V selected views, define Q_+ as the mean of the largest $\lceil V/4 \rceil$ values in a set:

$$\begin{aligned} d_{v,h} &= \frac{\ell_{v,h} - \ell_{v,h+1}}{t_{h+1} - t_h}, \\ m_v &= \frac{1}{4}\sum_{h=0}^{3} \max(-d_{v,h}, 0), \qquad s_v = \frac{1}{4}\sum_{h=0}^{3} |d_{v,h} - (\ell_{v,0} - \ell_{v,4})|, \\ J(p,q) &= \frac{1}{3V}\sum_{v=1}^{V}\sum_{h=1}^{3} \ell_{v,h} \\ &\quad + Q_+(\{m_v\}) + 0.25 Q_+(\{s_v\}). \end{aligned} \tag{10}$$

The image loss is $\ell = 0.8L_1 + 0.2(1 - \mathrm{SSIM})$. Each interval stores the minimizing p, q; an exact tie favors larger p and then larger q. No per-Gaussian progress correction is stored. Deployment restores endpoints from the compressed atlas while retaining the calibrated exponents.

An exact anchor request restores that anchor, and $K = N$ uses Full. For other counts, shared members follow Equation (4) in raw parameter coordinates. Entering members use their upper-anchor state, multiply their activated opacity by t^q, clip it to $[10^{-6}, 1 - 10^{-6}]$, and convert it back to a logit. The decoder outputs standard K-Gaussian parameters. The deployed asset comprises the reordered Full, compressed atlas, interval exponents, and member-order metadata; by default, the two most recently used endpoints are cached.

B Per-Scene Experimental Results

The following tables expand Tables 1 and 2 into per-scene quality, Gaussian counts, and training, encoding, calibration, packaging, and full-construction time.

Table 5: Per-scene PSNR (dB) at retention ratios from each method's own Full.

Scene	Method	100%	30%	10%	5%	1%	0.1%
Bicycle	CLOD-3DGS	25.537	25.169	23.829	22.546	16.328	10.199
Bicycle	MGS	25.497	24.733	22.350	20.877	18.761	14.280
Bicycle	ParetoGS	25.867	25.814	25.183	24.627	23.362	21.619
Flowers	CLOD-3DGS	21.869	21.466	20.155	18.955	12.234	7.729
Flowers	MGS	21.888	21.111	18.725	17.646	15.402	11.733
Flowers	ParetoGS	21.788	21.835	21.234	20.656	19.245	17.717
Garden	CLOD-3DGS	27.934	27.537	25.830	23.928	15.276	8.157
Garden	MGS	27.950	27.296	24.848	22.678	18.484	13.988
Garden	ParetoGS	28.025	27.930	26.647	25.703	23.636	21.519
Stump	CLOD-3DGS	27.162	26.948	25.773	24.372	16.648	12.076
Stump	MGS	27.120	26.616	24.447	22.326	20.050	16.773
Stump	ParetoGS	27.239	27.314	26.902	26.342	24.469	21.983
Treehill	CLOD-3DGS	23.044	22.880	22.310	21.511	12.872	8.652
Treehill	MGS	22.880	22.136	20.288	19.538	17.440	12.686
Treehill	ParetoGS	22.811	23.158	22.974	22.750	22.018	20.804
Room	CLOD-3DGS	32.355	32.026	30.221	24.196	11.958	8.869
Room	MGS	32.047	31.798	30.715	29.555	24.186	16.052
Room	ParetoGS	32.601	32.552	31.835	31.128	28.614	23.364
Counter	CLOD-3DGS	29.474	29.094	26.945	20.856	11.980	8.911
Counter	MGS	29.841	29.494	28.212	26.881	21.951	15.694
Counter	ParetoGS	29.672	29.656	28.858	28.027	25.527	21.366
Kitchen	CLOD-3DGS	32.064	31.565	29.131	23.953	11.906	6.197
Kitchen	MGS	32.333	31.996	30.463	28.703	22.799	16.113
Kitchen	ParetoGS	32.520	32.145	30.283	28.682	25.165	21.171
Bonsai	CLOD-3DGS	32.599	32.132	28.423	20.458	12.091	9.536
Bonsai	MGS	32.980	32.546	30.702	28.651	23.337	17.093
Bonsai	ParetoGS	32.872	32.893	31.276	29.813	26.304	22.270
Train	CLOD-3DGS	22.332	21.765	19.213	14.797	8.522	5.663
Train	MGS	23.005	22.483	21.173	19.856	16.863	12.212
Train	ParetoGS	22.357	22.482	22.205	21.858	20.409	17.478
Truck	CLOD-3DGS	26.455	26.198	24.795	22.455	11.061	5.611
Truck	MGS	26.403	26.150	24.818	23.128	19.033	13.839
Truck	ParetoGS	26.439	26.524	25.984	25.315	23.115	19.468
Playroom	CLOD-3DGS	28.518	28.865	28.482	26.992	11.284	5.449
Playroom	MGS	29.661	29.661	29.492	28.916	23.090	14.958
Playroom	ParetoGS	30.677	30.322	30.263	29.997	27.688	22.414

Scene	Method	100%	30%	10%	5%	1%	0.1%
DrJohnson	CLOD-3DGS	28.611	28.523	27.607	25.930	15.072	8.885
DrJohnson	MGS	29.154	29.088	28.397	27.415	23.181	16.903
DrJohnson	ParetoGS	29.725	29.811	29.609	29.301	27.680	23.746

Table 6: Per-scene SSIM at retention ratios from each method's own Full.

Scene	Method	100%	30%	10%	5%	1%	0.1%
Bicycle	CLOD-3DGS	0.79261	0.76801	0.67969	0.59452	0.35390	0.07284
Bicycle	MGS	0.79385	0.74920	0.56546	0.45685	0.34595	0.22454
Bicycle	ParetoGS	0.80498	0.79879	0.75313	0.70719	0.57283	0.43879
Flowers	CLOD-3DGS	0.63806	0.60470	0.51455	0.43602	0.21838	0.03950
Flowers	MGS	0.64514	0.60220	0.44273	0.36447	0.25452	0.15963
Flowers	ParetoGS	0.64000	0.62722	0.56838	0.51652	0.39069	0.28587
Garden	CLOD-3DGS	0.87770	0.86294	0.79471	0.70624	0.37595	0.09642
Garden	MGS	0.87889	0.85618	0.73246	0.58870	0.37961	0.25744
Garden	ParetoGS	0.88218	0.87387	0.82070	0.76825	0.60391	0.43280
Stump	CLOD-3DGS	0.79911	0.78731	0.72499	0.64857	0.33698	0.07784
Stump	MGS	0.79963	0.78055	0.66092	0.51937	0.38037	0.29144
Stump	ParetoGS	0.81182	0.81582	0.78998	0.75323	0.61841	0.44728
Treehill	CLOD-3DGS	0.65297	0.63109	0.57198	0.51870	0.27923	0.06283
Treehill	MGS	0.65224	0.59945	0.43494	0.40345	0.35798	0.25855
Treehill	ParetoGS	0.65619	0.65674	0.62378	0.59093	0.50092	0.42837
Room	CLOD-3DGS	0.93010	0.92262	0.89384	0.81113	0.40764	0.10121
Room	MGS	0.93281	0.92690	0.90259	0.87710	0.77958	0.57827
Room	ParetoGS	0.93325	0.92923	0.91199	0.89650	0.84974	0.76413
Counter	CLOD-3DGS	0.91645	0.90466	0.85259	0.71966	0.36057	0.07319
Counter	MGS	0.92304	0.91509	0.88410	0.84647	0.70538	0.54729
Counter	ParetoGS	0.91914	0.91515	0.89164	0.86727	0.78764	0.68573
Kitchen	CLOD-3DGS	0.93325	0.92447	0.88691	0.78814	0.39201	0.05545
Kitchen	MGS	0.93624	0.93004	0.89751	0.86209	0.66448	0.44705
Kitchen	ParetoGS	0.93661	0.92904	0.89668	0.86488	0.76615	0.62506
Bonsai	CLOD-3DGS	0.94725	0.94019	0.88963	0.72210	0.37173	0.09648
Bonsai	MGS	0.95214	0.94709	0.92045	0.87976	0.73176	0.56222
Bonsai	ParetoGS	0.94970	0.94682	0.92517	0.89993	0.80507	0.69214
Train	CLOD-3DGS	0.83021	0.80069	0.69605	0.53329	0.25079	0.03808
Train	MGS	0.85789	0.83899	0.76425	0.69018	0.54188	0.39130
Train	ParetoGS	0.84418	0.83871	0.80097	0.76702	0.66949	0.52974
Truck	CLOD-3DGS	0.90081	0.89134	0.85240	0.78764	0.42194	0.10863
Truck	MGS	0.90219	0.88975	0.83885	0.77649	0.61727	0.45786
Truck	ParetoGS	0.89913	0.89750	0.87885	0.85681	0.77162	0.62278
Playroom	CLOD-3DGS	0.89643	0.90135	0.89230	0.87239	0.58054	0.09448
Playroom	MGS	0.91095	0.91095	0.90434	0.89329	0.80696	0.67308
Playroom	ParetoGS	0.91521	0.90951	0.90602	0.89958	0.86717	0.79749
DrJohnson	CLOD-3DGS	0.90240	0.89822	0.87833	0.84767	0.62353	0.08275
DrJohnson	MGS	0.90601	0.90438	0.88658	0.86617	0.78142	0.66518
DrJohnson	ParetoGS	0.91222	0.91229	0.90652	0.89782	0.86075	0.79180

Table 7: Per-scene LPIPS at retention ratios from each method's own Full.

Scene	Method	100%	30%	10%	5%	1%	0.1%
Bicycle	CLOD-3DGS	0.17114	0.20779	0.30996	0.38589	0.56435	0.72614
Bicycle	MGS	0.16702	0.22563	0.41849	0.52812	0.61848	0.68876
Bicycle	ParetoGS	0.16128	0.19595	0.28717	0.34258	0.46341	0.58365
Flowers	CLOD-3DGS	0.28766	0.32940	0.41244	0.47628	0.65360	0.81904
Flowers	MGS	0.26585	0.30913	0.43901	0.51370	0.64526	0.75778
Flowers	ParetoGS	0.28129	0.33218	0.41760	0.47123	0.57560	0.69107
Garden	CLOD-3DGS	0.09880	0.12293	0.22229	0.31907	0.56343	0.76212
Garden	MGS	0.09619	0.13083	0.28811	0.42179	0.58730	0.70568
Garden	ParetoGS	0.09022	0.11352	0.22346	0.30082	0.44834	0.58494
Stump	CLOD-3DGS	0.18383	0.20569	0.28685	0.36379	0.55394	0.69418
Stump	MGS	0.17769	0.20675	0.34077	0.47090	0.60834	0.66997
Stump	ParetoGS	0.17251	0.18652	0.25487	0.31370	0.45016	0.59703
Treehill	CLOD-3DGS	0.27732	0.32187	0.41356	0.47860	0.62833	0.71543
Treehill	MGS	0.26310	0.33518	0.53229	0.57749	0.62943	0.69155
Treehill	ParetoGS	0.28162	0.33528	0.41557	0.46661	0.55726	0.61555
Room	CLOD-3DGS	0.20025	0.21898	0.27095	0.35374	0.57175	0.66292
Room	MGS	0.19054	0.20300	0.24736	0.28849	0.40629	0.56538
Room	ParetoGS	0.19425	0.20737	0.25362	0.28628	0.36318	0.46482
Counter	CLOD-3DGS	0.18790	0.21014	0.27647	0.38617	0.57277	0.66291
Counter	MGS	0.16789	0.18065	0.22480	0.27351	0.42756	0.57307
Counter	ParetoGS	0.18040	0.19330	0.24015	0.27888	0.39318	0.51456
Kitchen	CLOD-3DGS	0.12336	0.13960	0.19589	0.29849	0.58857	0.73865
Kitchen	MGS	0.11492	0.12581	0.18113	0.22869	0.42844	0.60992
Kitchen	ParetoGS	0.11511	0.13074	0.19946	0.25229	0.37615	0.48895
Bonsai	CLOD-3DGS	0.19561	0.21207	0.28329	0.40940	0.56852	0.61220
Bonsai	MGS	0.18079	0.19048	0.23022	0.28300	0.43977	0.56284
Bonsai	ParetoGS	0.18784	0.19815	0.24410	0.28616	0.40082	0.51535
Train	CLOD-3DGS	0.19937	0.24103	0.34966	0.47190	0.64094	0.70284
Train	MGS	0.14560	0.17087	0.26314	0.34663	0.49265	0.62553
Train	ParetoGS	0.17111	0.19657	0.26342	0.30739	0.41391	0.55754
Truck	CLOD-3DGS	0.11095	0.13523	0.19847	0.27258	0.54939	0.70919
Truck	MGS	0.09557	0.12694	0.22212	0.30812	0.47345	0.61349
Truck	ParetoGS	0.11788	0.12269	0.17115	0.21610	0.33516	0.48500
Playroom	CLOD-3DGS	0.27185	0.27143	0.29705	0.32752	0.58480	0.83783
Playroom	MGS	0.23329	0.23329	0.25475	0.28132	0.40722	0.56112
Playroom	ParetoGS	0.22893	0.24123	0.26584	0.28806	0.35820	0.45663
DrJohnson	CLOD-3DGS	0.25385	0.26731	0.30645	0.34893	0.50995	0.60819
DrJohnson	MGS	0.23741	0.24125	0.27436	0.30986	0.42070	0.53314
DrJohnson	ParetoGS	0.22800	0.23725	0.26172	0.28444	0.35426	0.45505

Table 8: Per-scene PSNR (dB) at requested ratios from the shared Full.

Scene	Method	30%	10%	5%	1%	0.1%
Bicycle	FlexGS-imp	20.619	23.210	22.670	21.684	15.972
Bicycle	NanoGS-imp	21.363	19.053	18.259	17.293	16.405
Bicycle	ParetoGS	25.814	25.183	24.627	23.362	21.619

Scene	Method	30%	10%	5%	1%	0.1%
Flowers	FlexGS-imp	16.993	19.450	18.870	17.997	14.407
Flowers	NanoGS-imp	18.161	16.735	16.257	15.427	14.022
Flowers	ParetoGS	21.835	21.234	20.656	19.245	17.717
Garden	FlexGS-imp	21.240	24.063	22.887	21.403	12.820
Garden	NanoGS-imp	22.513	20.120	19.152	18.064	16.057
Garden	ParetoGS	27.930	26.647	25.703	23.636	21.519
Stump	FlexGS-imp	17.635	23.554	22.377	21.073	12.172
Stump	NanoGS-imp	23.147	20.588	19.886	19.033	17.708
Stump	ParetoGS	27.314	26.902	26.342	24.469	21.983
Treehill	FlexGS-imp	20.925	21.846	21.613	20.956	17.422
Treehill	NanoGS-imp	19.502	18.318	18.028	17.726	16.504
Treehill	ParetoGS	23.158	22.974	22.750	22.018	20.804
Room	FlexGS-imp	25.306	29.990	28.473	24.947	17.473
Room	NanoGS-imp	28.535	24.936	23.816	21.511	17.647
Room	ParetoGS	32.552	31.835	31.128	28.614	23.364
Counter	FlexGS-imp	25.028	27.327	25.936	22.723	15.761
Counter	NanoGS-imp	25.978	23.320	22.052	20.033	16.734
Counter	ParetoGS	29.656	28.858	28.027	25.527	21.366
Kitchen	FlexGS-imp	21.594	27.309	25.362	22.494	14.473
Kitchen	NanoGS-imp	26.373	23.203	21.831	19.492	16.647
Kitchen	ParetoGS	32.145	30.283	28.682	25.165	21.171
Bonsai	FlexGS-imp	26.809	29.347	27.371	23.247	17.997
Bonsai	NanoGS-imp	27.722	23.975	22.735	20.697	17.137
Bonsai	ParetoGS	32.893	31.276	29.813	26.304	22.270
Train	FlexGS-imp	19.545	21.207	20.268	18.108	10.241
Train	NanoGS-imp	18.810	16.477	15.326	13.888	12.591
Train	ParetoGS	22.482	22.205	21.858	20.409	17.478
Truck	FlexGS-imp	18.948	22.522	21.259	18.856	7.054
Truck	NanoGS-imp	22.506	19.853	18.644	16.872	14.604
Truck	ParetoGS	26.524	25.984	25.315	23.115	19.468
Playroom	FlexGS-imp	25.445	29.283	27.796	24.320	8.957
Playroom	NanoGS-imp	28.479	25.503	24.366	22.548	17.880
Playroom	ParetoGS	30.322	30.263	29.997	27.688	22.414
DrJohnson	FlexGS-imp	25.650	28.608	27.522	24.925	11.281
DrJohnson	NanoGS-imp	27.119	24.619	24.009	22.376	18.840
DrJohnson	ParetoGS	29.811	29.609	29.301	27.680	23.746

Table 9: Per-scene SSIM at requested ratios from the shared Full.

Scene	Method	30%	10%	5%	1%	0.1%
Bicycle	FlexGS-imp	0.61926	0.66428	0.59500	0.47001	0.31088
Bicycle	NanoGS-imp	0.55555	0.41020	0.36755	0.32473	0.29897
Bicycle	ParetoGS	0.79879	0.75313	0.70719	0.57283	0.43879
Flowers	FlexGS-imp	0.37608	0.43193	0.39087	0.32018	0.23142
Flowers	NanoGS-imp	0.42791	0.32584	0.29242	0.24842	0.22208
Flowers	ParetoGS	0.62722	0.56838	0.51652	0.39069	0.28587
Garden	FlexGS-imp	0.67256	0.70674	0.62390	0.47661	0.25187
Garden	NanoGS-imp	0.60245	0.44745	0.39251	0.33304	0.30137
Garden	ParetoGS	0.87387	0.82070	0.76825	0.60391	0.43280

Scene	Method	30%	10%	5%	1%	0.1%
Stump	FlexGS-imp	0.44183	0.68332	0.59999	0.48821	0.20003
Stump	NanoGS-imp	0.60730	0.45477	0.41086	0.36538	0.33798
Stump	ParetoGS	0.81582	0.78998	0.75323	0.61841	0.44728
Treehill	FlexGS-imp	0.57951	0.57752	0.53007	0.45139	0.36801
Treehill	NanoGS-imp	0.45548	0.38411	0.37190	0.36674	0.35114
Treehill	ParetoGS	0.65674	0.62378	0.59093	0.50092	0.42837
Room	FlexGS-imp	0.83081	0.88825	0.85924	0.79807	0.66359
Room	NanoGS-imp	0.85213	0.77523	0.74734	0.69542	0.62462
Room	ParetoGS	0.92923	0.91199	0.89650	0.84974	0.76413
Counter	FlexGS-imp	0.83051	0.85606	0.81562	0.73345	0.59456
Counter	NanoGS-imp	0.81725	0.72237	0.68238	0.63553	0.57553
Counter	ParetoGS	0.91515	0.89164	0.86727	0.78764	0.68573
Kitchen	FlexGS-imp	0.77631	0.86065	0.81331	0.69541	0.46453
Kitchen	NanoGS-imp	0.81817	0.68709	0.61681	0.49595	0.46645
Kitchen	ParetoGS	0.92904	0.89668	0.86488	0.76615	0.62506
Bonsai	FlexGS-imp	0.87955	0.89264	0.84060	0.73134	0.64543
Bonsai	NanoGS-imp	0.86468	0.75731	0.71240	0.64353	0.56922
Bonsai	ParetoGS	0.94682	0.92517	0.89993	0.80507	0.69214
Train	FlexGS-imp	0.72859	0.74651	0.69643	0.59453	0.37843
Train	NanoGS-imp	0.69435	0.56958	0.52039	0.46376	0.42745
Train	ParetoGS	0.83871	0.80097	0.76702	0.66949	0.52974
Truck	FlexGS-imp	0.74132	0.79470	0.74921	0.65902	0.23108
Truck	NanoGS-imp	0.76656	0.64622	0.59535	0.52496	0.48572
Truck	ParetoGS	0.89750	0.87885	0.85681	0.77162	0.62278
Playroom	FlexGS-imp	0.84991	0.89749	0.87780	0.83081	0.41377
Playroom	NanoGS-imp	0.87862	0.83735	0.81667	0.78672	0.73227
Playroom	ParetoGS	0.90951	0.90602	0.89958	0.86717	0.79749
DrJohnson	FlexGS-imp	0.84029	0.88360	0.86175	0.81822	0.34511
DrJohnson	NanoGS-imp	0.86790	0.81622	0.79911	0.76315	0.71350
DrJohnson	ParetoGS	0.91229	0.90652	0.89782	0.86075	0.79180

Table 10: Per-scene LPIPS at requested ratios from the shared Full.

Scene	Method	30%	10%	5%	1%	0.1%
Bicycle	FlexGS-imp	0.34589	0.34941	0.41595	0.54225	0.68721
Bicycle	NanoGS-imp	0.36029	0.48012	0.53489	0.64970	0.73212
Bicycle	ParetoGS	0.19595	0.28717	0.34258	0.46341	0.58365
Flowers	FlexGS-imp	0.55582	0.52457	0.56194	0.63987	0.76824
Flowers	NanoGS-imp	0.45624	0.55264	0.59844	0.71282	0.77451
Flowers	ParetoGS	0.33218	0.41760	0.47123	0.57560	0.69107
Garden	FlexGS-imp	0.29432	0.30381	0.38939	0.53023	0.70719
Garden	NanoGS-imp	0.31168	0.43730	0.49704	0.61566	0.75966
Garden	ParetoGS	0.11352	0.22346	0.30082	0.44834	0.58494
Stump	FlexGS-imp	0.53284	0.33107	0.41509	0.52423	0.66585
Stump	NanoGS-imp	0.34062	0.46998	0.52486	0.63369	0.70894
Stump	ParetoGS	0.18652	0.25487	0.31370	0.45016	0.59703
Treehill	FlexGS-imp	0.40792	0.45551	0.51746	0.59366	0.66525
Treehill	NanoGS-imp	0.48848	0.57233	0.61059	0.68148	0.67097
Treehill	ParetoGS	0.33528	0.41557	0.46661	0.55726	0.61555

Scene	Method	30%	10%	5%	1%	0.1%
Room	FlexGS-imp	0.33105	0.29201	0.34287	0.42541	0.51834
Room	NanoGS-imp	0.29862	0.38245	0.42232	0.50196	0.53753
Room	ParetoGS	0.20737	0.25362	0.28628	0.36318	0.46482
Counter	FlexGS-imp	0.29191	0.28713	0.34496	0.44483	0.55609
Counter	NanoGS-imp	0.28956	0.39133	0.43882	0.52556	0.57303
Counter	ParetoGS	0.19330	0.24015	0.27888	0.39318	0.51456
Kitchen	FlexGS-imp	0.29614	0.24986	0.30965	0.43323	0.62703
Kitchen	NanoGS-imp	0.24310	0.37516	0.44214	0.57297	0.64835
Kitchen	ParetoGS	0.13074	0.19946	0.25229	0.37615	0.48895
Bonsai	FlexGS-imp	0.27572	0.28756	0.34871	0.46322	0.53580
Bonsai	NanoGS-imp	0.28637	0.39702	0.44536	0.52613	0.57159
Bonsai	ParetoGS	0.19815	0.24410	0.28616	0.40082	0.51535
Train	FlexGS-imp	0.29608	0.30840	0.36571	0.48042	0.64700
Train	NanoGS-imp	0.32279	0.45304	0.51193	0.59871	0.63349
Train	ParetoGS	0.19657	0.26342	0.30739	0.41391	0.55754
Truck	FlexGS-imp	0.28975	0.26604	0.32905	0.44347	0.69049
Truck	NanoGS-imp	0.26590	0.39598	0.45955	0.56911	0.65719
Truck	ParetoGS	0.12269	0.17115	0.21610	0.33516	0.48500
Playroom	FlexGS-imp	0.34355	0.29683	0.33552	0.41137	0.69256
Playroom	NanoGS-imp	0.29749	0.36644	0.39885	0.45569	0.49818
Playroom	ParetoGS	0.24123	0.26584	0.28806	0.35820	0.45663
DrJohnson	FlexGS-imp	0.35177	0.30173	0.33901	0.40949	0.57383
DrJohnson	NanoGS-imp	0.31454	0.40023	0.43527	0.48716	0.51959
DrJohnson	ParetoGS	0.23725	0.26172	0.28444	0.35426	0.45505

Table 11: Per-scene actual Gaussian counts (K) at retention ratios from each method's own Full.

Scene	Method	100%	30%	10%	5%	1%	0.1%
Bicycle	CLOD-3DGS	6132	1840	613	307	61	6
Bicycle	MGS	5000	1500	500	250	50	5
Bicycle	ParetoGS	3000	900	300	150	30	3
Flowers	CLOD-3DGS	3636	1091	364	182	36	4
Flowers	MGS	5000	1500	500	250	50	5
Flowers	ParetoGS	1500	450	150	75	15	2
Garden	CLOD-3DGS	5835	1750	583	292	58	6
Garden	MGS	5000	1500	500	250	50	5
Garden	ParetoGS	3000	900	300	150	30	3
Stump	CLOD-3DGS	4962	1489	496	248	50	5
Stump	MGS	5000	1500	500	250	50	5
Stump	ParetoGS	3000	900	300	150	30	3
Treehill	CLOD-3DGS	3784	1135	378	189	38	4
Treehill	MGS	5000	1500	500	250	50	5
Treehill	ParetoGS	1500	450	150	75	15	2
Room	CLOD-3DGS	1593	478	159	80	16	2
Room	MGS	5000	1500	500	250	50	5
Room	ParetoGS	1000	300	100	50	10	1
Counter	CLOD-3DGS	1223	367	122	61	12	1
Counter	MGS	5000	1500	500	250	50	5
Counter	ParetoGS	1000	300	100	50	10	1

Scene	Method	100%	30%	10%	5%	1%	0.1%
Kitchen	CLOD-3DGS	1852	556	185	93	19	2
Kitchen	MGS	5000	1500	500	250	50	5
Kitchen	ParetoGS	1000	300	100	50	10	1
Bonsai	CLOD-3DGS	1245	373	124	62	12	1
Bonsai	MGS	5000	1500	500	250	50	5
Bonsai	ParetoGS	1000	300	100	50	10	1
Train	CLOD-3DGS	1027	308	103	51	10	1
Train	MGS	5000	1500	500	250	50	5
Train	ParetoGS	1000	300	100	50	10	1
Truck	CLOD-3DGS	2541	762	254	127	25	3
Truck	MGS	5000	1500	500	250	50	5
Truck	ParetoGS	1500	450	150	75	15	2
Playroom	CLOD-3DGS	2546	764	255	127	25	3
Playroom	MGS	5000	1500	500	250	50	5
Playroom	ParetoGS	1000	300	100	50	10	1
DrJohnson	CLOD-3DGS	3000	900	300	150	30	3
DrJohnson	MGS	4000	1200	400	200	40	4
DrJohnson	ParetoGS	1500	450	150	75	15	2

Table 12: Per-scene requested and FlexGS-imp actual output counts (K) from the shared Full.

Scene	Count	30%	10%	5%	1%	0.1%
Bicycle	Requested	900	300	150	30	3
Bicycle	FlexGS actual	900	300	150	30	3
Flowers	Requested	450	150	75	15	2
Flowers	FlexGS actual	450	150	75	15	2
Garden	Requested	900	300	150	30	3
Garden	FlexGS actual	900	300	150	30	3
Stump	Requested	900	300	150	30	3
Stump	FlexGS actual	900	300	150	30	3
Treehill	Requested	450	150	75	15	2
Treehill	FlexGS actual	450	150	75	15	3
Room	Requested	300	100	50	10	1
Room	FlexGS actual	300	100	50	10	3
Counter	Requested	300	100	50	10	1
Counter	FlexGS actual	300	100	50	10	3
Kitchen	Requested	300	100	50	10	1
Kitchen	FlexGS actual	300	100	50	10	1
Bonsai	Requested	300	100	50	10	1
Bonsai	FlexGS actual	300	100	50	10	5
Train	Requested	300	100	50	10	1
Train	FlexGS actual	300	100	50	10	1
Truck	Requested	450	150	75	15	2
Truck	FlexGS actual	450	150	75	15	2
Playroom	Requested	300	100	50	10	1
Playroom	FlexGS actual	300	100	50	10	1
DrJohnson	Requested	450	150	75	15	2
DrJohnson	FlexGS actual	450	150	75	15	2

Table 13: Per-scene NanoGS-imp offline conversion time (seconds) from the shared Full to each target PLY.

Scene	30%	10%	5%	1%	0.1%
Bicycle	99	108	110	111	110
Flowers	49	57	57	58	59
Garden	96	106	111	110	110
Stump	87	101	103	104	104
Treehill	49	56	57	58	59
Room	28	32	32	33	34
Counter	27	30	31	31	31
Kitchen	28	31	32	32	32
Bonsai	26	27	29	30	30
Train	28	31	32	32	32
Truck	37	39	41	42	42
Playroom	29	33	33	34	34
DrJohnson	46	52	53	54	54

Table 14: Per-scene offline time. CLOD, MGS, FlexGS, ParetoGS training, and total are in minutes; encoding, calibration, and packaging are in seconds. ParetoGS total includes its training and all three post-processing stages.

Scene	CLOD	MGS	FlexGS	Training	Encoding	Calibration	Packaging	Total
Bicycle	57.24	68.74	120.18	12.28	1.98	31.91	3.98	12.92
Flowers	38.10	71.78	74.74	9.29	1.60	26.29	2.43	9.80
Garden	56.35	72.58	123.04	13.53	1.83	33.10	3.86	14.18
Stump	46.94	68.18	119.88	12.12	1.74	29.84	3.75	12.71
Treehill	38.46	68.18	72.29	8.79	2.07	24.74	2.41	9.27
Room	29.36	82.03	69.89	9.62	1.43	33.23	1.91	10.23
Counter	27.92	89.00	70.15	10.91	1.28	33.93	1.90	11.53
Kitchen	32.91	91.10	73.24	10.51	1.35	34.16	1.88	11.14
Bonsai	24.78	85.42	72.10	9.85	1.29	33.78	1.89	10.46
Train	14.16	76.40	56.66	6.84	1.29	22.17	1.91	7.26
Truck	27.51	75.08	74.65	8.01	1.47	23.76	2.39	8.47
Playroom	27.67	64.02	57.02	7.08	1.24	22.62	1.90	7.51
DrJohnson	36.90	54.52	73.28	8.31	1.43	26.94	2.48	8.83
13-scene mean	35.25	74.39	81.32	9.78	1.54	28.96	2.51	10.33

B.1 Per-Scene Decoder Components

Table 15 expands Table 3 over all 13 scenes.

Table 15: Per-scene results for Table 3. Each scene averages ten interval midpoints; the Full-parameter row uses the same member IDs.

Scene	Variant	PSNR↑	SSIM↑	LPIPS↓
Bicycle	Full method	24.158	0.65435	0.36872
Bicycle	Fixed shared state	24.043	0.65228	0.37010
Bicycle	No opacity gate	24.025	0.65220	0.36863
Bicycle	Linear progress	24.076	0.64939	0.37316
Bicycle	Full parameters	15.574	0.38713	0.49462
Flowers	Full method	20.118	0.47887	0.48940
Flowers	Fixed shared state	20.071	0.47660	0.49148
Flowers	No opacity gate	20.033	0.47788	0.48890
Flowers	Linear progress	20.049	0.47324	0.49429
Flowers	Full parameters	13.843	0.31859	0.58598
Garden	Full method	25.132	0.70119	0.32858
Garden	Fixed shared state	24.944	0.69877	0.32998
Garden	No opacity gate	24.814	0.69712	0.32935
Garden	Linear progress	24.961	0.69475	0.33330
Garden	Full parameters	14.807	0.41770	0.49165
Stump	Full method	25.425	0.68647	0.35596
Stump	Fixed shared state	25.326	0.68406	0.35822
Stump	No opacity gate	25.258	0.68365	0.35620
Stump	Linear progress	25.304	0.68081	0.36058
Stump	Full parameters	17.615	0.41274	0.47134
Treehill	Full method	22.322	0.55971	0.47156
Treehill	Fixed shared state	22.314	0.55851	0.47309
Treehill	No opacity gate	22.248	0.55780	0.47182
Treehill	Linear progress	22.221	0.55575	0.47494
Treehill	Full parameters	15.138	0.38706	0.55845
Room	Full method	29.590	0.87230	0.31181
Room	Fixed shared state	29.402	0.87096	0.31356
Room	No opacity gate	29.363	0.87073	0.31160
Room	Linear progress	29.434	0.87001	0.31380
Room	Full parameters	20.300	0.70125	0.42897
Counter	Full method	26.753	0.83114	0.32172
Counter	Fixed shared state	26.600	0.82943	0.32411
Counter	No opacity gate	26.607	0.82973	0.32106
Counter	Linear progress	26.607	0.82797	0.32427
Counter	Full parameters	18.584	0.63934	0.43900
Kitchen	Full method	27.571	0.82022	0.28496
Kitchen	Fixed shared state	27.325	0.81770	0.28728
Kitchen	No opacity gate	26.754	0.81577	0.28561
Kitchen	Linear progress	27.214	0.81640	0.28646
Kitchen	Full parameters	15.568	0.53158	0.47131
Bonsai	Full method	28.585	0.85504	0.32608
Bonsai	Fixed shared state	28.413	0.85342	0.32767
Bonsai	No opacity gate	28.213	0.85308	0.32559
Bonsai	Linear progress	28.358	0.85147	0.32895
Bonsai	Full parameters	19.324	0.65239	0.43914
Train	Full method	20.926	0.72447	0.34072
Train	Fixed shared state	20.873	0.72239	0.34285
Train	No opacity gate	20.816	0.72269	0.34020
Train	Linear progress	20.837	0.72073	0.34366
Train	Full parameters	14.657	0.54427	0.46297

Scene	Variant	PSNR↑	SSIM↑	LPIPS↓
Truck	Full method	24.117	0.80901	0.26242
Truck	Fixed shared state	23.992	0.80683	0.26516
Truck	No opacity gate	23.971	0.80751	0.26192
Truck	Linear progress	23.998	0.80500	0.26623
Truck	Full parameters	16.535	0.61102	0.40963
Playroom	Full method	28.287	0.87814	0.31911
Playroom	Fixed shared state	28.173	0.87727	0.32097
Playroom	No opacity gate	28.223	0.87772	0.31861
Playroom	Linear progress	28.147	0.87637	0.32141
Playroom	Full parameters	19.887	0.75819	0.43984
DrJohnson	Full method	28.162	0.87622	0.31407
DrJohnson	Fixed shared state	28.022	0.87514	0.31584
DrJohnson	No opacity gate	28.063	0.87541	0.31373
DrJohnson	Linear progress	28.072	0.87457	0.31661
DrJohnson	Full parameters	20.116	0.73512	0.43986

B.2 Per-Scene State Encoding

Table 16 expands Table 4 into per-scene quality and additional state storage.

Table 16: Per-scene results for Table 4. Quality averages ten interval midpoints per scene; MB is additional state storage.

Scene	Encoding	PSNR↑	SSIM↑	LPIPS↓	MB↓
Bicycle	FP32	24.177	0.65499	0.36784	724.303
Bicycle	FP16	24.177	0.65499	0.36784	362.161
Bicycle	Atlas	24.158	0.65435	0.36872	158.666
Flowers	FP32	20.126	0.47915	0.48900	362.161
Flowers	FP16	20.126	0.47915	0.48900	181.090
Flowers	Atlas	20.118	0.47887	0.48940	79.573
Garden	FP32	25.148	0.70159	0.32779	724.303
Garden	FP16	25.148	0.70159	0.32779	362.161
Garden	Atlas	25.132	0.70119	0.32858	158.666
Stump	FP32	25.439	0.68691	0.35514	724.303
Stump	FP16	25.439	0.68691	0.35514	362.161
Stump	Atlas	25.425	0.68647	0.35596	158.666
Treehill	FP32	22.315	0.56004	0.47114	362.161
Treehill	FP16	22.315	0.56004	0.47114	181.090
Treehill	Atlas	22.322	0.55971	0.47156	79.573
Room	FP32	29.604	0.87259	0.31155	241.447
Room	FP16	29.604	0.87259	0.31155	120.733
Room	Atlas	29.590	0.87230	0.31181	53.200
Counter	FP32	26.762	0.83145	0.32140	241.447
Counter	FP16	26.762	0.83145	0.32140	120.733
Counter	Atlas	26.753	0.83114	0.32172	53.200
Kitchen	FP32	27.597	0.82202	0.28387	241.447
Kitchen	FP16	27.597	0.82202	0.28387	120.733
Kitchen	Atlas	27.571	0.82022	0.28496	53.200
Bonsai	FP32	28.601	0.85548	0.32571	241.447
Bonsai	FP16	28.601	0.85548	0.32571	120.733
Bonsai	Atlas	28.585	0.85504	0.32608	53.200

Scene	Encoding	PSNR↑	SSIM↑	LPIPS↓	MB↓
Train	FP32	20.941	0.72530	0.34029	241.447
Train	FP16	20.941	0.72529	0.34029	120.733
Train	Atlas	20.926	0.72447	0.34072	53.200
Truck	FP32	24.129	0.80956	0.26198	362.161
Truck	FP16	24.129	0.80956	0.26198	181.090
Truck	Atlas	24.117	0.80901	0.26242	79.573
Playroom	FP32	28.294	0.87848	0.31876	241.447
Playroom	FP16	28.294	0.87848	0.31876	120.733
Playroom	Atlas	28.287	0.87814	0.31911	53.200
DrJohnson	FP32	28.174	0.87653	0.31368	362.161
DrJohnson	FP16	28.174	0.87653	0.31368	181.090
DrJohnson	Atlas	28.162	0.87622	0.31407	79.573

C BUDGET-SWITCH PARAMETER CONSTRUCTION TIME

We measure the time required to construct Gaussian parameters for a requested budget after the model has been loaded. The four methods use the same 13 scenes and an RTX 4090 GPU, running serially in separate processes. Before timing, CLOD-3DGS and MGS have their Full parameters and prefix orders resident on the GPU; FlexGS-imp has its Full parameters and selector; ParetoGS has all anchor fields predecoded and resident. At a switch, CLOD-3DGS and MGS construct rendering parameters for the requested prefix, FlexGS-imp selects members with its native threshold and applies deformation, and ParetoGS interpolates endpoint states and gates entering members. A Full request to ParetoGS reuses the resident model.

We synchronize CUDA and read host wall time t_0 before a request, then synchronize after target parameters are constructed and read t_1. The reported time is $T_{\text{switch}} = t_1 - t_0$. After each timed request, we render the first test view of that scene before the next request. The trajectory covers 16 ratios from 0.1% through 100%, including forward, reverse, and cross-interval changes. After three repetitions and removing requests with no change in ratio, each method contributes 108 switches per scene and 1,404 across 13 scenes.

Table 17: Synchronized budget-switch parameter preparation time across 13 scenes (milliseconds), pooling 1,404 switches per method.

Method	Median	P95	Maximum
CLOD-3DGS	**0.118**	3.009	7.460
MGS	0.406	7.787	8.099
FlexGS-imp	18.658	46.911	98.869
ParetoGS	0.268	**2.773**	**5.650**

ParetoGS has a median of 0.268 ms, P95 of 2.773 ms, and maximum of 5.650 ms in Table 17, showing low computational cost for endpoint interpolation and parameter construction. Its one-time predecoding of all anchors averages 0.393 s, and the decoded endpoint fields occupy 390.0 MB of GPU storage on average. An application can instead keep decoded endpoints in host memory or on disk and load them as needed according to its GPU memory budget; the table measures interpolation and parameter construction with endpoints already resident on the GPU.

D Cross-Budget Visual Comparisons on Other Scenes

The following figures use the same method columns and budget rows as Figure 1. Each panel labels its actual Gaussian count; two scenes appear on each page.

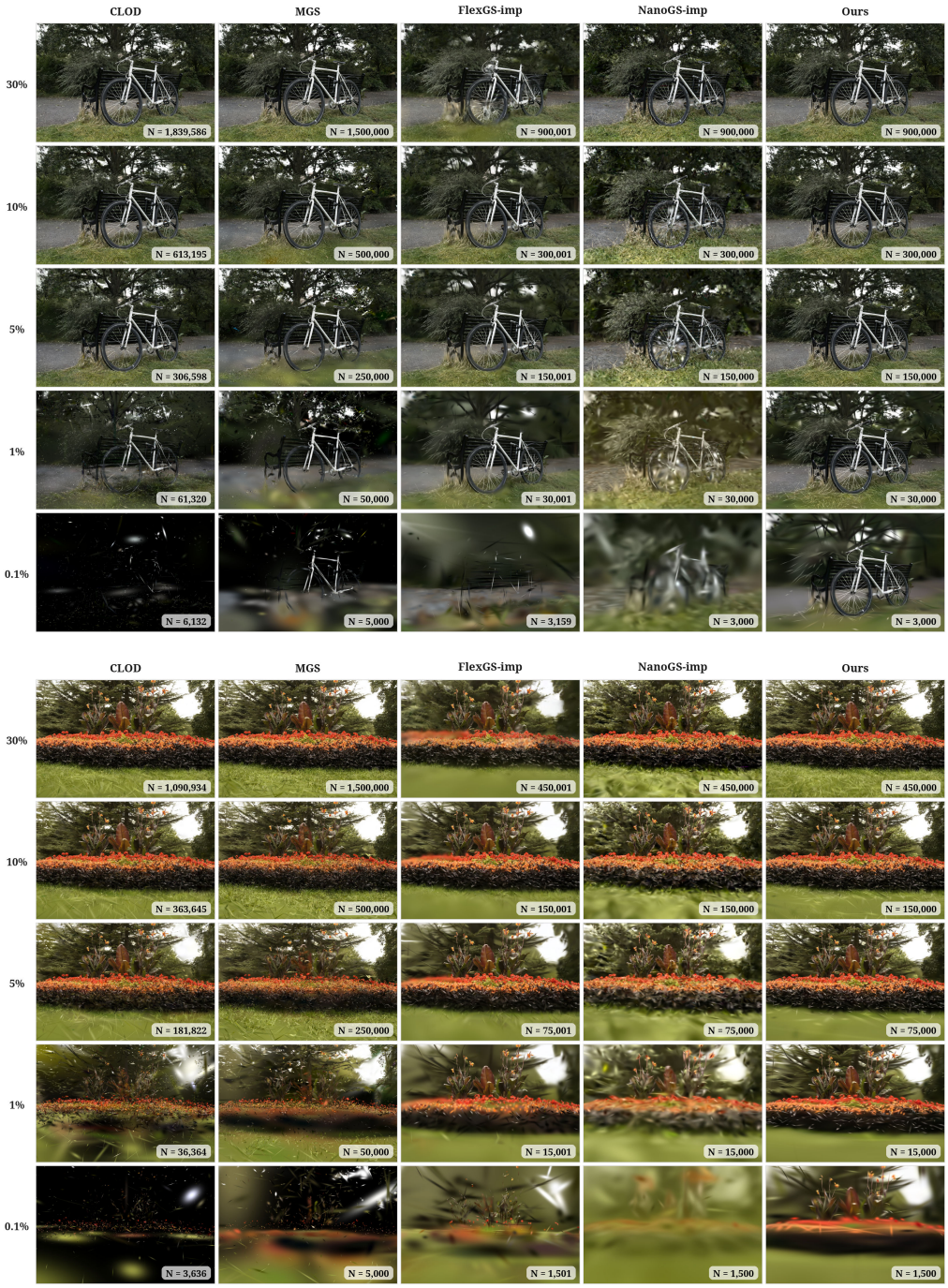


Figure 5: Cross-budget reconstruction on Bicycle (top) and Flowers (bottom).



Figure 6: Cross-budget reconstruction on Stump (top) and Treehill (bottom).

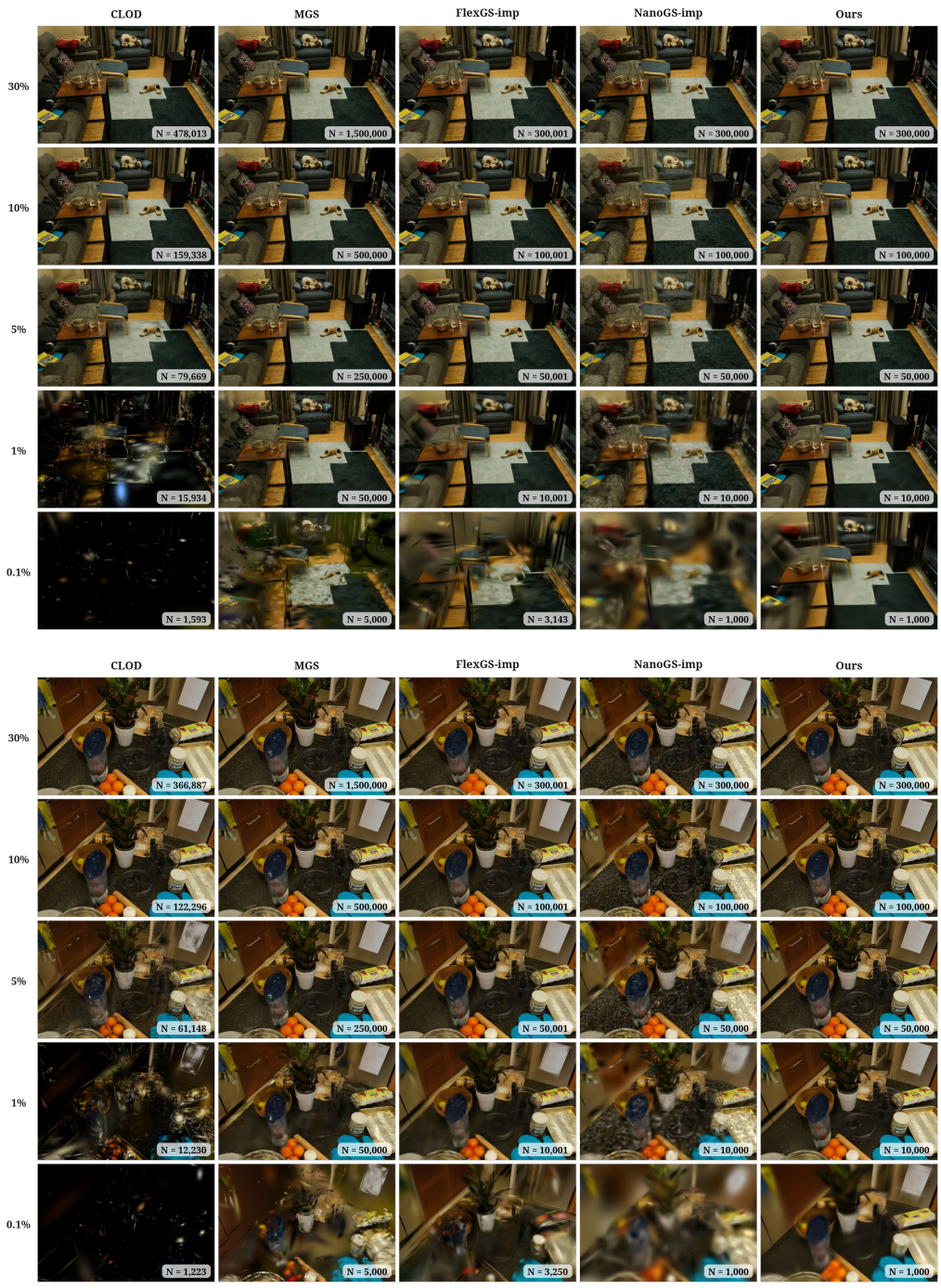


Figure 7: Cross-budget reconstruction on Room (top) and Counter (bottom).

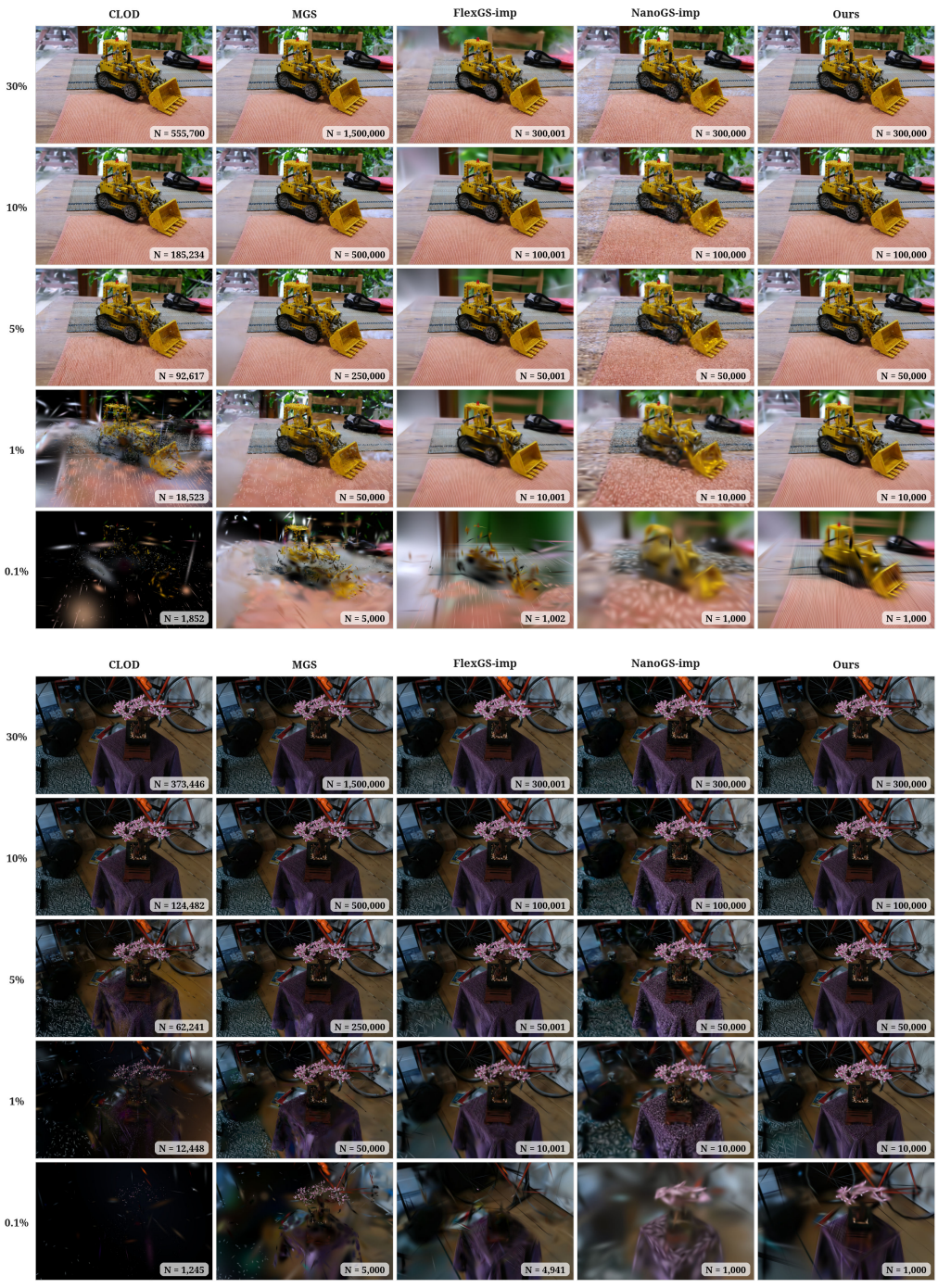


Figure 8: Cross-budget reconstruction on Kitchen (top) and Bonsai (bottom).

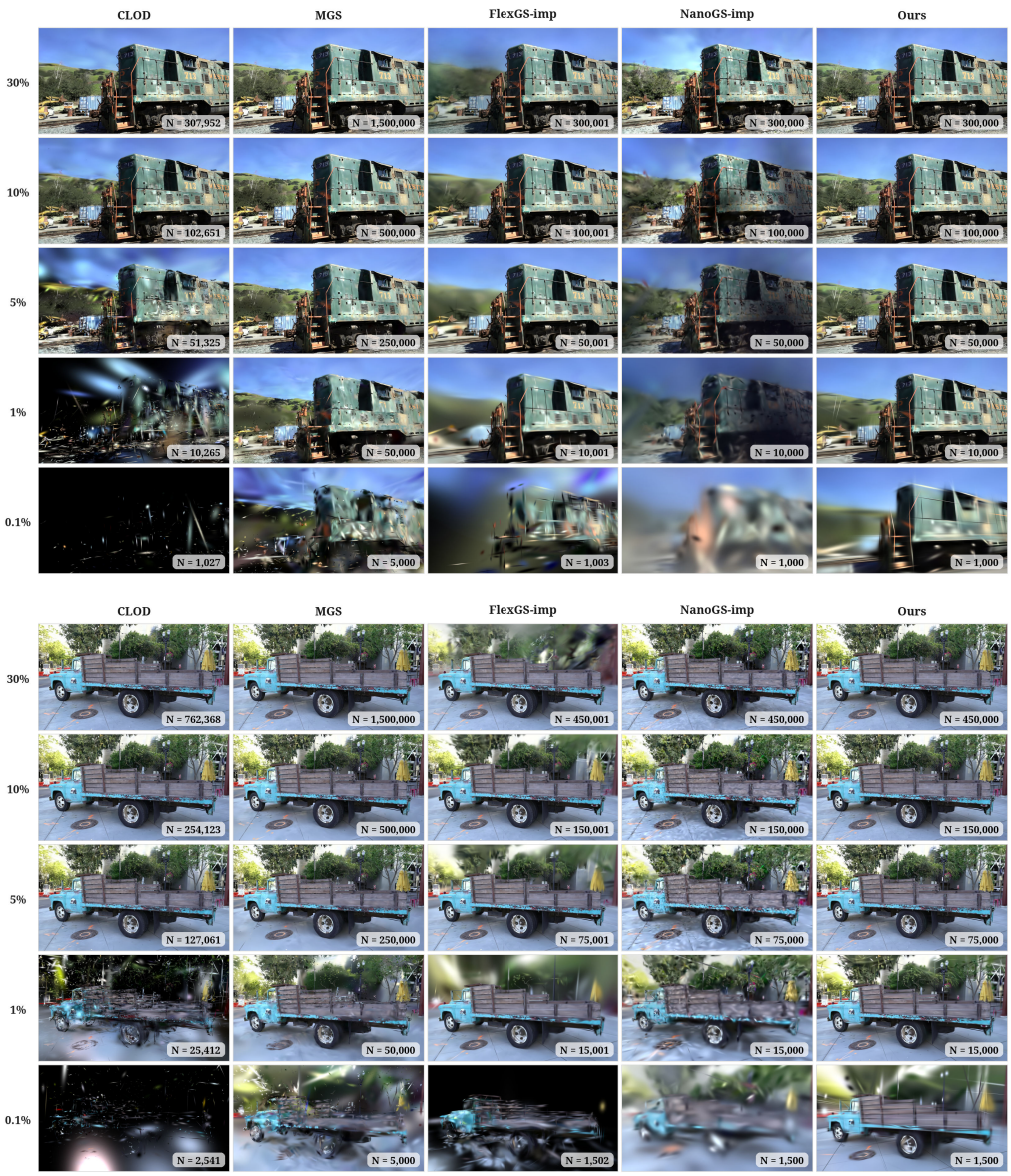


Figure 9: Cross-budget reconstruction on Train (top) and Truck (bottom).

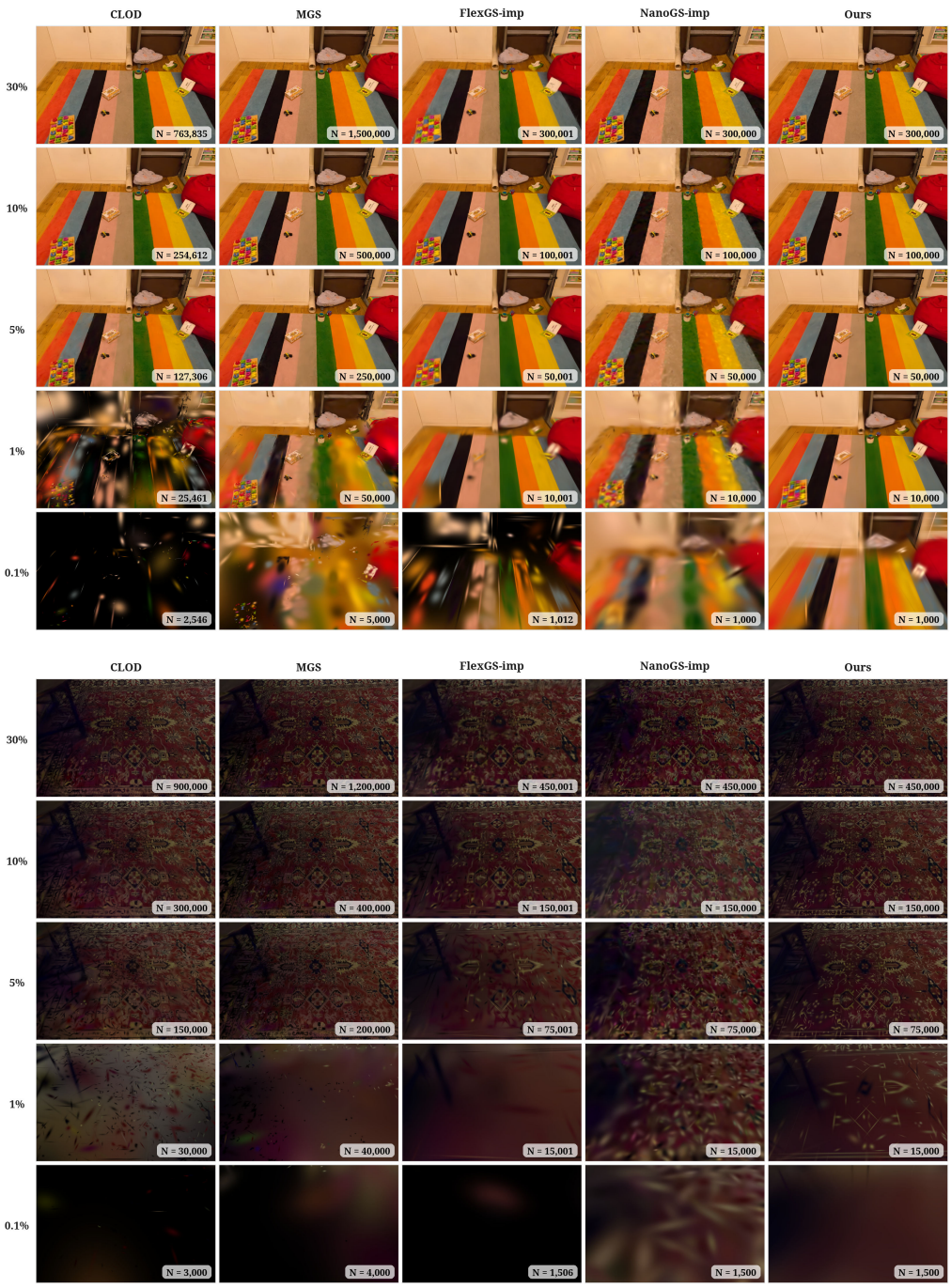


Figure 10: Cross-budget reconstruction on Playroom (top) and DrJohnson (bottom).

E CONTINUOUS-BUDGET QUALITY CURVES ON OTHER SCENES

Figures 11–13 show anchors, actual interval-decoding results, and independent GNS references for the 12 scenes beyond Garden in Figure 3.

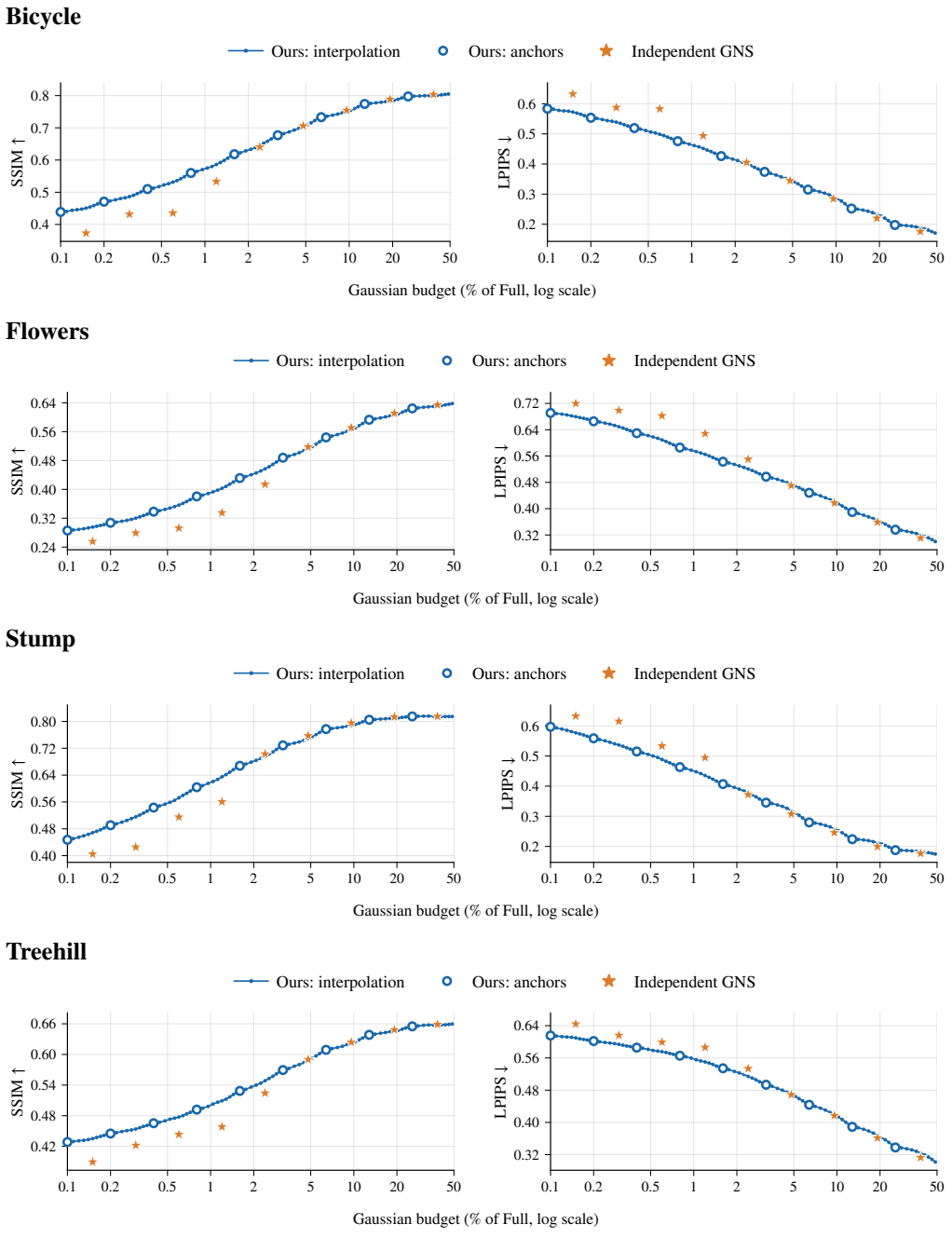


Figure 11: Continuous-budget quality curves for Bicycle, Flowers, Stump, and Treehill, top to bottom.

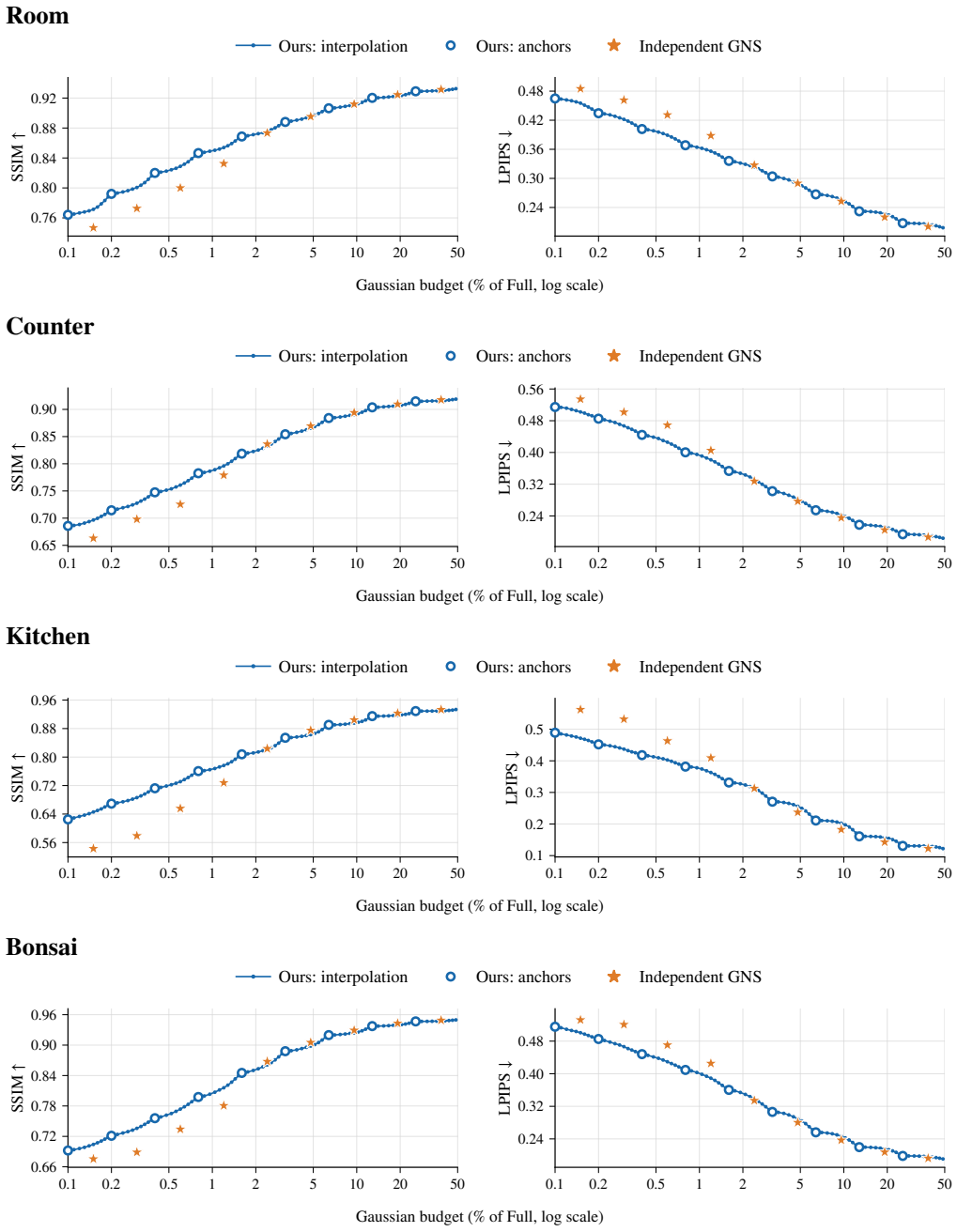


Figure 12: Continuous-budget quality curves for Room, Counter, Kitchen, and Bonsai, top to bottom.

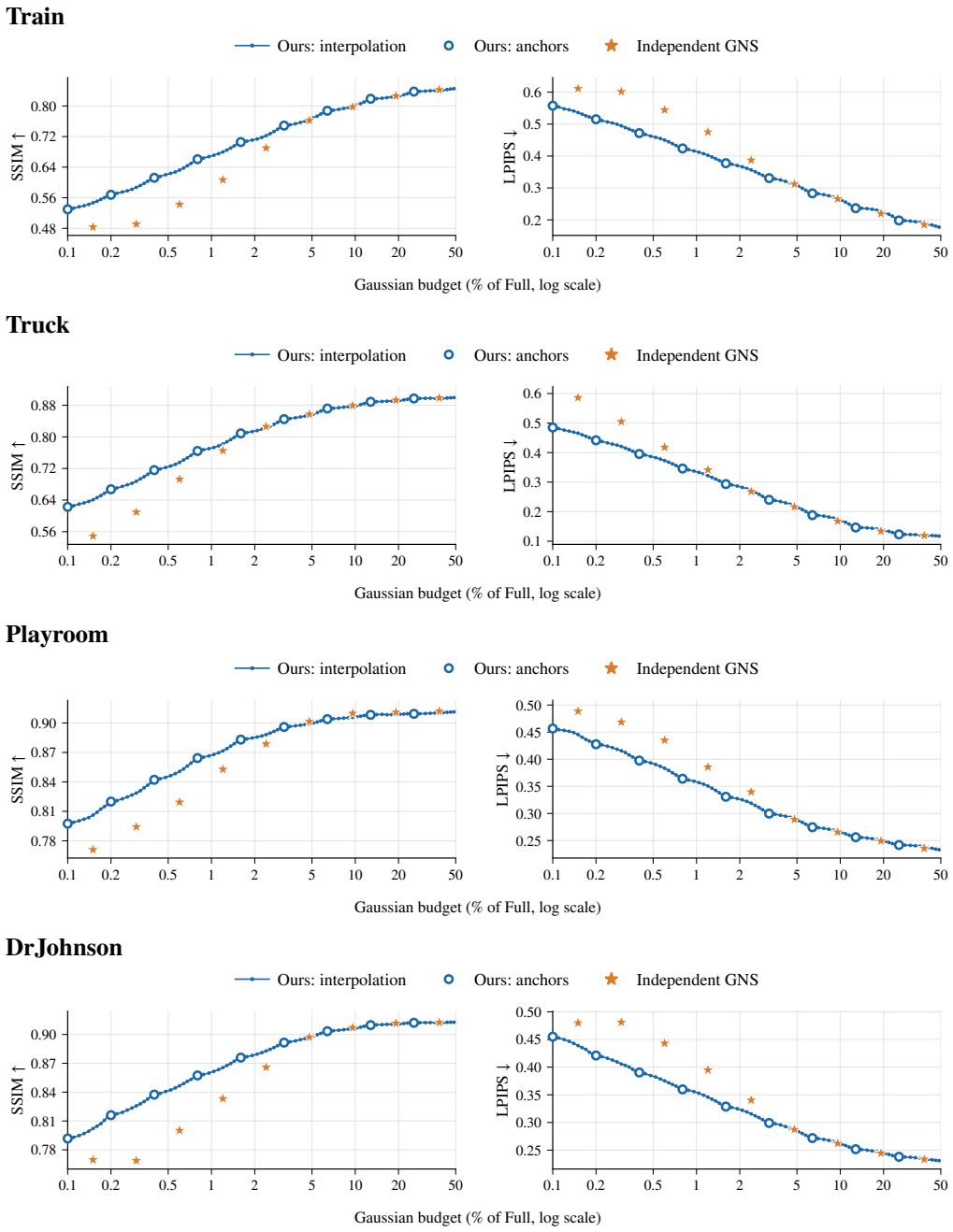


Figure 13: Continuous-budget quality curves for Train, Truck, Playroom, and DrJohnson, top to bottom.