

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

Xiaobin Deng Changyu Diao* Min Li Duanqing Xu*
Zhejiang University

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 390.0 to 85.7 MB with an average PSNR change of 0.01 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 bud-

*Corresponding authors.

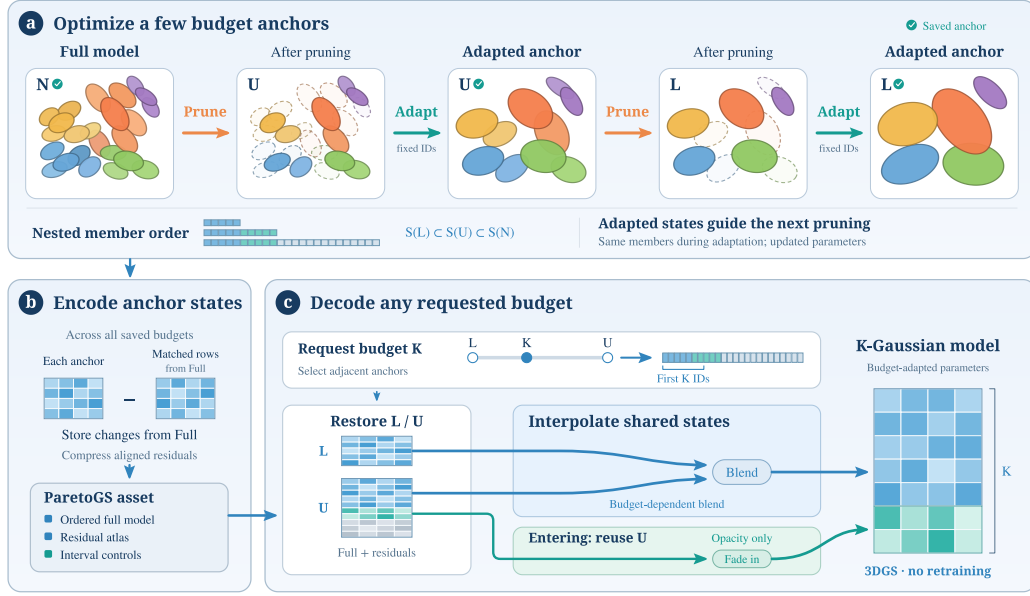


Figure 1: **ParetoGS overview.** (a) Alternating pruning and fixed-member adaptation builds sparse nested anchors; each adapted state seeds the next contraction. (b) Persistent identities align anchor states, whose parameter changes relative to the full model are encoded in a compact residual atlas. (c) For a requested count K between adjacent anchors L and U , decoding interpolates shared-member parameters and introduces $K - L$ entering members from U with opacity gating, producing a standard K -Gaussian model.

get anchors provide endpoints for the latter. Aligned states at adjacent anchors define a piecewise parameter trajectory for estimating shared-member parameters at intermediate counts.

We propose ParetoGS (Figure 1), 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 390.0 to 85.7 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 :

$$I(u) = \sum_i T_i(u) \tilde{a}_i(u) c_i(u) + T_{\text{bg}}(u) c_{\text{bg}}, \quad T_i(u) = \prod_{k \prec_u i} (1 - \tilde{a}_k(u)). \quad (1)$$

Here $\tilde{a}_i(u)$ 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:

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

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 $K_{\min} = K_1 < K_2 < \dots < K_M = N$. 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 $\pi_{L \rightarrow U}$ as budget grows. For $L < K < U$, decoding retains every member of the lower anchor and admits the first $K - L$ entries:

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

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 $K_j = \text{round}(N\rho_j)$ for $\rho_j = 0.001, 0.002, \dots, 0.512, 1$. 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. The Full model uses ImprovedGS (Deng et al., 2026) densification; subsequent anchor construction follows GNS parameter optimization and pruning with densification disabled.

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 $t \in (0, 1)$,

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

$\theta_i(L)$ 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; the shared interval gate t^q controls their rendering contribution. With $q > 0$ and upper-anchor opacity $a_i(U) = \sigma(o_i(U))$,

$$a_i(K) = \text{clip}(t^q a_i(U), \epsilon, 1 - \epsilon), \quad L < i \leq K. \quad (5)$$

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:

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

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 $\mathcal{S}(K_0) = \emptyset$. 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:

$$D_s^f = C_s^f B_s^f. \quad (7)$$

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 $\hat{\Delta}_j$ restored from the atlas is added to the corresponding reference rows: $\hat{\Theta}_{K_j} = \Theta_N[1 : K_j] + \hat{\Delta}_j$. 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

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 $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Num	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Num	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	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

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 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

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



Figure 2: 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.

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 achieves a mean PSNR of 24.40 dB, compared with 18.84 dB for NanoGS-imp. 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 2 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 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

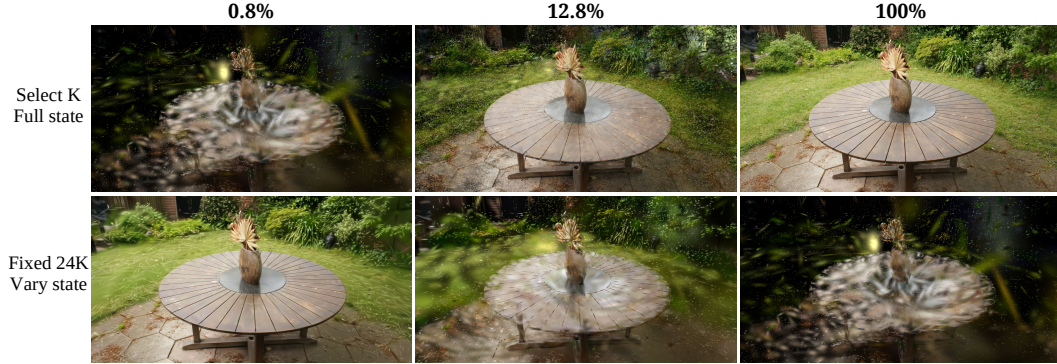


Figure 3: Parameter states of the same members across budgets. The Garden columns correspond to 0.8%, 12.8%, and 100% of Full. The first row selects 24K, 384K, and 3M members, respectively, while retaining Full parameters. The second row fixes the same 24K members from the upper-left panel; column headings 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 3 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.

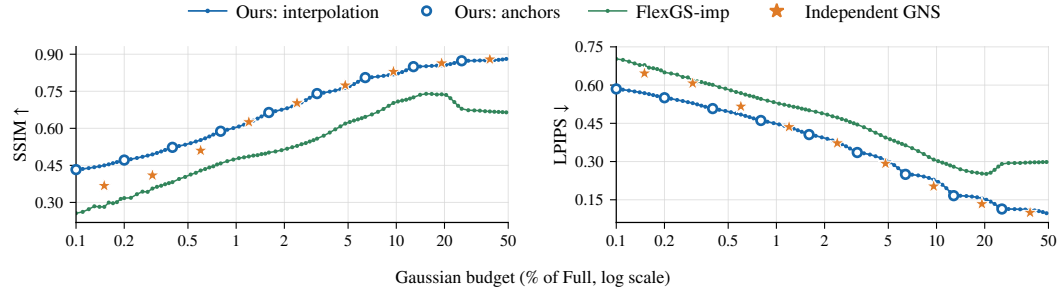


Figure 4: Continuous-budget quality on Garden. The full model has 3M Gaussians. SSIM and LPIPS are averaged over 24 test views against retention percentage on a log axis. Blue open circles are ParetoGS anchors and small blue dots are decoded intervals; green denotes FlexGS-imp, and orange stars are independently optimized GNS references. Line segments connect measured points over 0.1%–50%.

Quality between sparse anchors. Figure 4 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 $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Full method	25.47	0.7498	0.3458
Fixed shared state	25.35	0.7480	0.3477
No opacity gate	25.26	0.7478	0.3456
Linear progress	25.33	0.7459	0.3491
Full parameters	17.07	0.5459	0.4718

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

Encoding	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	MB $\downarrow$
FP32	25.49	0.7503	0.3452	390.0
FP16	25.49	0.7503	0.3452	195.0
Atlas	25.47	0.7498	0.3458	85.7

Fixing common members at the lower-anchor state reduces mean PSNR by 0.13 dB and SSIM by 0.0018, while increasing LPIPS by 0.0019. 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 25.47 to 17.07 dB, SSIM from 0.7498 to 0.5459, and raises LPIPS from 0.3458 to 0.4718. This quantifies the budget-specific state change illustrated in Figure 3; all 13 scenes show the same decline (Appendix Table 15).

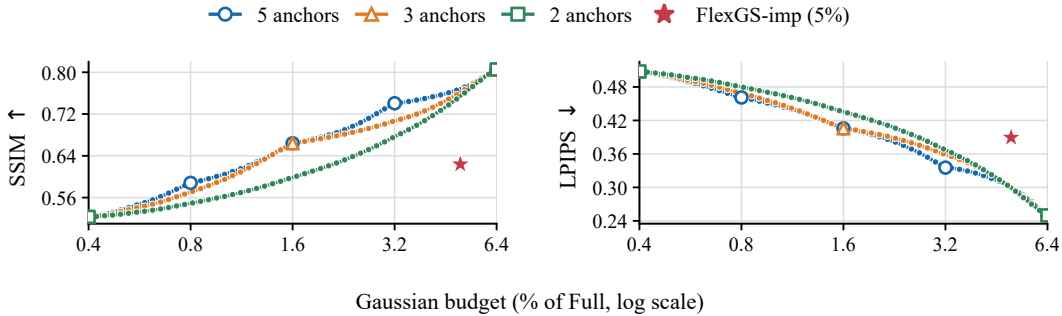


Figure 5: Effect of intermediate anchors on connection quality. From the same Garden state library and common 0.4% and 6.4% endpoints, we retain five, three, or two anchors. Open circles, triangles, and squares mark their anchors; small solid points show real evaluations at 50 shared non-anchor budgets. 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.

The decoded quality curves in Figure 5 remain smooth when anchor spacing doubles or quadruples, with mean SSIM changing from 0.6569 to 0.6467 and 0.6180. 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 0.6239, LPIPS 0.3894). Intermediate anchors raise the connecting curve’s average quality.

4.5 STATE ENCODING AND STORAGE

Across 13 scenes, the atlas reduces mean additional state from 390.0 MB (FP32) to 85.7 MB (atlas), or 78.0%, with only a 0.01 dB PSNR drop, 0.0005 SSIM drop, and 0.0005 LPIPS increase (Table 4). FP16 residuals use 195.0 MB with the same quality at shown precision. For Bicycle, Garden, and Stump, the atlas averages 158.7 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 2.77 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 member sets and persistent 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 390.0 to 85.7 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 Moënné-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/revs/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:

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

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

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

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 $h = 0, \dots, 4$. 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:

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

The image loss is $\ell = 0.8L_1 + 0.2(1 - \text{SSIM})$. 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	25.54	25.17	23.83	22.55	16.33	10.20
Bicycle	MGS	25.50	24.73	22.35	20.88	18.76	14.28
Bicycle	ParetoGS	25.87	25.81	25.18	24.63	23.36	21.62
Flowers	CLOD-3DGS	21.87	21.47	20.16	18.96	12.23	7.73
Flowers	MGS	21.89	21.11	18.73	17.65	15.40	11.73
Flowers	ParetoGS	21.79	21.84	21.23	20.66	19.25	17.72
Garden	CLOD-3DGS	27.93	27.54	25.83	23.93	15.28	8.16
Garden	MGS	27.95	27.30	24.85	22.68	18.48	13.99
Garden	ParetoGS	28.03	27.93	26.65	25.70	23.64	21.52
Stump	CLOD-3DGS	27.16	26.95	25.77	24.37	16.65	12.08
Stump	MGS	27.12	26.62	24.45	22.33	20.05	16.77
Stump	ParetoGS	27.24	27.31	26.90	26.34	24.47	21.98
Treehill	CLOD-3DGS	23.04	22.88	22.31	21.51	12.87	8.65
Treehill	MGS	22.88	22.14	20.29	19.54	17.44	12.69
Treehill	ParetoGS	22.81	23.16	22.97	22.75	22.02	20.80
Room	CLOD-3DGS	32.36	32.03	30.22	24.20	11.96	8.87
Room	MGS	32.05	31.80	30.72	29.56	24.19	16.05
Room	ParetoGS	32.60	32.55	31.84	31.13	28.61	23.36
Counter	CLOD-3DGS	29.47	29.09	26.95	20.86	11.98	8.91
Counter	MGS	29.84	29.49	28.21	26.88	21.95	15.69
Counter	ParetoGS	29.67	29.66	28.86	28.03	25.53	21.37
Kitchen	CLOD-3DGS	32.06	31.57	29.13	23.95	11.91	6.20
Kitchen	MGS	32.33	32.00	30.46	28.70	22.80	16.11
Kitchen	ParetoGS	32.52	32.15	30.28	28.68	25.17	21.17
Bonsai	CLOD-3DGS	32.60	32.13	28.42	20.46	12.09	9.54
Bonsai	MGS	32.98	32.55	30.70	28.65	23.34	17.09
Bonsai	ParetoGS	32.87	32.89	31.28	29.81	26.30	22.27
Train	CLOD-3DGS	22.33	21.77	19.21	14.80	8.52	5.66
Train	MGS	23.01	22.48	21.17	19.86	16.86	12.21
Train	ParetoGS	22.36	22.48	22.21	21.86	20.41	17.48
Truck	CLOD-3DGS	26.46	26.20	24.80	22.46	11.06	5.61
Truck	MGS	26.40	26.15	24.82	23.13	19.03	13.84
Truck	ParetoGS	26.44	26.52	25.98	25.32	23.12	19.47
Playroom	CLOD-3DGS	28.52	28.87	28.48	26.99	11.28	5.45
Playroom	MGS	29.66	29.66	29.49	28.92	23.09	14.96
Playroom	ParetoGS	30.68	30.32	30.26	30.00	27.69	22.41

Scene	Method	100%	30%	10%	5%	1%	0.1%
DrJohnson	CLOD-3DGS	28.61	28.52	27.61	25.93	15.07	8.89
DrJohnson	MGS	29.15	29.09	28.40	27.42	23.18	16.90
DrJohnson	ParetoGS	29.73	29.81	29.61	29.30	27.68	23.75

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	0.7926	0.7680	0.6797	0.5945	0.3539	0.0728
Bicycle	MGS	0.7939	0.7492	0.5655	0.4569	0.3460	0.2245
Bicycle	ParetoGS	0.8050	0.7988	0.7531	0.7072	0.5728	0.4388
Flowers	CLOD-3DGS	0.6381	0.6047	0.5146	0.4360	0.2184	0.0395
Flowers	MGS	0.6451	0.6022	0.4427	0.3645	0.2545	0.1596
Flowers	ParetoGS	0.6400	0.6272	0.5684	0.5165	0.3907	0.2859
Garden	CLOD-3DGS	0.8777	0.8629	0.7947	0.7062	0.3760	0.0964
Garden	MGS	0.8789	0.8562	0.7325	0.5887	0.3796	0.2574
Garden	ParetoGS	0.8822	0.8739	0.8207	0.7683	0.6039	0.4328
Stump	CLOD-3DGS	0.7991	0.7873	0.7250	0.6486	0.3370	0.0778
Stump	MGS	0.7996	0.7806	0.6609	0.5194	0.3804	0.2914
Stump	ParetoGS	0.8118	0.8158	0.7900	0.7532	0.6184	0.4473
Treehill	CLOD-3DGS	0.6530	0.6311	0.5720	0.5187	0.2792	0.0628
Treehill	MGS	0.6522	0.5995	0.4349	0.4035	0.3580	0.2586
Treehill	ParetoGS	0.6562	0.6567	0.6238	0.5909	0.5009	0.4284
Room	CLOD-3DGS	0.9301	0.9226	0.8938	0.8111	0.4076	0.1012
Room	MGS	0.9328	0.9269	0.9026	0.8771	0.7796	0.5783
Room	ParetoGS	0.9333	0.9292	0.9120	0.8965	0.8497	0.7641
Counter	CLOD-3DGS	0.9165	0.9047	0.8526	0.7197	0.3606	0.0732
Counter	MGS	0.9230	0.9151	0.8841	0.8465	0.7054	0.5473
Counter	ParetoGS	0.9191	0.9152	0.8916	0.8673	0.7876	0.6857
Kitchen	CLOD-3DGS	0.9333	0.9245	0.8869	0.7881	0.3920	0.0555
Kitchen	MGS	0.9362	0.9300	0.8975	0.8621	0.6645	0.4471
Kitchen	ParetoGS	0.9366	0.9290	0.8967	0.8649	0.7662	0.6251
Bonsai	CLOD-3DGS	0.9473	0.9402	0.8896	0.7221	0.3717	0.0965
Bonsai	MGS	0.9521	0.9471	0.9205	0.8798	0.7318	0.5622
Bonsai	ParetoGS	0.9497	0.9468	0.9252	0.8999	0.8051	0.6921
Train	CLOD-3DGS	0.8302	0.8007	0.6961	0.5333	0.2508	0.0381
Train	MGS	0.8579	0.8390	0.7643	0.6902	0.5419	0.3913
Train	ParetoGS	0.8442	0.8387	0.8010	0.7670	0.6695	0.5297
Truck	CLOD-3DGS	0.9008	0.8913	0.8524	0.7876	0.4219	0.1086
Truck	MGS	0.9022	0.8898	0.8389	0.7765	0.6173	0.4579
Truck	ParetoGS	0.8991	0.8975	0.8789	0.8568	0.7716	0.6228
Playroom	CLOD-3DGS	0.8964	0.9014	0.8923	0.8724	0.5805	0.0945
Playroom	MGS	0.9110	0.9110	0.9043	0.8933	0.8070	0.6731
Playroom	ParetoGS	0.9152	0.9095	0.9060	0.8996	0.8672	0.7975
DrJohnson	CLOD-3DGS	0.9024	0.8982	0.8783	0.8477	0.6235	0.0828
DrJohnson	MGS	0.9060	0.9044	0.8866	0.8662	0.7814	0.6652
DrJohnson	ParetoGS	0.9122	0.9123	0.9065	0.8978	0.8608	0.7918

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	0.1711	0.2078	0.3100	0.3859	0.5644	0.7261
Bicycle	MGS	0.1670	0.2256	0.4185	0.5281	0.6185	0.6888
Bicycle	ParetoGS	0.1613	0.1960	0.2872	0.3426	0.4634	0.5837
Flowers	CLOD-3DGS	0.2877	0.3294	0.4124	0.4763	0.6536	0.8190
Flowers	MGS	0.2659	0.3091	0.4390	0.5137	0.6453	0.7578
Flowers	ParetoGS	0.2813	0.3322	0.4176	0.4712	0.5756	0.6911
Garden	CLOD-3DGS	0.0988	0.1229	0.2223	0.3191	0.5634	0.7621
Garden	MGS	0.0962	0.1308	0.2881	0.4218	0.5873	0.7057
Garden	ParetoGS	0.0902	0.1135	0.2235	0.3008	0.4483	0.5849
Stump	CLOD-3DGS	0.1838	0.2057	0.2869	0.3638	0.5539	0.6942
Stump	MGS	0.1777	0.2068	0.3408	0.4709	0.6083	0.6700
Stump	ParetoGS	0.1725	0.1865	0.2549	0.3137	0.4502	0.5970
Treehill	CLOD-3DGS	0.2773	0.3219	0.4136	0.4786	0.6283	0.7154
Treehill	MGS	0.2631	0.3352	0.5323	0.5775	0.6294	0.6916
Treehill	ParetoGS	0.2816	0.3353	0.4156	0.4666	0.5573	0.6156
Room	CLOD-3DGS	0.2003	0.2190	0.2710	0.3537	0.5718	0.6629
Room	MGS	0.1905	0.2030	0.2474	0.2885	0.4063	0.5654
Room	ParetoGS	0.1943	0.2074	0.2536	0.2863	0.3632	0.4648
Counter	CLOD-3DGS	0.1879	0.2101	0.2765	0.3862	0.5728	0.6629
Counter	MGS	0.1679	0.1807	0.2248	0.2735	0.4276	0.5731
Counter	ParetoGS	0.1804	0.1933	0.2402	0.2789	0.3932	0.5146
Kitchen	CLOD-3DGS	0.1234	0.1396	0.1959	0.2985	0.5886	0.7387
Kitchen	MGS	0.1149	0.1258	0.1811	0.2287	0.4284	0.6099
Kitchen	ParetoGS	0.1151	0.1307	0.1995	0.2523	0.3762	0.4890
Bonsai	CLOD-3DGS	0.1956	0.2121	0.2833	0.4094	0.5685	0.6122
Bonsai	MGS	0.1808	0.1905	0.2302	0.2830	0.4398	0.5628
Bonsai	ParetoGS	0.1878	0.1982	0.2441	0.2862	0.4008	0.5154
Train	CLOD-3DGS	0.1994	0.2410	0.3497	0.4719	0.6409	0.7028
Train	MGS	0.1456	0.1709	0.2631	0.3466	0.4927	0.6255
Train	ParetoGS	0.1711	0.1966	0.2634	0.3074	0.4139	0.5575
Truck	CLOD-3DGS	0.1110	0.1352	0.1985	0.2726	0.5494	0.7092
Truck	MGS	0.0956	0.1269	0.2221	0.3081	0.4735	0.6135
Truck	ParetoGS	0.1179	0.1227	0.1712	0.2161	0.3352	0.4850
Playroom	CLOD-3DGS	0.2719	0.2714	0.2971	0.3275	0.5848	0.8378
Playroom	MGS	0.2333	0.2333	0.2548	0.2813	0.4072	0.5611
Playroom	ParetoGS	0.2289	0.2412	0.2658	0.2881	0.3582	0.4566
DrJohnson	CLOD-3DGS	0.2539	0.2673	0.3065	0.3489	0.5100	0.6082
DrJohnson	MGS	0.2374	0.2413	0.2744	0.3099	0.4207	0.5331
DrJohnson	ParetoGS	0.2280	0.2373	0.2617	0.2844	0.3543	0.4551

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

Scene	Method	30%	10%	5%	1%	0.1%
Bicycle	FlexGS-imp	20.62	23.21	22.67	21.68	15.97
Bicycle	NanoGS-imp	21.36	19.05	18.26	17.29	16.41
Bicycle	ParetoGS	25.81	25.18	24.63	23.36	21.62

Scene	Method	30%	10%	5%	1%	0.1%
Flowers	FlexGS-imp	16.99	19.45	18.87	18.00	14.41
Flowers	NanoGS-imp	18.16	16.74	16.26	15.43	14.02
Flowers	ParetoGS	21.84	21.23	20.66	19.25	17.72
Garden	FlexGS-imp	21.24	24.06	22.89	21.40	12.82
Garden	NanoGS-imp	22.51	20.12	19.15	18.06	16.06
Garden	ParetoGS	27.93	26.65	25.70	23.64	21.52
Stump	FlexGS-imp	17.64	23.55	22.38	21.07	12.17
Stump	NanoGS-imp	23.15	20.59	19.89	19.03	17.71
Stump	ParetoGS	27.31	26.90	26.34	24.47	21.98
Treehill	FlexGS-imp	20.93	21.85	21.61	20.96	17.42
Treehill	NanoGS-imp	19.50	18.32	18.03	17.73	16.50
Treehill	ParetoGS	23.16	22.97	22.75	22.02	20.80
Room	FlexGS-imp	25.31	29.99	28.47	24.95	17.47
Room	NanoGS-imp	28.54	24.94	23.82	21.51	17.65
Room	ParetoGS	32.55	31.84	31.13	28.61	23.36
Counter	FlexGS-imp	25.03	27.33	25.94	22.72	15.76
Counter	NanoGS-imp	25.98	23.32	22.05	20.03	16.73
Counter	ParetoGS	29.66	28.86	28.03	25.53	21.37
Kitchen	FlexGS-imp	21.59	27.31	25.36	22.49	14.47
Kitchen	NanoGS-imp	26.37	23.20	21.83	19.49	16.65
Kitchen	ParetoGS	32.15	30.28	28.68	25.17	21.17
Bonsai	FlexGS-imp	26.81	29.35	27.37	23.25	18.00
Bonsai	NanoGS-imp	27.72	23.98	22.74	20.70	17.14
Bonsai	ParetoGS	32.89	31.28	29.81	26.30	22.27
Train	FlexGS-imp	19.55	21.21	20.27	18.11	10.24
Train	NanoGS-imp	18.81	16.48	15.33	13.89	12.59
Train	ParetoGS	22.48	22.21	21.86	20.41	17.48
Truck	FlexGS-imp	18.95	22.52	21.26	18.86	7.05
Truck	NanoGS-imp	22.51	19.85	18.64	16.87	14.60
Truck	ParetoGS	26.52	25.98	25.32	23.12	19.47
Playroom	FlexGS-imp	25.45	29.28	27.80	24.32	8.96
Playroom	NanoGS-imp	28.48	25.50	24.37	22.55	17.88
Playroom	ParetoGS	30.32	30.26	30.00	27.69	22.41
DrJohnson	FlexGS-imp	25.65	28.61	27.52	24.93	11.28
DrJohnson	NanoGS-imp	27.12	24.62	24.01	22.38	18.84
DrJohnson	ParetoGS	29.81	29.61	29.30	27.68	23.75

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

Scene	Method	30%	10%	5%	1%	0.1%
Bicycle	FlexGS-imp	0.6193	0.6643	0.5950	0.4700	0.3109
Bicycle	NanoGS-imp	0.5556	0.4102	0.3676	0.3247	0.2990
Bicycle	ParetoGS	0.7988	0.7531	0.7072	0.5728	0.4388
Flowers	FlexGS-imp	0.3761	0.4319	0.3909	0.3202	0.2314
Flowers	NanoGS-imp	0.4279	0.3258	0.2924	0.2484	0.2221
Flowers	ParetoGS	0.6272	0.5684	0.5165	0.3907	0.2859
Garden	FlexGS-imp	0.6726	0.7067	0.6239	0.4766	0.2519
Garden	NanoGS-imp	0.6025	0.4475	0.3925	0.3330	0.3014
Garden	ParetoGS	0.8739	0.8207	0.7683	0.6039	0.4328

Scene	Method	30%	10%	5%	1%	0.1%
Stump	FlexGS-imp	0.4418	0.6833	0.6000	0.4882	0.2000
Stump	NanoGS-imp	0.6073	0.4548	0.4109	0.3654	0.3380
Stump	ParetoGS	0.8158	0.7900	0.7532	0.6184	0.4473
Treehill	FlexGS-imp	0.5795	0.5775	0.5301	0.4514	0.3680
Treehill	NanoGS-imp	0.4555	0.3841	0.3719	0.3667	0.3511
Treehill	ParetoGS	0.6567	0.6238	0.5909	0.5009	0.4284
Room	FlexGS-imp	0.8308	0.8883	0.8592	0.7981	0.6636
Room	NanoGS-imp	0.8521	0.7752	0.7473	0.6954	0.6246
Room	ParetoGS	0.9292	0.9120	0.8965	0.8497	0.7641
Counter	FlexGS-imp	0.8305	0.8561	0.8156	0.7335	0.5946
Counter	NanoGS-imp	0.8173	0.7224	0.6824	0.6355	0.5755
Counter	ParetoGS	0.9152	0.8916	0.8673	0.7876	0.6857
Kitchen	FlexGS-imp	0.7763	0.8607	0.8133	0.6954	0.4645
Kitchen	NanoGS-imp	0.8182	0.6871	0.6168	0.4960	0.4665
Kitchen	ParetoGS	0.9290	0.8967	0.8649	0.7662	0.6251
Bonsai	FlexGS-imp	0.8796	0.8926	0.8406	0.7313	0.6454
Bonsai	NanoGS-imp	0.8647	0.7573	0.7124	0.6435	0.5692
Bonsai	ParetoGS	0.9468	0.9252	0.8999	0.8051	0.6921
Train	FlexGS-imp	0.7286	0.7465	0.6964	0.5945	0.3784
Train	NanoGS-imp	0.6944	0.5696	0.5204	0.4638	0.4275
Train	ParetoGS	0.8387	0.8010	0.7670	0.6695	0.5297
Truck	FlexGS-imp	0.7413	0.7947	0.7492	0.6590	0.2311
Truck	NanoGS-imp	0.7666	0.6462	0.5954	0.5250	0.4857
Truck	ParetoGS	0.8975	0.8789	0.8568	0.7716	0.6228
Playroom	FlexGS-imp	0.8499	0.8975	0.8778	0.8308	0.4138
Playroom	NanoGS-imp	0.8786	0.8374	0.8167	0.7867	0.7323
Playroom	ParetoGS	0.9095	0.9060	0.8996	0.8672	0.7975
DrJohnson	FlexGS-imp	0.8403	0.8836	0.8618	0.8182	0.3451
DrJohnson	NanoGS-imp	0.8679	0.8162	0.7991	0.7632	0.7135
DrJohnson	ParetoGS	0.9123	0.9065	0.8978	0.8608	0.7918

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

Scene	Method	30%	10%	5%	1%	0.1%
Bicycle	FlexGS-imp	0.3459	0.3494	0.4160	0.5423	0.6872
Bicycle	NanoGS-imp	0.3603	0.4801	0.5349	0.6497	0.7321
Bicycle	ParetoGS	0.1960	0.2872	0.3426	0.4634	0.5837
Flowers	FlexGS-imp	0.5558	0.5246	0.5619	0.6399	0.7682
Flowers	NanoGS-imp	0.4562	0.5526	0.5984	0.7128	0.7745
Flowers	ParetoGS	0.3322	0.4176	0.4712	0.5756	0.6911
Garden	FlexGS-imp	0.2943	0.3038	0.3894	0.5302	0.7072
Garden	NanoGS-imp	0.3117	0.4373	0.4970	0.6157	0.7597
Garden	ParetoGS	0.1135	0.2235	0.3008	0.4483	0.5849
Stump	FlexGS-imp	0.5328	0.3311	0.4151	0.5242	0.6659
Stump	NanoGS-imp	0.3406	0.4700	0.5249	0.6337	0.7089
Stump	ParetoGS	0.1865	0.2549	0.3137	0.4502	0.5970
Treehill	FlexGS-imp	0.4079	0.4555	0.5175	0.5937	0.6653
Treehill	NanoGS-imp	0.4885	0.5723	0.6106	0.6815	0.6710
Treehill	ParetoGS	0.3353	0.4156	0.4666	0.5573	0.6156

Scene	Method	30%	10%	5%	1%	0.1%
Room	FlexGS-imp	0.3311	0.2920	0.3429	0.4254	0.5183
Room	NanoGS-imp	0.2986	0.3825	0.4223	0.5020	0.5375
Room	ParetoGS	0.2074	0.2536	0.2863	0.3632	0.4648
Counter	FlexGS-imp	0.2919	0.2871	0.3450	0.4448	0.5561
Counter	NanoGS-imp	0.2896	0.3913	0.4388	0.5256	0.5730
Counter	ParetoGS	0.1933	0.2402	0.2789	0.3932	0.5146
Kitchen	FlexGS-imp	0.2961	0.2499	0.3097	0.4332	0.6270
Kitchen	NanoGS-imp	0.2431	0.3752	0.4421	0.5730	0.6484
Kitchen	ParetoGS	0.1307	0.1995	0.2523	0.3762	0.4890
Bonsai	FlexGS-imp	0.2757	0.2876	0.3487	0.4632	0.5358
Bonsai	NanoGS-imp	0.2864	0.3970	0.4454	0.5261	0.5716
Bonsai	ParetoGS	0.1982	0.2441	0.2862	0.4008	0.5154
Train	FlexGS-imp	0.2961	0.3084	0.3657	0.4804	0.6470
Train	NanoGS-imp	0.3228	0.4530	0.5119	0.5987	0.6335
Train	ParetoGS	0.1966	0.2634	0.3074	0.4139	0.5575
Truck	FlexGS-imp	0.2898	0.2660	0.3291	0.4435	0.6905
Truck	NanoGS-imp	0.2659	0.3960	0.4596	0.5691	0.6572
Truck	ParetoGS	0.1227	0.1712	0.2161	0.3352	0.4850
Playroom	FlexGS-imp	0.3436	0.2968	0.3355	0.4114	0.6926
Playroom	NanoGS-imp	0.2975	0.3664	0.3989	0.4557	0.4982
Playroom	ParetoGS	0.2412	0.2658	0.2881	0.3582	0.4566
DrJohnson	FlexGS-imp	0.3518	0.3017	0.3390	0.4095	0.5738
DrJohnson	NanoGS-imp	0.3145	0.4002	0.4353	0.4872	0.5196
DrJohnson	ParetoGS	0.2373	0.2617	0.2844	0.3543	0.4551

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

Scene	Method	100%	30%	10%	5%	1%	0.1%
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
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

Scene	Count	30%	10%	5%	1%	0.1%
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 $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Bicycle	Full method	24.16	0.6544	0.3687
Bicycle	Fixed shared state	24.04	0.6523	0.3701
Bicycle	No opacity gate	24.03	0.6522	0.3686
Bicycle	Linear progress	24.08	0.6494	0.3732
Bicycle	Full parameters	15.57	0.3871	0.4946
Flowers	Full method	20.12	0.4789	0.4894
Flowers	Fixed shared state	20.07	0.4766	0.4915
Flowers	No opacity gate	20.03	0.4779	0.4889
Flowers	Linear progress	20.05	0.4732	0.4943
Flowers	Full parameters	13.84	0.3186	0.5860
Garden	Full method	25.13	0.7012	0.3286
Garden	Fixed shared state	24.94	0.6988	0.3300
Garden	No opacity gate	24.81	0.6971	0.3294
Garden	Linear progress	24.96	0.6948	0.3333
Garden	Full parameters	14.81	0.4177	0.4917
Stump	Full method	25.43	0.6865	0.3560
Stump	Fixed shared state	25.33	0.6841	0.3582
Stump	No opacity gate	25.26	0.6837	0.3562
Stump	Linear progress	25.30	0.6808	0.3606
Stump	Full parameters	17.62	0.4127	0.4713
Treehill	Full method	22.32	0.5597	0.4716
Treehill	Fixed shared state	22.31	0.5585	0.4731
Treehill	No opacity gate	22.25	0.5578	0.4718
Treehill	Linear progress	22.22	0.5558	0.4749
Treehill	Full parameters	15.14	0.3871	0.5585
Room	Full method	29.59	0.8723	0.3118
Room	Fixed shared state	29.40	0.8710	0.3136
Room	No opacity gate	29.36	0.8707	0.3116
Room	Linear progress	29.43	0.8700	0.3138
Room	Full parameters	20.30	0.7013	0.4290
Counter	Full method	26.75	0.8311	0.3217
Counter	Fixed shared state	26.60	0.8294	0.3241
Counter	No opacity gate	26.61	0.8297	0.3211
Counter	Linear progress	26.61	0.8280	0.3243
Counter	Full parameters	18.58	0.6393	0.4390
Kitchen	Full method	27.57	0.8202	0.2850
Kitchen	Fixed shared state	27.33	0.8177	0.2873
Kitchen	No opacity gate	26.75	0.8158	0.2856
Kitchen	Linear progress	27.21	0.8164	0.2865
Kitchen	Full parameters	15.57	0.5316	0.4713
Bonsai	Full method	28.59	0.8550	0.3261
Bonsai	Fixed shared state	28.41	0.8534	0.3277
Bonsai	No opacity gate	28.21	0.8531	0.3256
Bonsai	Linear progress	28.36	0.8515	0.3290
Bonsai	Full parameters	19.32	0.6524	0.4391

Scene	Variant	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Train	Full method	20.93	0.7245	0.3407
Train	Fixed shared state	20.87	0.7224	0.3429
Train	No opacity gate	20.82	0.7227	0.3402
Train	Linear progress	20.84	0.7207	0.3437
Train	Full parameters	14.66	0.5443	0.4630
Truck	Full method	24.12	0.8090	0.2624
Truck	Fixed shared state	23.99	0.8068	0.2652
Truck	No opacity gate	23.97	0.8075	0.2619
Truck	Linear progress	24.00	0.8050	0.2662
Truck	Full parameters	16.54	0.6110	0.4096
Playroom	Full method	28.29	0.8781	0.3191
Playroom	Fixed shared state	28.17	0.8773	0.3210
Playroom	No opacity gate	28.22	0.8777	0.3186
Playroom	Linear progress	28.15	0.8764	0.3214
Playroom	Full parameters	19.89	0.7582	0.4398
DrJohnson	Full method	28.16	0.8762	0.3141
DrJohnson	Fixed shared state	28.02	0.8751	0.3158
DrJohnson	No opacity gate	28.06	0.8754	0.3137
DrJohnson	Linear progress	28.07	0.8746	0.3166
DrJohnson	Full parameters	20.12	0.7351	0.4399

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 $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	MB $\downarrow$
Bicycle	FP32	24.18	0.6550	0.3678	724.3
Bicycle	FP16	24.18	0.6550	0.3678	362.2
Bicycle	Atlas	24.16	0.6544	0.3687	158.7
Flowers	FP32	20.13	0.4792	0.4890	362.2
Flowers	FP16	20.13	0.4792	0.4890	181.1
Flowers	Atlas	20.12	0.4789	0.4894	79.6
Garden	FP32	25.15	0.7016	0.3278	724.3
Garden	FP16	25.15	0.7016	0.3278	362.2
Garden	Atlas	25.13	0.7012	0.3286	158.7
Stump	FP32	25.44	0.6869	0.3551	724.3
Stump	FP16	25.44	0.6869	0.3551	362.2
Stump	Atlas	25.43	0.6865	0.3560	158.7
Treehill	FP32	22.32	0.5600	0.4711	362.2
Treehill	FP16	22.32	0.5600	0.4711	181.1
Treehill	Atlas	22.32	0.5597	0.4716	79.6
Room	FP32	29.60	0.8726	0.3116	241.4
Room	FP16	29.60	0.8726	0.3116	120.7
Room	Atlas	29.59	0.8723	0.3118	53.2
Counter	FP32	26.76	0.8315	0.3214	241.4
Counter	FP16	26.76	0.8315	0.3214	120.7
Counter	Atlas	26.75	0.8311	0.3217	53.2

Scene	Encoding	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	MB $\downarrow$
Kitchen	FP32	27.60	0.8220	0.2839	241.4
Kitchen	FP16	27.60	0.8220	0.2839	120.7
Kitchen	Atlas	27.57	0.8202	0.2850	53.2
Bonsai	FP32	28.60	0.8555	0.3257	241.4
Bonsai	FP16	28.60	0.8555	0.3257	120.7
Bonsai	Atlas	28.59	0.8550	0.3261	53.2
Train	FP32	20.94	0.7253	0.3403	241.4
Train	FP16	20.94	0.7253	0.3403	120.7
Train	Atlas	20.93	0.7245	0.3407	53.2
Truck	FP32	24.13	0.8096	0.2620	362.2
Truck	FP16	24.13	0.8096	0.2620	181.1
Truck	Atlas	24.12	0.8090	0.2624	79.6
Playroom	FP32	28.29	0.8785	0.3188	241.4
Playroom	FP16	28.29	0.8785	0.3188	120.7
Playroom	Atlas	28.29	0.8781	0.3191	53.2
DrJohnson	FP32	28.17	0.8765	0.3137	362.2
DrJohnson	FP16	28.17	0.8765	0.3137	181.1
DrJohnson	Atlas	28.16	0.8762	0.3141	79.6

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	0.12	3.01	7.46
MGS	0.41	7.79	8.10
FlexGS-imp	18.66	46.91	98.87
ParetoGS	0.27	2.77	5.65

ParetoGS has a median of 0.27 ms, P95 of 2.77 ms, and maximum of 5.65 ms in Table 17, showing low computational cost for endpoint interpolation and parameter construction. Its one-time pre-decoding of all anchors averages 0.39 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 2. Each panel labels its actual Gaussian count; two scenes appear on each page.

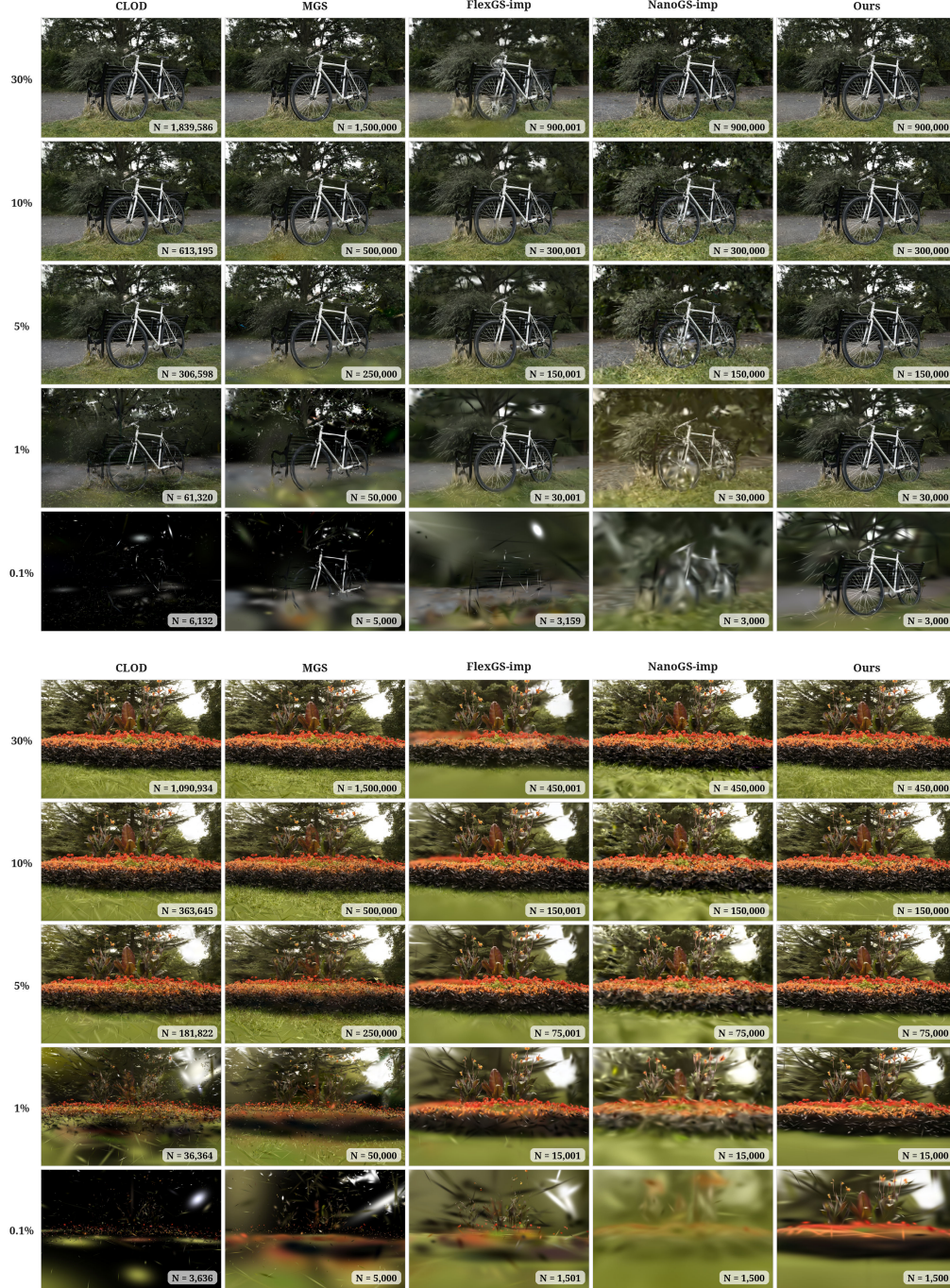


Figure 6: Cross-budget reconstruction on Bicycle (top) and Flowers (bottom).



Figure 7: Cross-budget reconstruction on Stump (top) and Treehill (bottom).

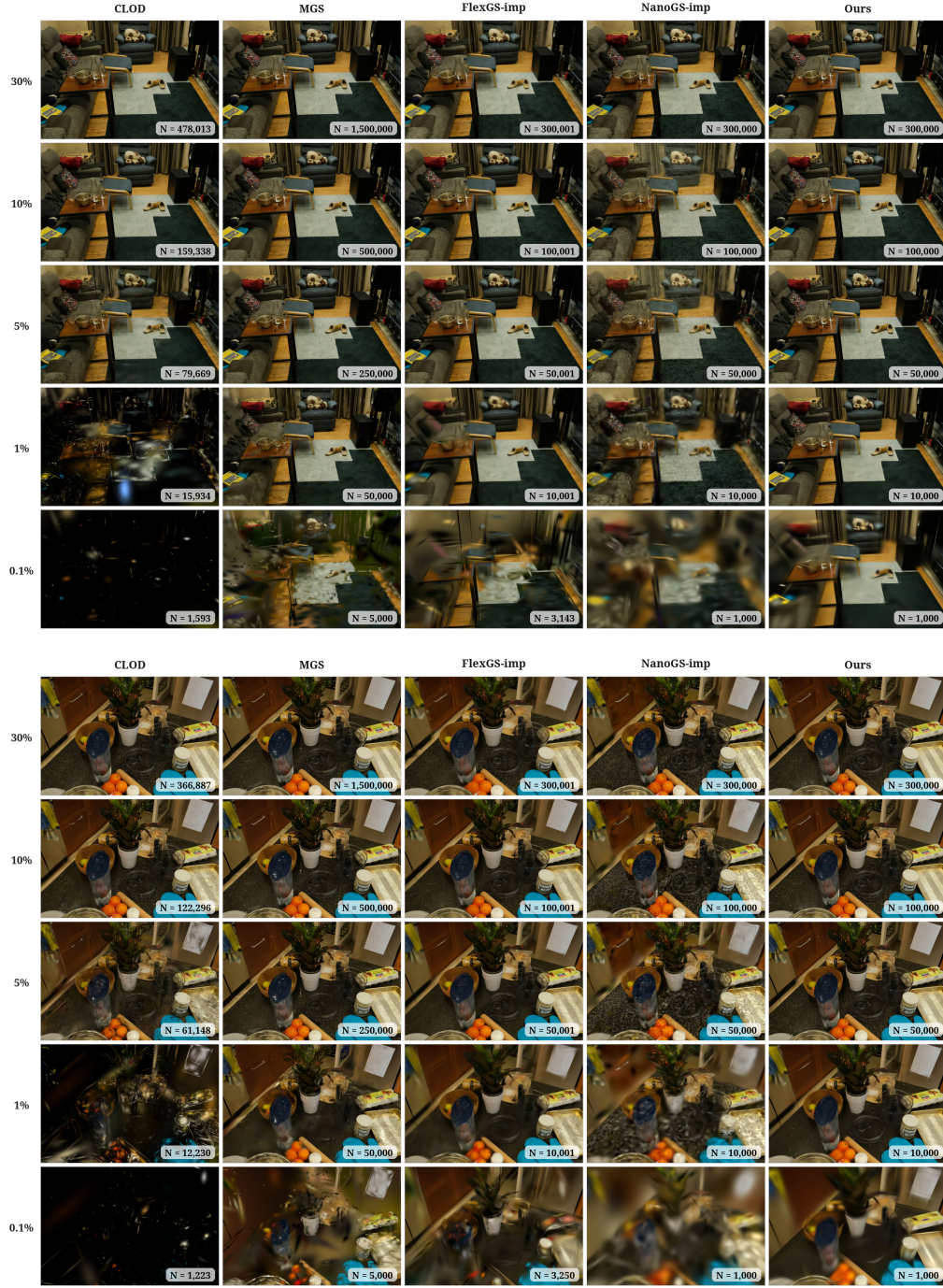


Figure 8: Cross-budget reconstruction on Room (top) and Counter (bottom).

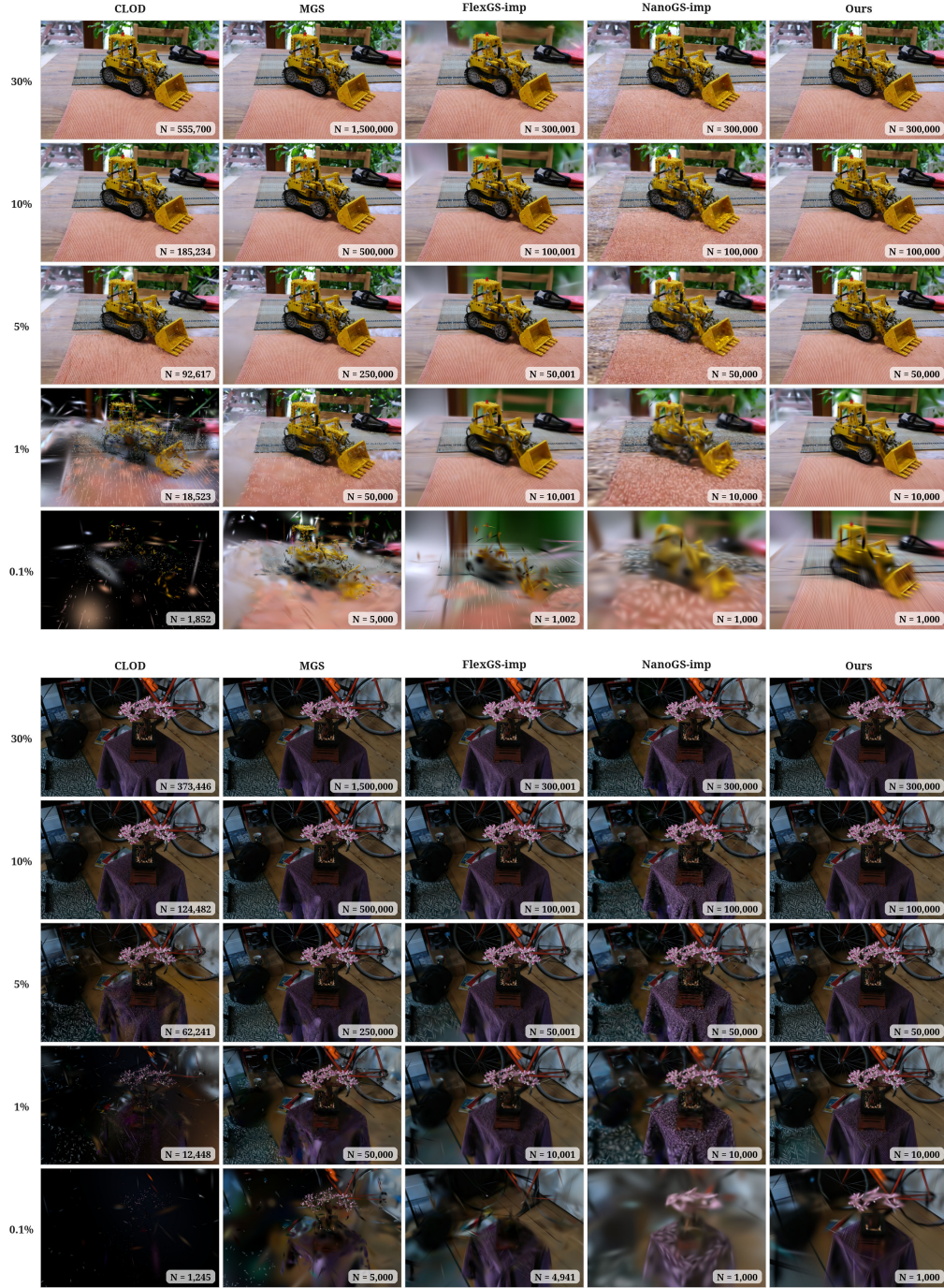


Figure 9: Cross-budget reconstruction on Kitchen (top) and Bonsai (bottom).

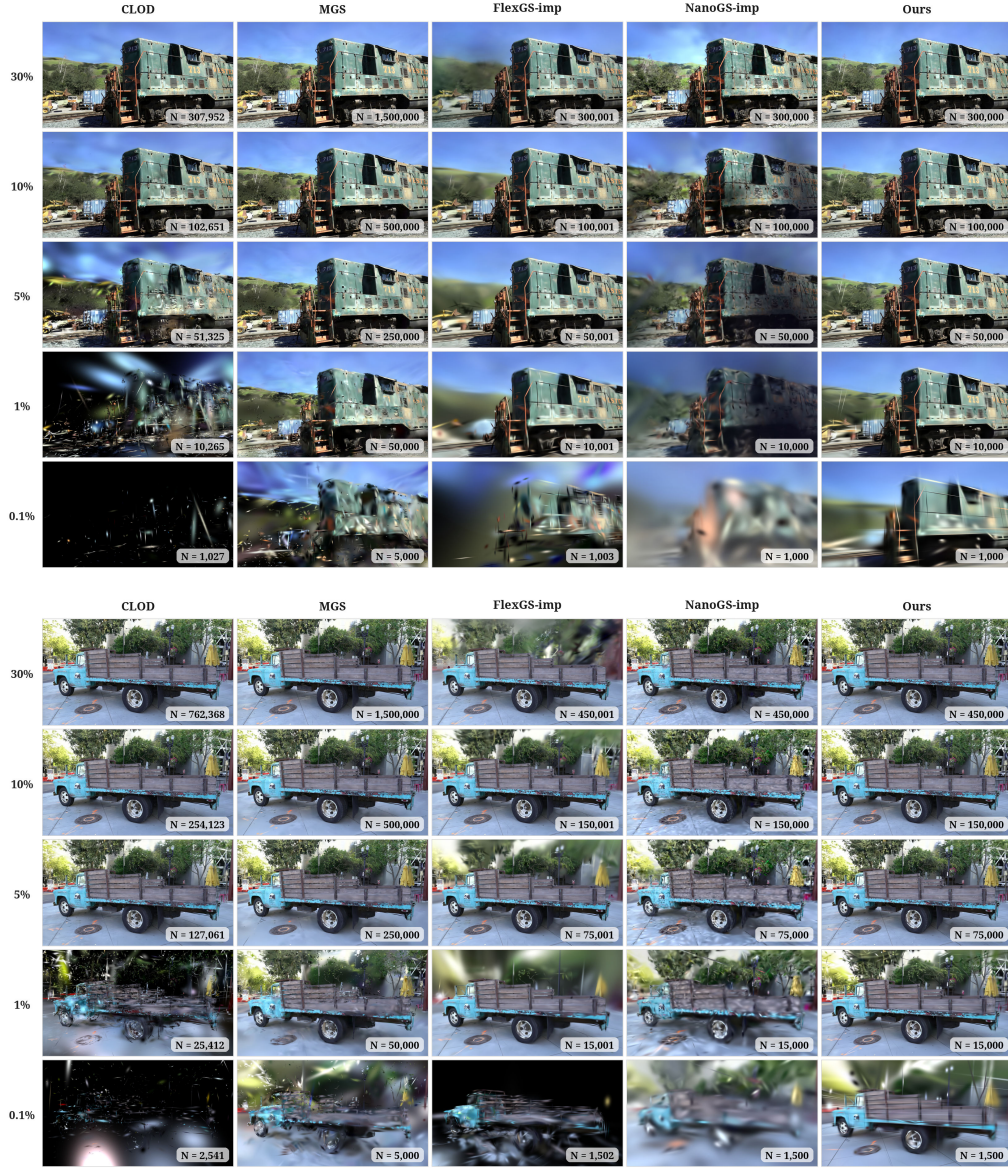


Figure 10: Cross-budget reconstruction on Train (top) and Truck (bottom).

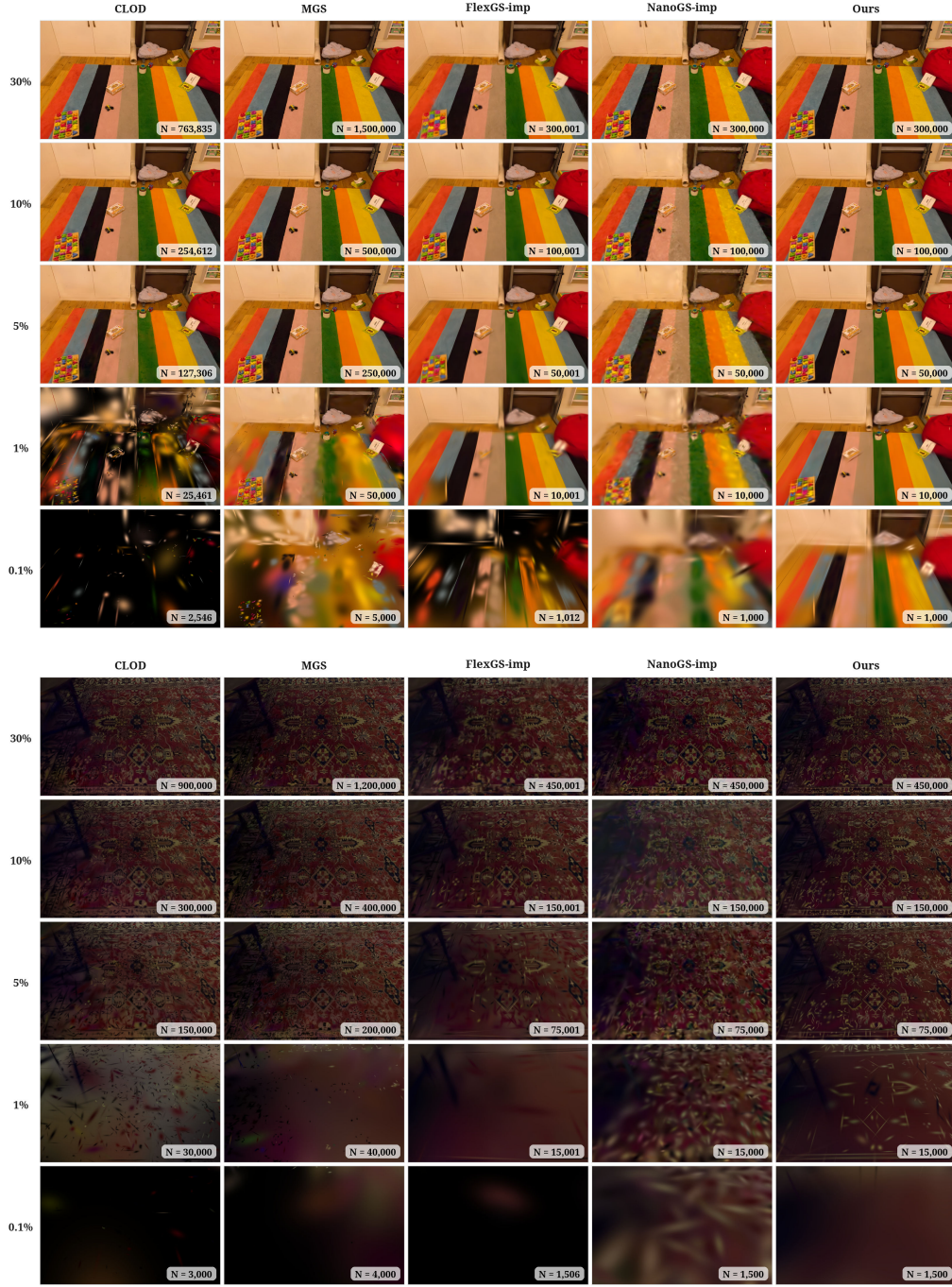
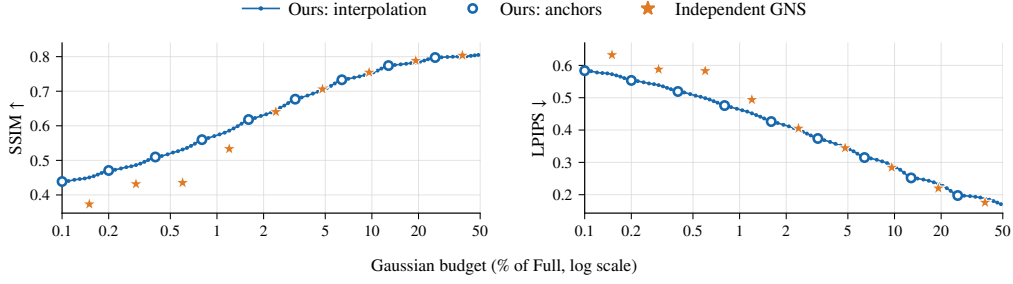


Figure 11: Cross-budget reconstruction on Playroom (top) and DrJohnson (bottom).

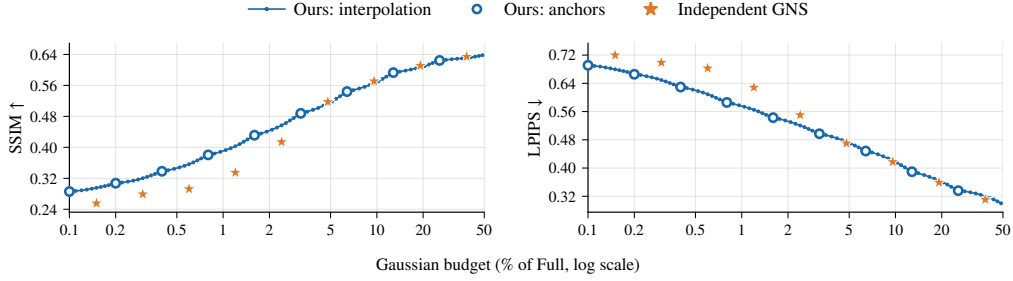
E CONTINUOUS-BUDGET QUALITY CURVES ON OTHER SCENES

Figures 12–14 show anchors, actual interval-decoding results, and independent GNS references for the 12 scenes beyond Garden in Figure 4.

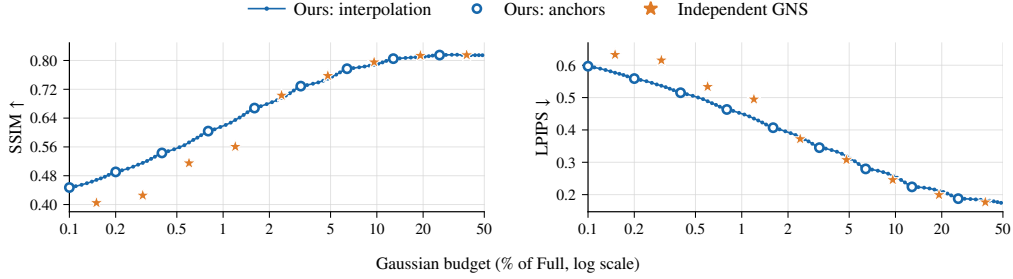
Bicycle



Flowers



Stump



Treehill

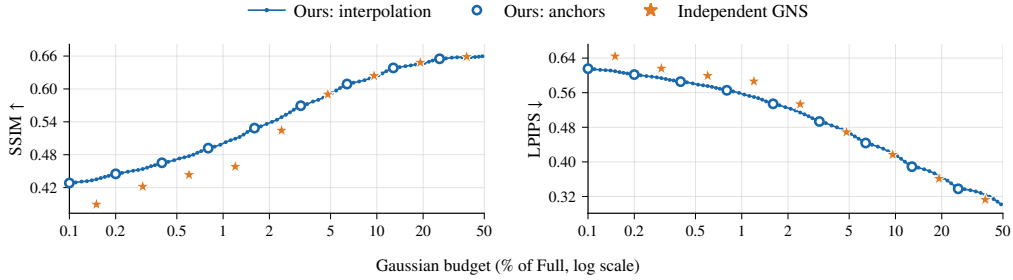
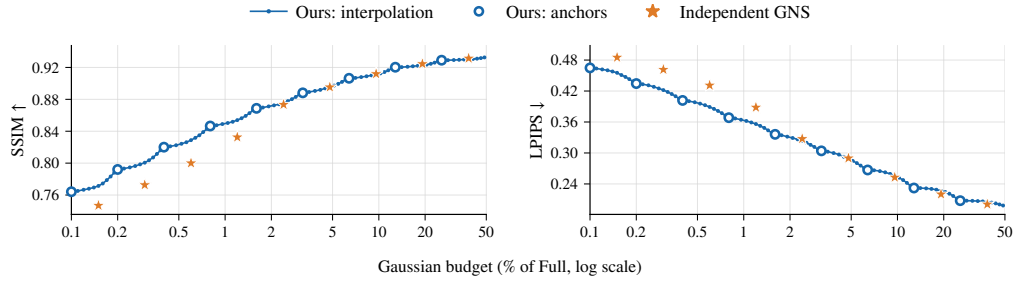
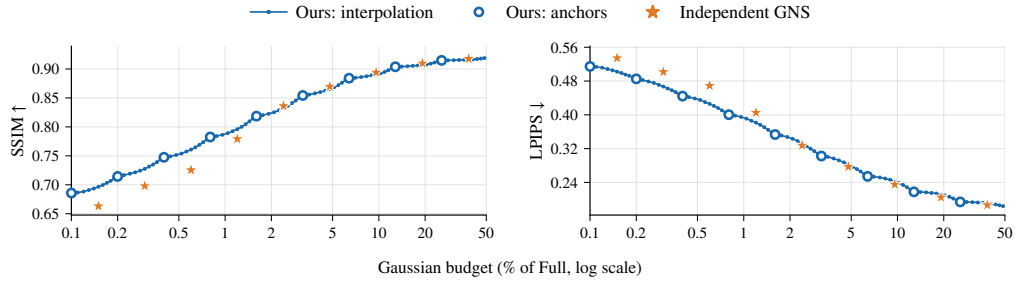


Figure 12: Continuous-budget quality curves for Bicycle, Flowers, Stump, and Treehill, top to bottom.

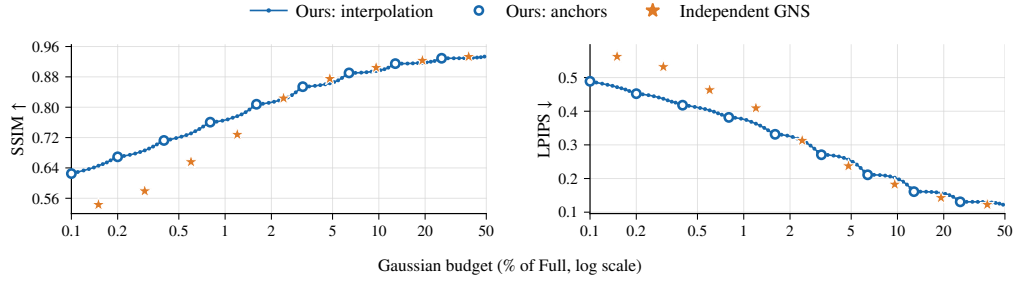
Room



Counter



Kitchen



Bonsai

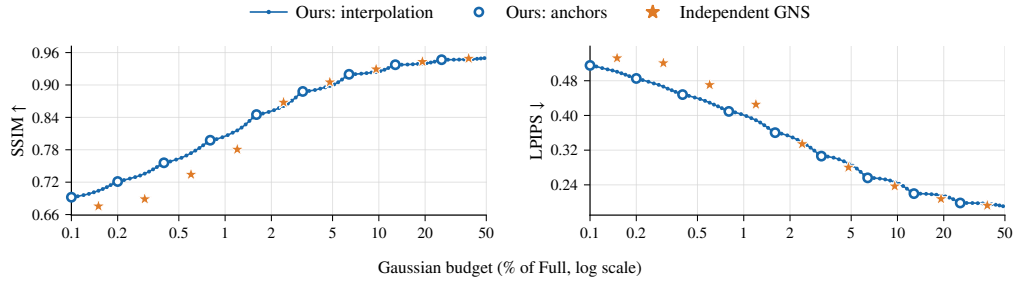
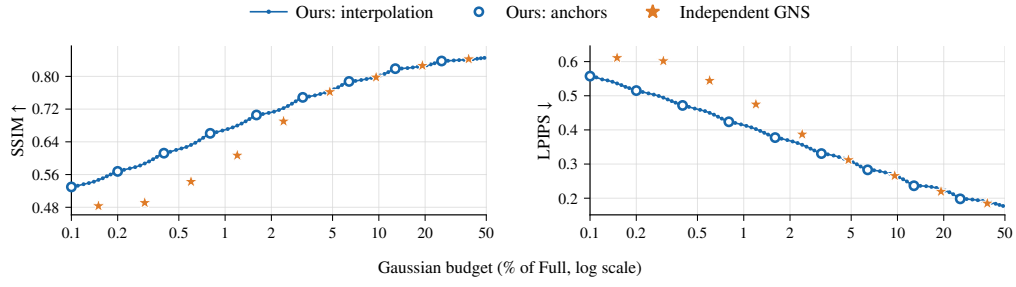
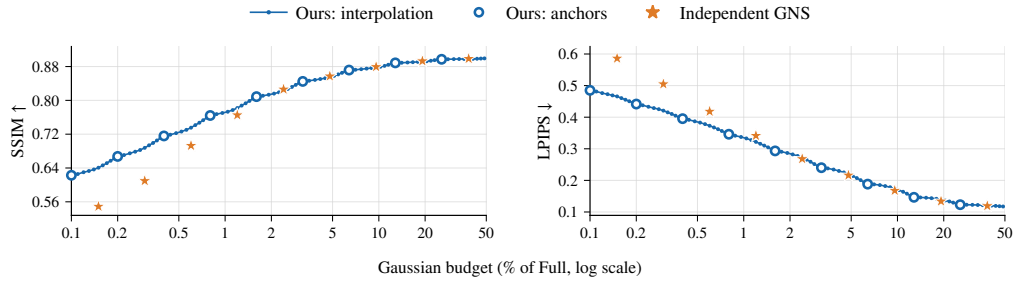


Figure 13: Continuous-budget quality curves for Room, Counter, Kitchen, and Bonsai, top to bottom.

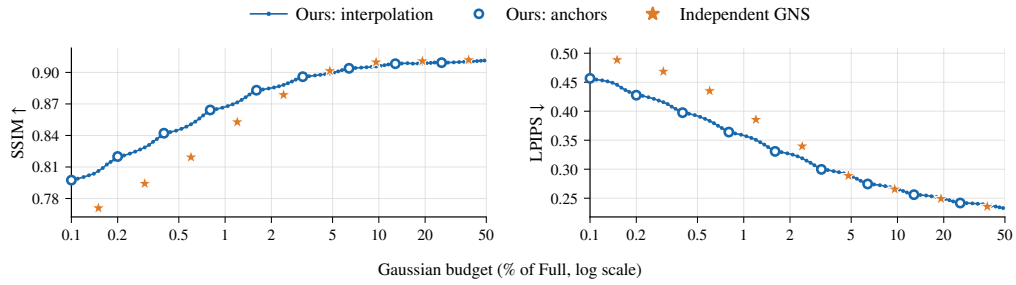
Train



Truck



Playroom



DrJohnson

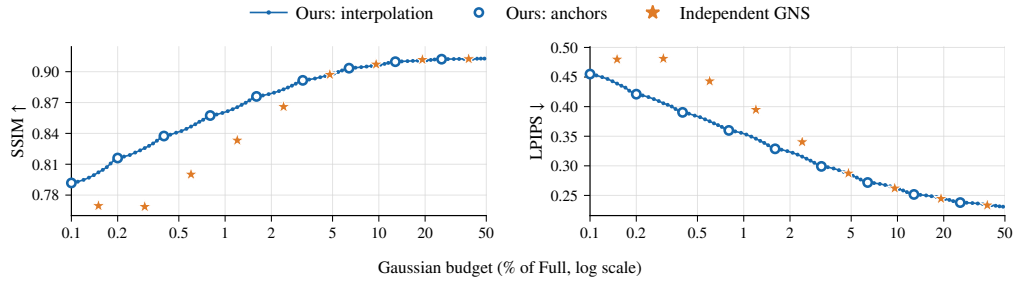


Figure 14: Continuous-budget quality curves for Train, Truck, Playroom, and DrJohnson, top to bottom.