ECHOGS: COMPACT 3D GAUSSIAN SPLATTING VIA CONDITIONAL RESPONSE CREDIT

Anonymous authors
Paper under double-blind review

ABSTRACT

Compact 3D Gaussian Splatting (3DGS) methods commonly score each Gaussian independently or merge primitives according to parameter similarity. Neither strategy answers the basic compression question: which rendered effects remain irreplaceable after nearby substitutes are considered? We introduce EchoGS, a three-stage compression pipeline centered on conditional response credit. Stage I collects demand to identify where reconstruction capacity is needed. The core Stage II encodes each Gaussian's multi-view rendering behavior in a compact, view-tagged signature and jointly compares signatures within spatial neighborhoods. Local ridge leverage measures each primitive's contribution to response coverage given its neighbors: repeated directions share credit, while sufficiently supported complementary directions retain value. Blending it with marginal demand produces the allocation score passed to Stage III, which distributes the target budget and fits replacement Gaussians by matching assigned geometric moments. In the reported comparison with vanilla 3DGS, the default configuration achieves $9.01\times$–$11.58\times$ count compression on three datasets with competitive rendering quality.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) renders high-quality novel views in real time, but often stores millions of anisotropic primitives (Kerbl et al., 2023; Yao et al., 2026; Zhao et al., 2026). Their number directly affects training time, memory and storage requirements, and rendering speed (Fei et al., 2024; Wu et al., 2024; Bao et al., 2025; Bagdasarian et al., 2025), motivating methods for compacting Gaussian representations (Mallick et al., 2024; Fang & Wang, 2024a; Fan et al., 2024).

The central compression decision is not whether a Gaussian is important in isolation but whether it remains necessary *given the other Gaussians*. A primitive may strongly affect many pixels and still duplicate a neighbor; another may have a weak total response yet be the only primitive supporting one view. For example, responses $(1, 0), (1, 0)$ have the same per-row magnitude as $(1, 0), (0, 1)$, but the first pair repeats one rendering mode while the second preserves two. A score based only on the row norm or coordinate sum assigns identical scores to all four rows and cannot distinguish the two systems.

Many compact 3DGS pipelines rely on per-primitive importance scoring or geometric aggregation. Methods such as Mini-Splatting (Fang & Wang, 2024a), Taming 3DGS, and FastGS (Mallick et al., 2024; Ren et al., 2026) use rendering evidence to compute an importance score for each primitive, but do not explicitly measure the response coverage lost when one primitive is removed from its candidate set. Geometric aggregation methods such as GHAP (Wang et al., 2025) relate nearby means and covariances, but geometric distances alone do not explicitly compare rendered responses under visibility, depth ordering, alpha compositing, and view-dependent appearance. Such importance scores are marginal, whereas substitutability depends on the neighboring responses. Tab. 1 summarizes this distinction between existing methods.

We introduce EchoGS, a three-stage pipeline that turns reconstruction demand into a compact Gaussian representation through conditional response scoring. Stage I compares rendered and training images to identify residual demand and records which Gaussians support those pixels in each sampled view. Stage II takes these per-view demand counts together with visibility, support, position, depth, and color features to describe how each primitive participates in rendering. It compresses these observations into a fixed-dimensional signature using a separate random projection for each camera.

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.

Table 1: Functional comparison of compact pipelines. Renderer evidence comes from rendered observations; primitive relation denotes explicit geometric or response-space affinities between source primitives, beyond compositing or scalar-score ranking; view identity is retained camera identity in that comparison. Conditional credit measures a primitive's additional response coverage given its candidate set; aggregation fits replacement Gaussians to source groups.

Method	Renderer evidence	Primitive relation	View identity	Conditional credit	Aggregation
LightGaussian Fan et al. (2024)	✓	–	–	–	–
Mini-Splatting Fang & Wang (2024a)	✓	–	–	–	–
Taming 3DGS Mallick et al. (2024)	✓	–	–	–	–
Speedy-Splat Hanson et al. (2025a)	✓	–	–	–	–
FastGS Ren et al. (2026)	✓	–	–	–	–
GHAP Wang et al. (2025)	–	✓	–	–	✓
QIRF Jiang et al. (2026)	✓	✓	–	–	–
EchoGS (ours)	✓	✓	✓	✓	✓

The core of Stage II is a joint comparison of these signatures within spatial neighborhoods. The normalized conditional score is blended with Stage-I demand, connecting evidence uniqueness to reconstruction need. Stage III uses this score to distribute the target budget across spatial groups and fits replacement Gaussians by matching assigned geometric moments. Thus, demand enters the response comparison, conditional credit determines where capacity is allocated, and geometric aggregation realizes that allocation in the reduced representation.

Our contributions are:

- We formulate 3DGS compression via conditional response creadit and introduce EchoGS, a three-stage pipeline that connects reconstruction demand, response uniqueness, and geometric aggregation.
- We propose Conditional Response Credit (CRC), which combines view-tagged response signatures with local ridge leverage to measure each Gaussian's additional response coverage given its neighbors.
- In the reported results, our default configuration achieves $9.01\times$–$11.58\times$ Gaussian-count compression relative to vanilla 3DGS across three datasets while maintaining competitive rendering quality.

2 Related Work

Efficient and compact 3DGS. Efficiency methods alter optimization schedules, primitive growth, rendering, or representation. DashGaussian accelerates training through a dynamic schedule (Chen et al., 2025), while 3DGS-LM and 3DGS^2 use second-order optimization structure to reduce convergence time (Höllein et al., 2025; Lan et al., 2025). Speedy-Splat, FlashGS, and FastGS reduce unnecessary rendering computation or primitive growth (Hanson et al., 2025a; Feng et al., 2025; Ren et al., 2026); StopThePop improves view-consistent rasterization order (Radl et al., 2024). Structured representations such as Octree-GS and Scaffold-GS organize or generate Gaussians from spatial anchors (Ren et al., 2024; Lu et al., 2024), whereas GaussianPro, AtomGS, and 3DGS as MCMC modify propagation, atomization, or optimization dynamics (Cheng et al., 2024; Liu et al., 2024; Kheradmand et al., 2024). LightGaussian, Compact-3DGS, RadSplat, HAC, and Trimming-the-Fat reduce storage through pruning, quantization, or entropy-aware coding (Fan et al., 2024; Lee et al., 2024; Niemeyer et al., 2025; Chen et al., 2024; Ali et al., 2024); Taming 3DGS, Mini-Splatting, and Mini-Splatting2 control primitive density by sampling, importance, or aggressive densification (Mallick et al., 2024; Fang & Wang, 2024a;b). PUP 3D-GS uses a sensitivity approximation for pruning (Hanson et al., 2025b), while LP-3DGS learns a binary mask (Zhang et al., 2024). Recent 3DGS surveys organize these rapidly expanding optimization, representation, and compression choices (Fei et al., 2024; Wu et al., 2024; Bao et al., 2025; Bagdasarian et al., 2025). Several of these methods use per-primitive importance signals. Most directly, FastGS averages the number of high-error pixels in a Gaussian's footprint over sampled views and uses the resulting scalar for densification and pruning (Ren et al., 2026). This is an effective demand signal, but it does not compare one Gaussian's evidence with that of another. EchoGS retains the view-conditioned responses and performs that comparison explicitly.

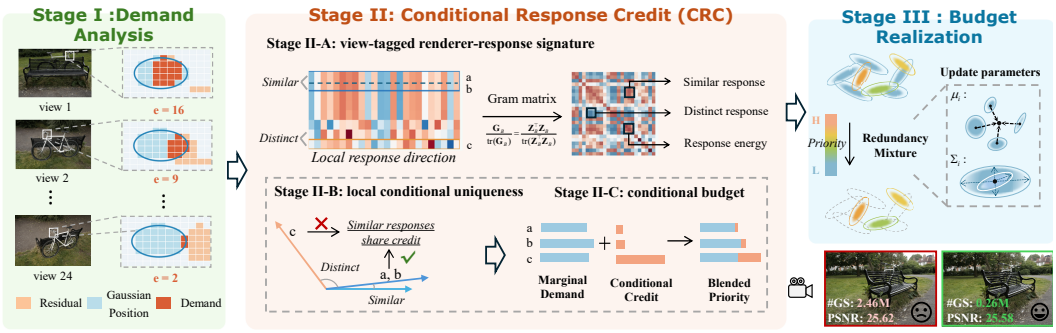

Figure 1: **EchoGS framework.** Stage I counts the intersection of a Gaussian's rasterized support with a residual mask. Stage II sketches view-tagged responses, summarizes their coordinate co-response by a Gram matrix, and blends conditional credit with marginal demand. The Gram axes index sketch coordinates: diagonal entries measure energy and off-diagonal entries measure signed co-response. Stage III allocates replacement capacity and updates geometric moments. Warm colors indicate higher priority; dashed outlines depict absorbed sources.

Gaussian aggregation. GHAP (Wang et al., 2025) reduces spatial Gaussian mixtures using geometric parameter distances; EchoGS instead allocates aggregation budgets using conditional response credit. QIRF (Jiang et al., 2026) also assigns a selection score to each Gaussian and computes these scores jointly from nearby primitives but does not directly score the response coverage lost when a Gaussian is removed. EchoGS uses camera-specific response sketches and row-wise ridge leverage (Drineas et al., 2012; Cohen et al., 2017) to measure each primitive's contribution given its neighbors.

3 Methodology

3.1 Problem Formulation

A scene contains N Gaussians $\mathcal{G} = \{g_i\}_{i=1}^{N}$. g_i has mean $\boldsymbol{\mu}_i$, covariance $\boldsymbol{\Sigma}_i$, opacity α_i, and spherical-harmonic (SH) coefficients. For camera v, the differentiable renderer produces $\hat{\boldsymbol{I}}_v = \mathcal{R}_v(\mathcal{G})$. Given a target budget $K < N$, reduction seeks a set $\widetilde{\mathcal{G}}$ with at most K primitives that approximately preserves $\{\mathcal{R}_v(\mathcal{G})\}_{v=1}^{V}$.

Fig. 1 summarizes how the method passes one budget through three distinct roles. **Stage I** marks evidence associated with reconstruction error in sampled views. **Stage II** estimates substitutability by comparing each primitive's response coverage with that of nearby primitives. **Stage III** turns the resulting budget into a smaller Gaussian mixture.

3.2 Stage I: Demand Interface

Stage I identifies where additional reconstruction capacity is requested. For view v, let E^v_{rgb} be the normalized RGB residual and E^v_{str} the mismatch between normalized first- and second-order luminance responses of the rendered and target images. We activate a pixel only when intensity error coincides with misplaced image structure:

$$\boldsymbol{M}_v = \mathbf{1}[E^v_{\mathrm{rgb}} > \tau] \odot \mathbf{1}[E^v_{\mathrm{str}} > \tau/2], \tag{1}$$

where $\tau = 0.1$. The RGB residual measures photometric inconsistency, while the filter response highlights structurally localized errors. Let $u^v_i(p) \in \{0, 1\}$ indicate that the flagged rasterization reports primitive i as influencing pixel p. Its view-specific demand is

$$e^v_i = \sum_{p \in \Omega_v} \boldsymbol{M}_v(p) u^v_i(p), \qquad b_i = \left\lfloor \frac{1}{V} \sum_{v=1}^{V} e^v_i \right\rfloor. \tag{2}$$

The average b_i supplies the base growth signal.

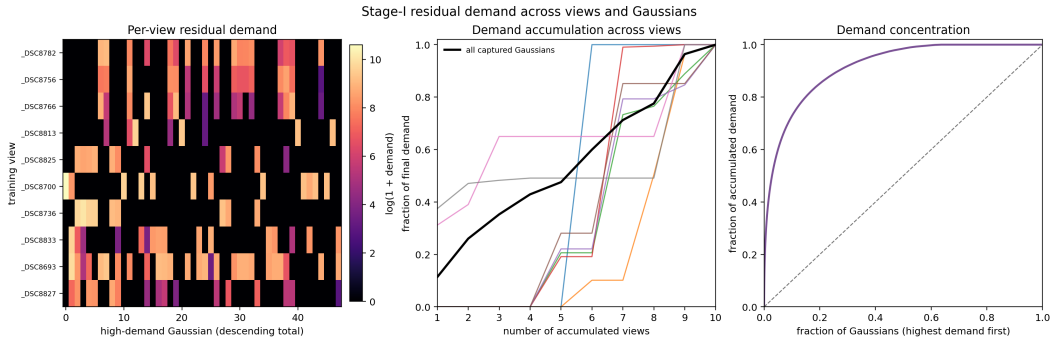

Figure 2: **Stage I: residual demand analysis. Left:** rows are ten sampled training cameras; columns are the 48 captured Gaussians with largest total demand. Color shows e_i^v (Eq. (2)); brighter cells mean larger counts. **Middle:** each colored curve shows cumulative demand for one of the eight highest-demand Gaussians in the displayed camera order, normalized by its total demand. Jumps indicate cameras supplying much of the demand. **Right:** Gaussians are sorted by total demand; the curve gives the fraction of captured demand carried by the top fraction of captured primitives.

3.3 Stage II-A: View-tagged renderer-response signature

Stage II retains $(e_i^1, \ldots, e_i^V)$ and compares the resulting view-conditioned responses jointly. We describe each primitive using renderer-derived response features. For training view v, we record visibility q_i^v, accumulated alpha support a_i^v, projected footprint A_i^v, and the Stage-I demand count e_i^v. Normalized image position (x_i^v, y_i^v), camera-relative depth d_i^v, and view-dependent color $\boldsymbol{c}_i^v$ help distinguish primitives that have similar total support but participate at different pixels, depths, or appearances. We define $h(t) = \min\{8, \log(1 + t/a_t)\}$. The ten-dimensional per-view response is

$$\boldsymbol{x}_i^v = \left[q_i^v,\ h(a_i^v),\ h(e_i^v),\ h(A_i^v),\ q_i^v x_i^v,\ q_i^v y_i^v,\ q_i^v d_i^v,\ h(a_i^v)(\boldsymbol{c}_i^v)^\top\right]^\top \in \mathbb{R}^{10}. \tag{3}$$

Each channel is RMS-normalized over all Gaussians. We assign every camera an independent fixed Gaussian projection $\boldsymbol{P}_v \in \mathbb{R}^{10\times d}$, with entries sampled from $\mathcal{N}(0, 1/d)$, and form

$$\boldsymbol{s}_i = \frac{1}{\sqrt{V}} \sum_{v=1}^{V} \boldsymbol{P}_v^\top \boldsymbol{D}_v^{-1} \boldsymbol{x}_i^v, \qquad \boldsymbol{z}_i = \mathcal{N}_{\mathrm{RMS}}(\boldsymbol{s}_i) \in \mathbb{R}^d. \tag{4}$$

We set $d = 24$ by default. Independent camera projections yield a compact sketch whose inner products retain same-view response similarity in expectation before the final RMS normalization. For independent zero-mean projections, the pre-normalization sketch has the exact expectation

$$\mathbb{E}[\boldsymbol{s}_i^\top \boldsymbol{s}_j] = \frac{1}{V} \sum_{v=1}^{V} (\boldsymbol{D}_v^{-1} \boldsymbol{x}_i^v)^\top (\boldsymbol{D}_v^{-1} \boldsymbol{x}_j^v). \tag{5}$$

3.4 Stage II-B: Local conditional uniqueness

Similar responses from distant surfaces need not be geometrically substitutable. We therefore restrict comparisons to spatial blocks of an importance-balanced KD-tree with split mass

$$m_i = \alpha_i \left[\mathcal{N}(c_i) + \mathcal{N}(b_i) + \epsilon_m\right], \tag{6}$$

where c_i is accumulated renderer contribution, b_i is marginal residual demand, $\mathcal{N}$ is scene-wise min–max normalization, and $\epsilon_m = 0.1$. Within score block B, let $\boldsymbol{D}_B$ hold the column RMS scales of $\{\boldsymbol{z}_i : i \in B\}$, and let a_B be the median positive mass, floored at 10^{-6} (or 1 if all masses vanish). Define

$$\rho_i = \log\left(1 + \frac{m_i}{a_B}\right), \qquad \widetilde{\boldsymbol{z}}_i = \sqrt{\kappa + \rho_i}\, \boldsymbol{D}_B^{-1} \boldsymbol{z}_i. \tag{7}$$

With $\kappa = 0.25$, stack $\widetilde{\boldsymbol{z}}_i^\top$ as rows of $\boldsymbol{Z}_B$ and set $\boldsymbol{G}_B = \boldsymbol{Z}_B^\top \boldsymbol{Z}_B$. The conditional credit is

$$\ell_i = \widetilde{\boldsymbol{z}}_i^\top (\boldsymbol{G}_B + \lambda_B \boldsymbol{I})^{-1} \widetilde{\boldsymbol{z}}_i, \tag{8}$$

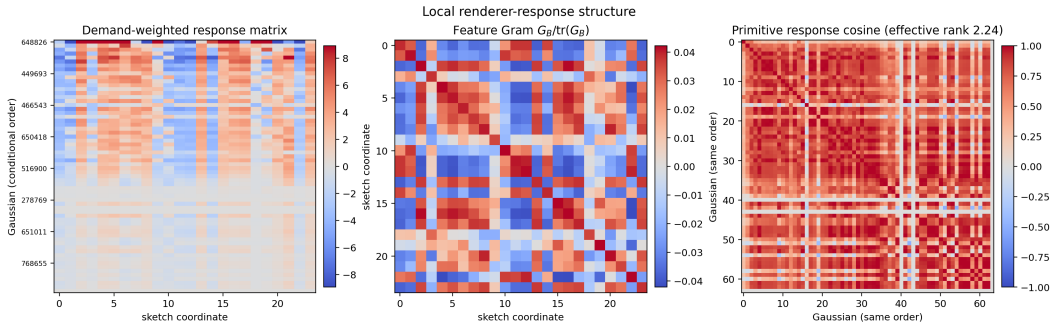

Figure 3: **Stage-II-A: Local renderer-response structure. Left:** 64 selected mass-weighted response rows $\widetilde{\boldsymbol{z}}_i^\top$ are ordered by decreasing conditional score. **Middle:** Visualization of $\boldsymbol{G}_B/\operatorname{tr}(\boldsymbol{G}_B)$. Diagonal entries give relative coordinate energy; off-diagonal entries give signed co-response. **Right:** the 64×64 cosine matrix compares the displayed rows in the same order: values near one indicate aligned response directions; values near zero indicate near-orthogonality.

where $\lambda_B = \lambda \max\{\operatorname{tr}(\boldsymbol{G}_B)/d, 10^{-8}\}$ and $\lambda = 10^{-3}$. The joint Gram makes this credit depend on all candidate substitutes. For retention weights $w_j \geq 0$, define response coverage as

$$F_B(\boldsymbol{w}) = \log\det\left(\lambda_B \boldsymbol{I} + \sum_{j\in B} w_j \widetilde{\boldsymbol{z}}_j \widetilde{\boldsymbol{z}}_j^\top\right). \tag{9}$$

Proposition 1 (Conditional credit and finite removal)**.** *Let $\boldsymbol{A}_B = \lambda_B \boldsymbol{I} + \sum_{j\in B} \widetilde{\boldsymbol{z}}_j \widetilde{\boldsymbol{z}}_j^\top$ with $\lambda_B > 0$, and hold the response scaling and ridge fixed. Then:*

1. $\partial F_B(\boldsymbol{w})/\partial w_i|_{\boldsymbol{w}=\mathbf{1}} = \ell_i$;
2. $0 \leq \ell_i < 1$, *and removing row i incurs the exact coverage loss*
$$F_B(\mathbf{1}) - F_B(\mathbf{1} - \boldsymbol{e}_i) = -\log(1-\ell_i); \tag{10}$$
3. *if additional substitutes contribute $\boldsymbol{H} \succeq 0$ to the Gram matrix, then*
$$\widetilde{\boldsymbol{z}}_i^\top (\boldsymbol{G}_B + \boldsymbol{H} + \lambda_B \boldsymbol{I})^{-1} \widetilde{\boldsymbol{z}}_i \leq \ell_i. \tag{11}$$

Thus ℓ_i ranks single-row coverage losses without recomputing each reduced system: neighbors with sufficient energy along the same response direction lower its credit. The guarantee concerns raw ℓ_i, before re-normalization or blending.

3.5 Stage II-C: Conditional Budget

Pure algebraic uniqueness is not sufficient: a tiny noisy response may be unique but irrelevant. The factor $\sqrt{\kappa + \rho_i}$ in $\widetilde{\boldsymbol{z}}_i$ couples span novelty to renderer support, while the Stage-I coordinate e_i^v couples it to reconstruction demand. Within each block, leverage and log mass are max-normalized, $\bar{\ell}_i = \ell_i / \max\{\max_{j\in B} \ell_j, 10^{-8}\}$ and $\bar{\rho}_i = \rho_i / \max\{\max_{j\in B} \rho_j, 10^{-8}\}$. The scene-level score is

$$r_i = \mathcal{N}\left(\bar{\ell}_i + \eta \bar{\rho}_i\right), \qquad s_i = (1-\beta)\mathcal{N}(b_i) + \beta r_i, \tag{12}$$

where $\eta = 0.05$ and $\beta = 0.25$ by default. The additive term retains a small mass-based preference when leverage alone is uninformative. Scene-wise min–max normalization is applied only after all block scores are assembled. At $\beta = 0$, allocation follows marginal demand; increasing β progressively assigns budget according to conditional response coverage.

3.6 STAGE III: BUDGET REALIZATION

Stage III separates capacity allocation from geometric fitting. With normalized opacity $\pi_i = \alpha_i / \sum_j \alpha_j$, renderer contribution c_i, and scene-normalized $\bar{s}_i, \bar{c}_i$, define

$$w_i^{\text{alloc}} = \pi_i(\bar{s}_i + \bar{c}_i + \epsilon_m), \qquad W_B = \sum_{i \in B} w_i^{\text{alloc}}. \tag{13}$$

The first quantity balances spatial blocks; the second assigns each block a share $W_B / \sum_{B'} W_{B'}$ of the target budget K. Within each block, sources are assigned to the nearest target under

$$d^2_{\text{base}}(i,k) = \|\boldsymbol{\mu}_i - \widetilde{\boldsymbol{\mu}}_k\|_2^2 + \|\boldsymbol{\Sigma}_i^{1/2} - \widetilde{\boldsymbol{\Sigma}}_k^{1/2}\|_F^2. \tag{14}$$

Split π_i equally among tied nearest targets to obtain assignment masses γ_{ik}. For $C_k = \{i : \gamma_{ik} > 0\}$, set $\omega_{ik} = \gamma_{ik} / \sum_{j \in C_k} \gamma_{jk}$ and update

$$\widetilde{\boldsymbol{\mu}}_k = \sum_{i \in C_k} \omega_{ik} \boldsymbol{\mu}_i, \quad \widetilde{\boldsymbol{\Sigma}}_k = \sum_{i \in C_k} \omega_{ik} \left[\boldsymbol{\Sigma}_i + (\boldsymbol{\mu}_i - \widetilde{\boldsymbol{\mu}}_k)(\boldsymbol{\mu}_i - \widetilde{\boldsymbol{\mu}}_k)^\top \right]. \tag{15}$$

Here w_i^{alloc} determines capacity, while ω_{ik} is normalized assigned opacity, equal to $\alpha_i / \sum_{j \in C_k} \alpha_j$ without ties.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and metrics. We evaluate on seven Mip-NeRF 360 scenes (Barron et al., 2022), two Tanks & Temples scenes (Knapitsch et al., 2017), two Deep Blending scenes (Hedman et al., 2018), and six Waymo scenes (Sun et al., 2020). We report PSNR, SSIM (Wang et al., 2004), LPIPS (Zhang et al., 2018), training time, final Gaussian count N_{GS}, FPS, and peak GPU memory.

Baselines. We compare with 3DGS (Kerbl et al., 2023), LightGaussian (Fan et al., 2024), Compact-3DGS (Lee et al., 2024), RadSplat (Niemeyer et al., 2025), DashGaussian (Chen et al., 2025), Speedy-Splat (Hanson et al., 2025a), GHAP (Wang et al., 2025), Mini-Splatting (Fang & Wang, 2024a), Taming 3DGS (Mallick et al., 2024), and FastGS (Ren et al., 2026).

Implementation. All experiments are conducted in PyTorch and run on one NVIDIA A6000 GPU. We redeploy all baselines on the same device and the same CUDA version. We follow the standard protocols used in prior work, adopting the same training/testing splits and evaluation metrics to ensure comparability. For GHAP, we use vanilla 3DGS as its baseline. The loss function for training is the same as vanilla 3DGS. Additional results and peak GPU memory measurements are provided in Appendix E. Appendix B reports default settings and pseudocode. We organize our method into **Turbo**, **Balanced**, and **Quality**, in this order. Turbo targets shorter training time and smaller Gaussian count, accepting lower rendering fidelity. Balanced is the default configuration unless otherwise specified and balances efficiency and fidelity; Quality prioritizes reconstruction accuracy with a larger capacity allowance.

4.2 COMPARISON RESULTS

4.2.1 QUANTITATIVE COMPARISON

Tab. 2–4, together with the rendering rates in Tab. 14, compare reconstruction quality, primitive counts and computational cost. The reported results show that compactness does not mean accepting lower fidelity. Relative to vanilla 3DGS, we achieve $11.58\times$ and $9.01\times$ count compression on Deep Blending and Tanks & Temples, with PSNR gains of 0.46 and 0.33 dB, respectively. The comparisons below identify where this advantage also holds against the stronger compact and accelerated baselines.

Balanced is particularly competitive with FastGS at a similar training cost. On Deep Blending, it uses 0.21M rather than 0.32M Gaussians, increases PSNR from 30.04 to 30.20 dB and raises rendering speed from 359 to 391 FPS. On Tanks & Temples, it reduces the count from 0.25M to 0.17M while

Table 2: Efficiency–quality trade-off. Compression ratio is the mean vanilla-3DGS count divided by the method mean count; ΔPSNR is the difference of dataset-mean PSNR values.

Method	Deep Blending		Tanks & Temples		Mip-NeRF 360	
	Ratio ↑	ΔPSNR ↑	Ratio ↑	ΔPSNR ↑	Ratio ↑	ΔPSNR ↑
DashGaussian	1.27×	-0.13	1.30×	+0.18	1.11×	+0.06
Speedy-Splat	5.13×	-0.11	7.48×	-0.31	4.75×	-0.17
GHAP	5.02×	+0.00	5.06×	-0.57	5.04×	-0.54
Mini-Splatting	4.39×	+0.27	5.23×	-0.33	5.26×	-0.04
Taming 3DGS	8.48×	+0.09	4.91×	+0.04	3.74×	-0.37
FastGS	7.69×	+0.30	6.28×	+0.30	5.88×	-0.15
Ours–Turbo	**13.07**×	+0.19	**9.98**×	-0.14	**10.30**×	-0.92
Ours–Balanced	11.58×	+0.46	9.01×	+0.33	9.52×	-0.18
Ours–Quality	5.31×	**+0.52**	4.60×	**+0.45**	4.36×	**+0.09**

Table 3: Quantitative comparison of baseline methods and the three EchoGS operating presets. Time is training time in minutes; N_{GS} is in millions. Bold and underline denote the best and second-best displayed values in each column.

Dataset	Mip-NeRF 360 Barron et al. (2022)					Tanks & Temples Knapitsch et al. (2017)					Deep Blending Hedman et al. (2018)				
Metric	Time↓	N_{GS} ↓	PSNR↑	SSIM↑	LPIPS↓	Time↓	N_{GS} ↓	PSNR↑	SSIM↑	LPIPS↓	Time↓	N_{GS} ↓	PSNR↑	SSIM↑	LPIPS↓
3DGS	30.46	2.47	29.06	0.870	0.185	17.80	1.57	23.76	0.850	**0.180**	30.18	2.46	29.74	0.903	0.250
LightGaussian	27.20	0.79	27.15	0.817	0.253	16.50	0.47	23.15	0.820	0.229	25.73	0.57	27.25	0.877	0.298
Compact-3DGS	34.13	1.35	27.46	0.816	0.246	20.45	0.83	23.26	0.831	0.201	30.50	1.04	29.67	0.900	0.258
RadSplat	27.83	0.71	27.54	0.825	0.239	17.44	0.42	23.38	0.831	0.208	28.02	0.90	29.98	**0.908**	0.255
DashGaussian	10.81	2.22	29.12	0.872	0.186	7.22	1.21	23.94	0.847	0.181	7.18	1.94	29.61	0.902	0.249
Speedy-Splat	24.34	0.52	28.89	0.869	0.198	13.50	0.21	23.45	0.820	0.240	18.60	0.48	29.63	0.905	0.256
GHAP	21.33	0.49	28.52	0.856	0.219	12.13	0.31	23.19	0.828	0.217	20.26	0.49	29.74	0.902	0.255
Mini-Splatting	26.07	0.47	29.02	0.869	**0.170**	16.63	0.30	23.43	0.844	0.181	21.84	0.56	30.01	0.907	**0.243**
Taming 3DGS	6.78	0.66	28.69	0.852	0.223	4.85	0.32	23.80	0.833	0.212	4.05	0.29	29.83	0.899	0.274
FastGS	4.15	0.42	28.91	0.864	0.207	2.84	0.25	24.06	0.835	0.212	2.98	0.32	30.04	0.904	0.257
Ours–Turbo	**2.49**	**0.24**	28.14	0.843	0.239	**1.83**	**0.16**	23.62	0.825	0.229	**1.88**	**0.19**	29.93	0.899	0.275
Ours–Balanced	3.98	0.26	28.88	0.862	0.214	2.87	0.17	24.09	0.836	0.211	2.97	0.21	30.20	0.907	0.263
Ours–Quality	6.13	0.57	**29.15**	**0.887**	0.182	4.03	0.34	**24.21**	**0.852**	0.181	4.03	0.46	**30.26**	**0.908**	0.250

improving PSNR, SSIM, and LPIPS to 24.09, 0.836, and 0.211. Mip-NeRF 360 exposes a different balance between fidelity and capacity. Balanced uses 0.26M Gaussians versus FastGS's 0.42M, trains in 3.98 versus 4.15 minutes, and incurs a 0.03 dB PSNR decrease from 28.91 to 28.88 dB. Against Taming 3DGS, it improves PSNR from 28.69 to 28.88 dB while reducing the count from 0.66M to 0.26M and training time from 6.78 to 3.98 minutes; rendering speed increases from 160 to 341 FPS. This comparison highlights the benefit of using a compact representation while preserving most of the reconstruction fidelity.

Quality attains the highest reported PSNR on all three datasets. On Mip-NeRF 360, Quality improves PSNR from 29.12 to 29.15 dB relative to DashGaussian while using fewer primitives and reducing training time. Quality spends more capacity than Balanced, whereas Turbo prioritizes training efficiency: it finishes in 1.83–2.49 minutes and has the smallest count on each of these datasets, yielding $9.98\times$–$13.07\times$ compression relative to 3DGS.

The Waymo results in Tab. 4 extend this comparison to larger driving scenes. Balanced has the smallest reported count and shortest training time: compared with FastGS, it uses 0.25M rather than 0.42M Gaussians, trains in 4.79 rather than 5.67 minutes, and renders at 342 rather than 283 FPS. It also improves PSNR/SSIM/LPIPS from 26.88/0.826/0.349 to 26.91/0.829/0.345.

4.2.2 Qualitative comparison

Fig. 4 compares the archived demand/reduction base with Taming 3DGS, Mini-Splatting, Speedy-Splat, and FastGS on outdoor and indoor scenes. In the vehicle example, the lettering on the rear plate remains more separated in our result. In the garden example, our reconstruction retains more distinct roof and chimney boundaries. In the indoor example, the circular marking on the small object remains visible, whereas FastGS smooths much of this local structure. Across these examples, our crops more closely preserve the local contrast and boundaries visible in the ground truth. These details test complementary aspects of compact reconstruction: narrow character strokes require localized support, distant roof edges require separation across depth, and small indoor patterns require capacity

Table 4: Waymo comparison of baseline methods with Balanced and Quality. Time is in minutes and N_{GS} is in millions.

Method	Time↓	N_{GS} ↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
Speedy-Splat	31.54	0.48M	25.14	0.805	0.363	216
GHAP	31.21	0.38M	26.42	0.824	0.345	**361**
Mini-Splatting	43.44	0.30M	23.42	0.784	0.391	360
Taming 3DGS	10.18	0.36M	26.78	0.824	0.351	146
FastGS	5.67	0.42M	26.88	0.826	0.349	283
Ours–Balanced	**4.79**	**0.25M**	26.91	0.829	0.345	342
Ours–Quality	6.53	0.52M	**27.22**	**0.831**	**0.332**	301

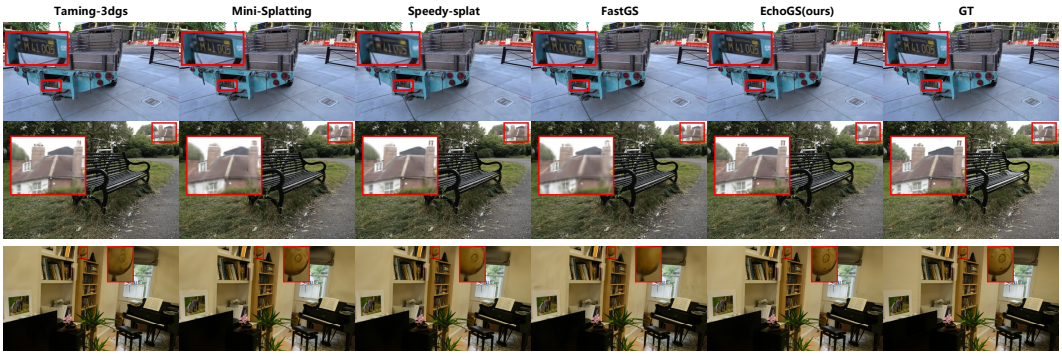

Figure 4: Visualization of Mip-NeRF 360, Tanks & Temples, and Deep Blending qualitative comparisons. Red boxes mark fine text, distant roof edges, and indoor object detail.

within an otherwise smooth region. Preserving them is the intended benefit of allocating primitives according to useful rendering evidence and realizing that allocation through aggregation.

4.3 Ablation Study and Analysis

Tab. 5 compares eight component variants with Ours–Balanced on Mip-NeRF 360. These comparisons describe quality–efficiency trade-offs rather than quality differences at a fixed count. Appendices E.2–E.4 examine parameters sensitivity about response dimension, error threshold, and mass floor.

Stage I ablation. Removing multi-view accumulation reduces PSNR to 28.84 dB, increases LPIPS to 0.218, raises the Gaussian count to 0.33M, and lowers rendering speed to 296 FPS, consistent with multi-view accumulation retaining demand from support that is weak or occluded in individual views. Using photometric demand alone likewise produces 0.29M Gaussians, with 28.86 dB PSNR and 0.220 LPIPS. Although its SSIM (0.863) is slightly higher than that of the full model (0.862), the gated variant has a smaller count and lower LPIPS in this comparison, consistent with focusing demand on localized reconstruction errors.

Stage II ablation. Sharing the projection across cameras reduces PSNR to 28.80 dB and SSIM to 0.859, suggesting that camera-specific tags help prevent responses from different visibility conditions from being treated as substitutes. Replacing conditional leverage with response norm gives 28.86 dB PSNR and 0.219 LPIPS, The 0.02 dB PSNR difference is small; the lower LPIPS and count of the full model are consistent with comparing shared response directions rather than response magnitude alone. Computing leverage in parameter space further lowers fidelity to 28.75 dB PSNR and 0.224 LPIPS, consistent with renderer-derived features being more informative about substitutability in this comparison. Finally, conditional-only allocation ($\beta = 1$) reduces the model to 0.25M Gaussians and increases rendering speed to 349 FPS, but degrades PSNR to 28.79 dB and LPIPS to 0.222. The blended setting trades a small increase in capacity for higher quality in these comparisons.

Stage III ablation. Using median KD blocks reduces PSNR to 28.55 dB and increases LPIPS to 0.244. In this comparison, the median-KD variant has lower fidelity despite retaining more primitives.

Table 5: **Component ablations on Mip-NeRF 360 for Ours–Balanced.** Time is in minutes and counts are in millions. Higher is better (↑), lower is better (↓).

Variant	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑
Ours (Full)	3.98	0.26M	**28.88**	0.862	**0.214**	341
Stage I: residual demand						
↪ w/o multi-view accumulation	4.07	0.33M	28.84	0.861	0.218	296
↪ w/o structural gate	4.05	0.29M	28.86	**0.863**	0.220	312
Stage II: conditional allocation						
↪ w/o view tags	3.95	0.28M	28.80	0.859	0.219	324
↪ w/o conditional leverage	**3.94**	0.29M	28.86	0.860	0.219	327
↪ Parameter-space leverage	3.98	0.29M	28.75	0.858	0.224	319
↪ Conditional-only allocation	3.99	**0.25M**	28.79	0.858	0.222	**349**
Stage III: budget realization						
↪ Median KD blocks	4.00	0.28M	28.55	0.858	0.244	292
↪ w/o mixture reduction	3.98	0.32M	28.78	0.860	0.217	307

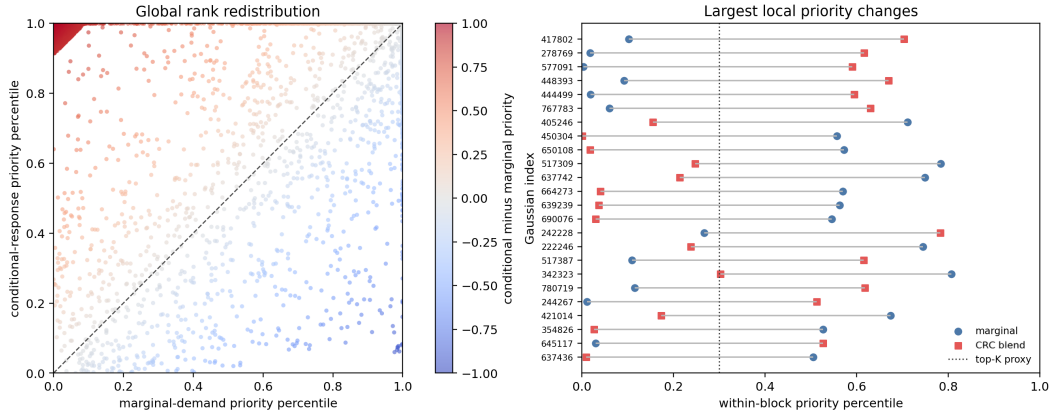

Figure 5: **Marginal and conditional priorities. Left:** 4,096 selected samples compare marginal demand b_i (horizontal) with the conditional score r_i (vertical), before blending. Points above/below the dashed equality line are promoted/demoted; color encodes the same conditional-minus-marginal percentile change. **Right:** the 24 largest absolute percentile shifts within the selected diagnostic block compare marginal demand (blue circles) with the blended score s_i (red squares). Gray segments connect each primitive's two priorities. The dotted boundary marks the top-budget ranking proxy.

Replacing moment-based aggregation with representative pruning yields 28.78 dB PSNR and 0.217 LPIPS with 0.32M Gaussians. Moment matching preserves the assigned mixture's first two geometric moments; the aggregated variant has higher quality with fewer primitives in this comparison, without implying exact preservation of rendered appearance.

5 Conclusion

We propose EchoGS, which connects residual demand, conditional response credit, and geometric mixture reduction in a coherent compression pipeline. Its central contribution is Stage II: view-tagged signatures describe how Gaussians participate in rendering, and a local Gram system evaluates each response jointly with its potential substitutes. The resulting ridge-leverage credit discounts repeated evidence and values supported complementary responses, with a precise connection to log-determinant coverage and single-row removal loss under fixed scaling and ridge. Stage I supplies reconstruction demand, and Stage III realizes the blended allocation score through replacement Gaussians that match assigned geometric moments. Tab. 2 reports $11.58\times$, $9.01\times$, and $9.52\times$ count compression for Balanced on Deep Blending, Tanks & Temples, and Mip-NeRF 360, with competitive rendering quality and speed.

REPRODUCIBILITY STATEMENT

The main paper specifies the scoring construction and its reduction interface. Default configurations are given below. Appendix B provides pseudocode and implementation-level details; Appendix A supplies complete proofs. Code and exact commands will be released upon publication.

AI USE STATEMENT

Generative AI tools assisted literature organization, language editing, LaTeX drafting, review of mathematical derivations, experiment planning, and interpretation of results. They also assisted preparation of provisional numerical estimates during drafting; such estimates are not experimental measurements. The authors remain responsible for the method, verification of results and derivations, citations, and all submitted content.

ETHICS AND BROADER IMPACT

Compact scene representations can reduce compute, energy, memory, and device requirements. Like the underlying 3DGS representation, the method may also be applied to captured people or private environments. Users should obtain appropriate consent and follow dataset licenses. The proposed score does not introduce a new data modality or an autonomous decision-making component.

REFERENCES

Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3d gaussian splats through pruning. *arXiv preprint arXiv:2406.18214*, 2024.

Milena T Bagdasarian, Paul Knoll, Y Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 3dgs. zip: A survey on 3d gaussian splatting compression methods. *Computer Graphics Forum*, 44(2):e70078, 2025.

Yanqi Bao, Tianyu Ding, Jing Huo, Yaoli Liu, Yuxin Li, Wenbin Li, Yang Gao, and Jiebo Luo. 3d gaussian splatting: Survey, technologies, challenges, and opportunities. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

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*, pp. 5470–5479, 2022.

Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. On the bures–wasserstein distance between positive definite matrices. *Expositiones mathematicae*, 37(2):165–191, 2019.

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*, pp. 422–438. Springer, 2024.

Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. Dash-gaussian: Optimizing 3d gaussian splatting in 200 seconds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 11146–11155, 2025.

Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. In *Forty-first International Conference on Machine Learning*, 2024.

Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1758–1777. SIAM, 2017.

Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13: 3475–3506, 2012.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.

Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*, pp. 165–181. Springer, 2024a.

Guangchi Fang and Bing Wang. Mini-splatting2: Building 360 scenes within minutes via aggressive gaussian densification. *arXiv preprint arXiv:2411.12788*, 2024b.

Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3d gaussian splatting as new era: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 2024.

Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Linning Xu, Zhilin Pei, Hengjie Li, et al. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26652–26662, 2025.

Matthias Gelbrich. On a formula for the l2 wasserstein metric between measures on euclidean and hilbert spaces. *Mathematische Nachrichten*, 147(1):185–203, 1990.

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 Computer Vision and Pattern Recognition Conference*, pp. 21537–21546, 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 Computer Vision and Pattern Recognition Conference*, pp. 5949–5958, 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 (ToG)*, 37(6):1–15, 2018.

Lukas Höllein, Aljaž Božič, Michael Zollhöfer, and Matthias Nießner. 3dgs-lm: Faster gaussian-splatting optimization with levenberg-marquardt. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 26740–26750, 2025.

Shizeng Jiang, Hao Zhang, Xuerui Ma, Ying Hu, and Tao Zhang. QIRF: Quantum-inspired non-orthogonal function-space compression for 3d gaussian splatting. *arXiv preprint arXiv:2607.18067*, 2026.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

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. *Advances in Neural Information Processing Systems*, 37:80965–80986, 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.

Lei Lan, Tianjia Shao, Zixuan Lu, Yu Zhang, Chenfanfu Jiang, and Yin Yang. 3dgs2: Near second-order converging 3d gaussian splatting. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–10, 2025.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian splatting for static and dynamic radiance fields. *arXiv preprint arXiv:2408.03822*, 2024.

Rong Liu, Rui Xu, Yue Hu, Meida Chen, and Andrew Feng. Atomgs: Atomizing gaussian splatting for high-fidelity radiance field. *arXiv preprint arXiv:2405.12369*, 2024.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20654–20664, 2024.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Eduardo Fernandes Montesuma, Fred Maurice Ngole Mboula, and Antoine Souloumiac. Recent advances in optimal transport for machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6): 47–60, 1996.

Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. In *2025 International Conference on 3D Vision (3DV)*, pp. 134–144. IEEE, 2025.

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 (TOG)*, 43(4):1–17, 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. *arXiv preprint arXiv:2403.17898*, 2024.

Shiwei Ren, Tianci Wen, Yongchun Fang, and Biao Lu. Fastgs: Training 3d gaussian splatting in 100 seconds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26094–26103, 2026.

Ludger Rüschendorf. The wasserstein distance and approximation theorems. *Probability Theory and Related Fields*, 70(1):117–129, 1985.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Matthew Thorpe. Introduction to optimal transport. *Notes of Course at University of Cambridge*, 3, 2018.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2008.

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. *arXiv preprint arXiv:2506.09534*, 2025.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3d gaussian splatting. *Computational Visual Media*, 10(4):613–642, 2024.

Nanjie Yao, Hao Wang, Chong Cheng, Zhikang Chen, Wenzhe Li, Jiafei Lyu, Li Shen, Peilin Zhao, Zongqing Lu, Gao Huang, et al. World models for embodied intelligence: From plausible to controllable to actionable. *arXiv preprint arXiv:2609.16697*, 2026.

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 conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. Lp-3dgs: Learning to prune 3d gaussian splatting. *Advances in Neural Information Processing Systems*, 37:122434–122457, 2024.

Beizhen Zhao, Boyi Fu, Sicheng Yu, Zijian Wang, Kaiyong Zhao, Pengcheng Wu, Hao Wang, Steven Hoi, and Hui Xiong. 3d gaussian splatting rasterization: A survey. 2026.

A Proofs and diagnostic examples

A.1 Proof of Proposition 1

Let $\boldsymbol{A}(\boldsymbol{w}) = \lambda_B \boldsymbol{I} + \sum_j w_j \widetilde{\boldsymbol{z}}_j \widetilde{\boldsymbol{z}}_j^\top$. Using $d \log\det \boldsymbol{A} = \operatorname{tr}(\boldsymbol{A}^{-1} d\boldsymbol{A})$,

$$\frac{\partial F_B}{\partial w_i} = \operatorname{tr}\big(\boldsymbol{A}^{-1} \widetilde{\boldsymbol{z}}_i \widetilde{\boldsymbol{z}}_i^\top\big) \tag{16}$$

$$= \widetilde{\boldsymbol{z}}_i^\top \boldsymbol{A}^{-1} \widetilde{\boldsymbol{z}}_i. \tag{17}$$

At $\boldsymbol{w} = \mathbf{1}$, $\boldsymbol{A} = \boldsymbol{G}_B + \lambda_B \boldsymbol{I}$, which gives the first statement of Proposition 1.

A.2 Finite removal gain

The derivative in Proposition 1 is also tied to the exact discrete loss from removing a row. This matters because reduction ultimately makes a finite, rather than infinitesimal, decision.

Write $\boldsymbol{A}_{-i} = \boldsymbol{A} - \widetilde{\boldsymbol{z}}_i \widetilde{\boldsymbol{z}}_i^\top$ and $q_i = \widetilde{\boldsymbol{z}}_i^\top \boldsymbol{A}_{-i}^{-1} \widetilde{\boldsymbol{z}}_i \geq 0$. The matrix determinant lemma gives $\det \boldsymbol{A} / \det \boldsymbol{A}_{-i} = 1 + q_i$. Sherman–Morrison gives $\ell_i = q_i/(1+q_i)$, hence $1 + q_i = (1-\ell_i)^{-1}$. Taking logarithms proves the claim and also shows $\ell_i < 1$.

Thus ranking by ℓ_i is equivalent to ranking by the exact one-row coverage loss when all other block responses are held fixed, proving the second statement.

A.3 Substitute monotonicity and block credit

For positive-definite matrices, inversion reverses the Loewner order. Since $\boldsymbol{G}_B + \boldsymbol{H} + \lambda_B \boldsymbol{I} \succeq \boldsymbol{G}_B + \lambda_B \boldsymbol{I}$, their inverses satisfy $(\boldsymbol{G}_B + \boldsymbol{H} + \lambda_B \boldsymbol{I})^{-1} \preceq (\boldsymbol{G}_B + \lambda_B \boldsymbol{I})^{-1}$. Taking the quadratic form with $\widetilde{\boldsymbol{z}}_i$ gives Eq. (11).

This proves the third statement. The block-credit identity in Eq. (18) follows by cyclicity of trace: $\sum_i \ell_i = \operatorname{tr}[\boldsymbol{Z}_B (\boldsymbol{G}_B + \lambda_B \boldsymbol{I})^{-1} \boldsymbol{Z}_B^\top] = \operatorname{tr}[\boldsymbol{G}_B (\boldsymbol{G}_B + \lambda_B \boldsymbol{I})^{-1}]$, followed by diagonalizing $\boldsymbol{G}_B$. Recomputing feature scales, masses, the trace-dependent ridge, or the later blockwise max-normalization defines a new scoring problem and is not part of the fixed-system monotonicity statement.

A.4 Block credit and response examples

The block-wide credit has an effective-dimension identity. If $\sigma_1, \ldots, \sigma_d$ are the eigenvalues of $\boldsymbol{G}_B$, then

$$\sum_{i \in B} \ell_i = \operatorname{tr}\big[\boldsymbol{G}_B (\boldsymbol{G}_B + \lambda_B \boldsymbol{I})^{-1}\big] = \sum_{r=1}^{d} \frac{\sigma_r}{\sigma_r + \lambda_B}. \tag{18}$$

At fixed response scaling and ridge, repeated rows redistribute the credit of an existing response direction, whereas a well-supported direction outside the current span increases the block's effective dimension. For k equal unit responses, each row receives $1/(k + \lambda_B)$; an orthogonal unit response receives $1/(1 + \lambda_B)$. Stage-I demand enters through e_i^v and m_i, so algebraic novelty is weighted by whether the response is useful to reconstruction.

Table 6: Equal marginal evidence can require different compression decisions. Here $\boldsymbol{u}^\top \boldsymbol{v} = 0$ and both vectors have unit norm.

Local responses	Row norms	Conditional credits	Decision
$\{\boldsymbol{u}, \boldsymbol{u}\}$	$1, 1$	$\frac{1}{2+\lambda_B}, \frac{1}{2+\lambda_B}$	Share one mode
$\{\boldsymbol{u}, \boldsymbol{v}\}$	$1, 1$	$\frac{1}{1+\lambda_B}, \frac{1}{1+\lambda_B}$	Preserve both

The joint Gram matrix captures more than pairwise similarity. A response may be only moderately similar to each neighbor while still lying in their combined span. Its leverage is reduced when the other rows supply sufficient energy along that direction; span membership alone does not imply low

credit. Eq. (8) directly answers how much each existing primitive contributes to the block's covered response span. That one-to-one correspondence makes the score usable as a budget signal: every source Gaussian receives credit, and the reducer can allocate targets without a separate mode-to-representative decision.

A.5 View-tagged sketch in expectation

Let $\boldsymbol{y}_i^v = \boldsymbol{D}_v^{-1}\boldsymbol{x}_i^v$ denote a normalized per-view response and define the pre-normalization sketch $\boldsymbol{s}_i = V^{-1/2}\sum_v \boldsymbol{P}_v^\top \boldsymbol{y}_i^v$.

Proposition 2 (No cross-view similarity in expectation)**.** *If the entries of each $\boldsymbol{P}_v \in \mathbb{R}^{10\times d}$ are independent $\mathcal{N}(0, 1/d)$ variables and the matrices are independent across views, then*

$$\mathbb{E}[\boldsymbol{s}_i^\top \boldsymbol{s}_j] = \frac{1}{V}\sum_{v=1}^{V}(\boldsymbol{y}_i^v)^\top \boldsymbol{y}_j^v. \tag{19}$$

Proof. Expanding the inner product produces terms $(\boldsymbol{y}_i^v)^\top \boldsymbol{P}_v \boldsymbol{P}_u^\top \boldsymbol{y}_j^u$. For $u \neq v$, independence and zero mean give $\mathbb{E}[\boldsymbol{P}_v \boldsymbol{P}_u^\top] = \boldsymbol{0}$. For $u = v$, the chosen variance gives $\mathbb{E}[\boldsymbol{P}_v \boldsymbol{P}_v^\top] = \boldsymbol{I}_{10}$. Summing the surviving terms yields the result. □

The independent camera tags therefore approximate the inner product of a conceptual view-concatenated descriptor without storing its $10V$ coordinates. A shared projection after view averaging would retain cross-view products and would not have this property.

A.6 Why marginal scalarization is insufficient

Theorem 1 (Marginal-score ambiguity)**.** *Let a marginal rule score each response using only its Euclidean norm and coordinate sum. There exist two two-primitive systems for which every primitive receives the same score, but whose response ranks are different.*

Proof. Consider

$$\boldsymbol{Z}_{\mathrm{dup}} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \qquad \boldsymbol{Z}_{\mathrm{comp}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Every row in both systems has norm one and coordinate sum one, so any rule restricted to those marginal statistics produces the same two scores. However, $\mathrm{rank}(\boldsymbol{Z}_{\mathrm{dup}}) = 1$, whereas $\mathrm{rank}(\boldsymbol{Z}_{\mathrm{comp}}) = 2$. Thus the systems require different preservation decisions despite identical marginal statistics. □

This theorem targets the common scalarization pattern, not every conceivable function of a single response vector. The broader point is that redundancy is a property of a *set*: even a rich single-row descriptor cannot know whether another row duplicates it without a joint comparison.

A.7 Duplicate and complementary responses

Proposition 3. *Let the ridge $\lambda_B > 0$ be fixed. If a block contains k copies of a unit response $\boldsymbol{u}$, the leverage of each copy is $1/(k + \lambda_B)$. If it also contains a unit response $\boldsymbol{v}$ orthogonal to $\boldsymbol{u}$, then the leverage of $\boldsymbol{v}$ is $1/(1 + \lambda_B)$.*

Proof. On the span of $\boldsymbol{u}$, the Gram matrix is $k\boldsymbol{u}\boldsymbol{u}^\top$; hence

$$\boldsymbol{u}^\top(k\boldsymbol{u}\boldsymbol{u}^\top + \lambda_B \boldsymbol{I})^{-1}\boldsymbol{u} = \frac{1}{k + \lambda_B}.$$

Adding an orthogonal $\boldsymbol{v}$ makes the Gram matrix diagonal in the orthonormal basis $\{\boldsymbol{u}, \boldsymbol{v}\}$, with eigenvalues k and 1. The quadratic form for $\boldsymbol{v}$ is therefore $1/(1 + \lambda_B)$. □

At fixed λ_B, as k grows, total credit along the repeated direction approaches one, rather than growing linearly with the number of copies. This limit does not assume recomputation of the trace-dependent ridge. This is the desired diminishing-return behavior.

A.8 EFFECTIVE-RANK DIAGNOSTIC

For analysis only, each block reports

$$r_{\text{eff}}(\boldsymbol{G}_B) = \frac{\operatorname{tr}(\boldsymbol{G}_B)^2}{\operatorname{tr}(\boldsymbol{G}_B^2)}. \tag{20}$$

For eigenvalues $\{\sigma_j\}$, this equals $(\sum_j \sigma_j)^2 / \sum_j \sigma_j^2$, lies in $[1, \operatorname{rank}(\boldsymbol{G}_B)]$ for a nonzero positive-semidefinite Gram matrix, and is largest when active directions carry equal energy. For the zero Gram we report zero. This participation ratio is different from the ridge effective dimension $\sum_r \sigma_r/(\sigma_r + \lambda_B)$ in Eq. (18); it is used only for diagnostics, not to allocate targets.

Fig. 6 compares marginal and conditional reduction on two bicycle views before recovery training. Relative to the pre-compression render, mean PSNR increases from 26.94 to 27.45 dB and mean L1 decreases from 0.02560 to 0.02313, corresponding to a 0.51 dB gain and a 9.65% error reduction. Both views improve.



Figure 6: Instantaneous compression on two bicycle views. Pre-compression render, marginal reduction, conditional response and their residuals are shown before recovery training. Residual scales are shared within each view.

B IMPLEMENTATION DETAILS

B.1 DEFAULT CONFIGURATION

Tab. 7 lists the default Balanced settings. Turbo trains for 18,000 iterations and stops densification at iteration 8,000. Balanced and Quality each train for 30,000 iterations. Quality uses $\tau = 0.075$ and stops densification and pruning at iteration 18,000. Its command-line parameter `highfeature_lr` is 0.04; the optimizer divides this value by 20, giving a learning rate of 0.002 for the higher-order SH coefficients.

Table 7: Default EchoGS hyperparameters.

Parameter	Value	Parameter	Value
Response dimension d	24	Conditional mix β	0.25
Score KD depth	10	Reducer KD depth	10
Ridge multiplier λ	10^{-3}	Mass safeguard η	0.05
Mass floor κ	0.25	Error threshold τ	0.1
Sampled cameras V	10	Default target fraction	0.7
Projection seed	42	Split-mass floor ϵ_m	0.1
Reducer tolerance	10^{-5}	Reducer iterations	100

B.2 RESPONSE CONSTRUCTION

Let $\ell(\boldsymbol{I})$ denote luminance, $\boldsymbol{K}_x, \boldsymbol{K}_y$ the Sobel filters, and $\boldsymbol{K}_{\mathrm{Lap}}$ the discrete Laplacian. The implemented image-structure response is

$$\boldsymbol{S}(\boldsymbol{I}) = \mathcal{N}\left(\sqrt{(\boldsymbol{K}_x * \ell(\boldsymbol{I}))^2 + (\boldsymbol{K}_y * \ell(\boldsymbol{I}))^2 + \epsilon} + 0.5\,|\boldsymbol{K}_{\mathrm{Lap}} * \ell(\boldsymbol{I})|\right). \tag{21}$$

The structural residual is the min–max-normalized absolute difference $E^v_{\mathrm{str}} = \mathcal{N}(|\boldsymbol{S}(\hat{\boldsymbol{I}}_v) - \boldsymbol{S}(\boldsymbol{I}_v)|)$. This combines edge displacement and local curvature mismatch without treating the filter response itself as a redundancy measure.

The structural mask is

$$\boldsymbol{M}_v = \mathbf{1}[E^v_{\mathrm{str}} > 0.5\tau] \odot \mathbf{1}\left[E^v_{\mathrm{rgb}} > \tau\right], \qquad \tau = 0.1.$$

Both RGB and structural residual maps are min–max-normalized over pixels in each view. Importance counts are summed over views and floor-averaged. Ten cameras are sampled without replacement. In the reducer, position and covariance follow Eq. (15); appearance and opacity are inherited from the representative source selected for the target.

B.3 CONDITIONAL SCORING ALGORITHM

Algorithm 1 makes the data flow through Stages I and II explicit. The view loop accumulates response evidence without constructing a $10V$-dimensional descriptor; the block loop then computes conditional credit only among spatially plausible substitutes.

Numerical inverse. The implementation forms the ridge matrix in float32 and calls `torch.linalg.pinv` with its default SVD-based settings, without explicit tolerance overrides. For the $d \times d$ matrix, these defaults are $\mathrm{atol} = 0$ and $\mathrm{rtol} = d\epsilon_{\mathrm{fp32}}$, where $\epsilon_{\mathrm{fp32}} = 2^{-23}$; at $d = 24$, $\mathrm{rtol} \approx 2.86 \times 10^{-6}$. Singular values below the largest singular value times this relative tolerance are discarded. Proposition 1 concerns the exact ridge inverse at fixed responses and ridge; finite-precision pseudoinversion need not satisfy its identities exactly.

Algorithm 1 View-tagged conditional renderer-response scoring

Require: Gaussians $\mathcal{G}$, cameras $\mathcal{V}$, dimensions d, ridge λ, blend β
Ensure: allocation scores $\{s_i\}$ and diagnostics

```
Initialize per-primitive demand, support, footprint, and sketch accumulators
for each camera v ∈ V do
    Render Î_v, compute M_v by Eq. (1), and run flagged/importance rasterization
    Form all x_i^v, compute D_v across all Gaussians, and accumulate e_i^v and contribution statistics
    for each primitive i do
        Form x_i^v by Eq. (3) and accumulate P_v^⊤ D_v^{-1} x_i^v / √V
    end for
end for
RMS-normalize sketches to obtain z_i; compute b_i and split masses m_i
Build the importance-balanced score KD-tree
for each leaf block B do
    RMS-normalize response coordinates within B and form z̃_i
    Compute G_B, λ_B, and ℓ_i using Eq. (8)
    Max-normalize leverage and log mass within B; store ℓ̄_i + η p̄_i
end for
Assemble all block scores; scene-wise normalize to r_i and blend to s_i by Eq. (12)
return {s_i}, effective-rank statistics, and score summaries
```

B.4 BUDGET-REALIZATION ALGORITHM

Algorithm 2 describes Stage III. Initially, $K_B^{(0)} = \max\{1, \lfloor KW_B / \sum_{B'} W_{B'} \rfloor\}$; the largest-mass leaf receives the discrepancy $K - \sum_B K_B^{(0)}$. All corrected budgets must remain positive. Assignment

uses normalized opacity, not allocation mass: $\gamma_{ik} = \pi_i \mathbf{1}[k \in T_i]/|T_i|$, with T_i the nearest-target set. The monitored objective is $J_B = \sum_{i \in B,k} \gamma_{ik} d^2_{\text{base}}(i,k)$. Moment weights normalize each nonempty column of γ; caps and empty-target removal may reduce the realized count below K.

Algorithm 2 Evidence-balanced geometric mixture reduction

Require: Gaussians $\mathcal{G}$, scores $\{s_i\}$, contributions $\{c_i\}$, target budget K
Ensure: reduced Gaussian set $\widetilde{\mathcal{G}}$

```
Compute allocation masses w_i^alloc by Eq. (13) and build the reducer KD-tree
Allocate K_B proportional to W_B and correct the largest-mass leaf; require all corrected K_B ≥ 1
for each leaf block B do
    Initialize min(K_B, |B|) targets from a seeded subset; set J_old = +∞
    repeat
        Compute all source–target costs using Eq. (14)
        Assign each source to every minimum-cost target; divide opacity equally across ties
        Evaluate assignment objective J_B; remove targets with incoming mass ≤ 10^-8
        if |J_old − J_B| < 10^-5 then
            break
        end if
        Set J_old = J_B
        Update target means and covariances by Eq. (15)
        Recompute covariance square roots and objective J_B
        Compute Δ = |J_old − J_B| and set J_old = J_B
    until Δ < 10^-5 or 100 iterations have completed
    for each surviving target k do
        Copy spherical harmonics and stored opacity from the closest source under the updated geometric cost
    end for
end for
Rebuild the optimizer on G̃ and continue training from iteration 15,000
return G̃
```

C GEOMETRIC MOMENT UPDATE

For completeness, let cluster C_k define the normalized mixture

$$p_k(\boldsymbol{x}) = \sum_{i \in C_k} \omega_{ik} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i). \tag{22}$$

Let I be its component index, with $\Pr(I = i) = \omega_{ik}$. Its expectation is $\mathbb{E}_{p_k}[\boldsymbol{x}] = \sum_i \omega_{ik} \boldsymbol{\mu}_i = \widetilde{\boldsymbol{\mu}}_k$. Using the law of total covariance,

$$\begin{aligned} \text{Cov}_{p_k}[\boldsymbol{x}] &= \mathbb{E}_I[\text{Cov}(\boldsymbol{x} \mid I)] + \text{Cov}_I(\mathbb{E}[\boldsymbol{x} \mid I]) && (23) \\ &= \sum_i \omega_{ik} \boldsymbol{\Sigma}_i + \sum_i \omega_{ik} (\boldsymbol{\mu}_i - \widetilde{\boldsymbol{\mu}}_k)(\boldsymbol{\mu}_i - \widetilde{\boldsymbol{\mu}}_k)^\top, && (24) \end{aligned}$$

which is Eq. (15). Therefore, in exact arithmetic, a single aggregated Gaussian matches the first two moments of its assigned normalized mixture. The update applies to geometry; appearance and opacity use the representative described in Appendix B.

C.1 MOMENT MATCHING AS A GAUSSIAN PROJECTION

Proposition 4 (Forward-KL Gaussian projection)**.** *Let p_k be the normalized cluster mixture above with finite second moments and positive-definite covariance. Among all nonsingular Gaussian densities $q = \mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$, the unique minimizer of $\text{KL}(p_k \| q)$ is $q^* = \mathcal{N}(\widetilde{\boldsymbol{\mu}}_k, \widetilde{\boldsymbol{\Sigma}}_k)$.*

Proof. The entropy of p_k is constant with respect to q, so minimizing $\text{KL}(p_k \| q)$ is equivalent to minimizing the Gaussian cross-entropy

$$\frac{1}{2} \log \det \boldsymbol{C} + \frac{1}{2} \mathbb{E}_{p_k}[(\boldsymbol{x} - \boldsymbol{m})^\top \boldsymbol{C}^{-1} (\boldsymbol{x} - \boldsymbol{m})].$$

For fixed $\boldsymbol{C}$, the quadratic term is minimized at $\boldsymbol{m} = \mathbb{E}_{p_k}[\boldsymbol{x}]$. Substitution leaves $\frac{1}{2}[\log\det \boldsymbol{C} + \operatorname{tr}(\boldsymbol{C}^{-1}\operatorname{Cov}_{p_k}[\boldsymbol{x}])]$, whose stationary point and global minimum is $\boldsymbol{C} = \operatorname{Cov}_{p_k}[\boldsymbol{x}]$. To see uniqueness, let $\boldsymbol{S} = \operatorname{Cov}_{p_k}[\boldsymbol{x}]$ and let $a_j > 0$ be the eigenvalues of $\boldsymbol{S}^{-1/2}\boldsymbol{C}\boldsymbol{S}^{-1/2}$. The excess objective is $\frac{1}{2}\sum_j(\log a_j + a_j^{-1} - 1) \geq 0$, with equality only at $a_j = 1$ for every j. The minimizer is therefore exactly Eq. (15). □

The update can therefore be read as the information projection of each assigned source mixture onto the Gaussian family. It is not guaranteed to decrease the square-root assignment objective. It is also the support-update step of a hard-assignment mixture-reduction loop, analogous in structure to classical alternating mixture fitting (Moon, 1996). The assignment matrix defines a sparse discrete coupling; unlike a global optimal-transport solver (Rüschendorf, 1985; Villani et al., 2008; Thorpe, 2018; Montesuma et al., 2024), it sends each source mass only to its nearest current target(s), splitting exact ties equally.

C.2 Relation of the square-root cost to Bures geometry

For positive-semidefinite covariances $\boldsymbol{A}$ and $\boldsymbol{B}$, the Bures covariance term has the Procrustes form (Gelbrich, 1990; Bhatia et al., 2019)

$$d_{\mathrm{B}}^2(\boldsymbol{A}, \boldsymbol{B}) = \min_{\boldsymbol{Q}^\top\boldsymbol{Q}=\boldsymbol{I}} \|\boldsymbol{A}^{1/2} - \boldsymbol{B}^{1/2}\boldsymbol{Q}\|_F^2 \leq \|\boldsymbol{A}^{1/2} - \boldsymbol{B}^{1/2}\|_F^2. \tag{25}$$

Thus Eq. (14) uses the identity-alignment upper bound to the Bures term; it is exact when the covariances commute and share an eigenbasis.

D Complexity

Let N be the number of source Gaussians, V the number of selected cameras, d the sketch dimension, D the KD-tree depth, and $M \leq 2^D$ the number of nonempty score blocks. Response accumulation uses rasterization used for marginal demand and renderer-contribution scoring plus $O(NVd)$ projection arithmetic and $O(Nd)$ storage. The weighted KD-tree costs $O(DN\log N)$ under the implementation's per-level sorting. For block sizes n_b, Gram construction costs $O(\sum_b n_b d^2) = O(Nd^2)$, and the M small inversions cost $O(Md^3)$. Evaluating all quadratic-form credits costs another $O(Nd^2)$; the final rowwise dot products cost $O(Nd)$. The scoring arithmetic is therefore $O(Nd^2 + Md^3)$, in addition to sketch construction and partitioning.

Let $g = 12$ denote its geometric feature dimension (three center coordinates and nine covariance-square-root entries). For reducer leaf sizes n_B and initialized target counts k_B, one assignment/update iteration costs $O(\sum_B n_B k_B g)$ and stores blockwise cost/coupling matrices. Only if the M_{red} reducer leaves have approximately equal source and target counts does this simplify to $O(NKg/M_{\mathrm{red}})$.

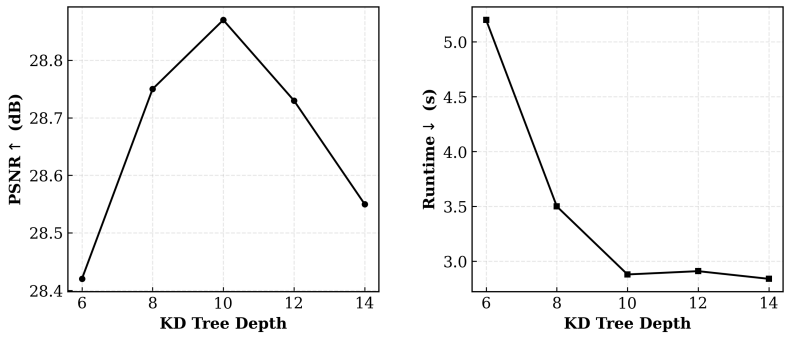

Figure 7: Evidence-balanced KD-tree depth on Mip-NeRF 360 for the $\beta = 0$ reducer. Depth ten gives the highest plotted PSNR with near-minimal runtime; deeper partitions reduce PSNR with little runtime benefit. This sweep evaluates reducer locality and does not isolate conditional scoring.

E ADDITIONAL EXPERIMENTAL RESULTS

This section provides per-scene results, runtime, memory, rendering rates, and supporting diagnostics for the operating presets in the main tables. Turbo is evaluated on Mip-NeRF 360, Tanks & Temples, and Deep Blending; Waymo reports Balanced and Quality.

Aggregation. Per-scene entries and dataset summaries use the same scene lists and arithmetic means. Compression ratios divide dataset-mean counts. Display rounding is applied only after aggregation.

E.1 MIXING-WEIGHT SENSITIVITY

Tab. 8 compares settings of the allocation mixing weight. Its default row is shared with the full-method row in Tab. 5 and the Balanced benchmark entry.

Table 8: Mixing-weight sensitivity on Mip-NeRF 360 for Ours–Balanced. Time is in minutes and counts are in millions.

β	Time↓	N_{GS} ↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
0.00	**3.94**	0.29M	28.87	0.862	0.217	330
0.10	3.96	0.27M	**28.90**	0.862	0.217	336
0.25	3.98	0.26M	28.88	0.862	**0.214**	341
0.50	3.98	0.26M	28.87	**0.863**	0.217	342
0.75	3.96	**0.25M**	28.85	0.860	0.218	346
1.00	3.99	**0.25M**	28.79	0.858	0.222	**349**

E.2 RESPONSE-DIMENSION SENSITIVITY

Tab. 9 varies the sketch dimension d in Tab. 7, retaining the Balanced schedule and the remaining default settings. Smaller sketches reduce scoring work but compress more response directions into the same coordinates. Increasing d from 8 to 24 improves the quality–efficiency balance in the table; larger sketches give little further quality change while increasing training time. The default $d = 24$ balances response resolution and computational cost.

Table 9: Response-dimension sensitivity on Mip-NeRF 360 for Ours–Balanced. Time is in minutes and counts are in millions. The shaded row uses the default setting.

d	Time↓	N_{GS} ↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
8	**3.94**	0.28M	28.77	0.858	0.222	329
16	3.96	0.27M	28.84	0.860	0.217	336
24	3.98	**0.26M**	28.88	0.862	0.214	**341**
32	4.03	**0.26M**	**28.90**	**0.863**	**0.213**	340
48	4.14	**0.26M**	28.89	0.862	0.214	339

E.3 ERROR-THRESHOLD SENSITIVITY

Tab. 10 changes the error threshold τ while keeping the structural threshold coupled at $\tau/2$ as in Eq. (1); all other settings follow Balanced. A lower threshold activates more residual pixels and can spread demand over weak discrepancies, whereas a high threshold can miss subtle reconstruction errors. The default $\tau = 0.1$ gives the lowest LPIPS in the table; higher thresholds favor smaller counts and faster rendering at the expense of fidelity.

Table 10: Error-threshold sensitivity on Mip-NeRF 360 for Ours–Balanced. Time is in minutes and counts are in millions. The shaded row uses the default setting.

τ	Time↓	N_{GS} ↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
0.050	4.07	0.29M	28.85	0.862	0.219	326
0.075	4.02	0.27M	**28.89**	**0.863**	0.216	335
0.100	3.98	0.26M	28.88	0.862	**0.214**	341
0.150	3.94	0.25M	28.80	0.859	0.219	347
0.200	**3.91**	**0.24M**	28.68	0.855	0.228	**354**

E.4 Mass-floor sensitivity

Tab. 11 varies the additive mass floor κ in the response weight $\sqrt{\kappa + \rho_i}$, leaving the remaining Balanced settings fixed. A positive floor moderates the suppression of low-mass responses, although it cannot create a response when the original signature is zero. An excessively large floor weakens the relative influence of renderer mass. The default $\kappa = 0.25$ gives the best PSNR and LPIPS in the table, balancing mass sensitivity and the contribution of weaker responses.

Table 11: Mass-floor sensitivity on Mip-NeRF 360 for Ours–Balanced. Time is in minutes and counts are in millions. The shaded row uses the default setting.

κ	Time↓	N_{GS} ↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
0.00	**3.97**	**0.25M**	28.79	0.858	0.222	**347**
0.10	3.98	0.26M	28.85	0.861	0.217	343
0.25	3.98	0.26M	**28.88**	**0.862**	**0.214**	341
0.50	4.00	0.27M	28.87	**0.862**	0.216	336
1.00	4.03	0.28M	28.81	0.859	0.221	329

E.5 Runtime, memory, and rendering rate

Tab. 12 reports total training time alongside Stage I and the combined Stage II–III timing. Tab. 13 and 14 report scene-mean peak memory and rendering FPS for the corresponding presets.

Table 12: Per-scene runtime for Turbo and Balanced on Mip-NeRF 360. Total training is in minutes; Stage I, Stage II and Stage III are in seconds.

Module	bicycle	bonsai	counter	garden	kitchen	room	stump
Ours–Turbo / Total (min)	**2.52**	**2.50**	**2.63**	**2.37**	**2.92**	**2.31**	**2.19**
Ours–Turbo / Stage I (s)	**6.47**	**7.46**	**7.54**	**6.66**	**7.58**	**7.42**	**6.21**
Ours–Turbo / Stage II & Stage III (s)	**16.01**	10.26	10.08	12.70	11.22	9.90	15.23
Ours–Balanced / Total (min)	4.04	3.82	4.41	3.74	4.51	3.82	3.53
Ours–Balanced / Stage I (s)	10.73	11.93	13.09	10.86	13.04	11.84	10.45
Ours–Balanced / Stage II & Stage III (s)	16.06	**9.59**	**9.20**	**12.56**	**10.30**	**9.19**	**14.56**

Table 13: Average per-scene peak GPU memory (GB). – denotes an unevaluated preset.

Method	Mip-NeRF 360	Deep Blending	Tanks & Temples	Waymo
3DGS	9.41	8.26	4.79	21.28
DashGaussian	9.01	7.74	4.48	21.19
Speedy-Splat	6.71	5.25	2.77	14.56
GHAP	9.05	8.11	4.74	19.72
Mini-Splatting	7.22	6.27	4.65	20.53
Taming 3DGS	5.23	3.93	2.49	14.59
FastGS	**4.71**	**3.88**	**2.42**	15.68
Ours–Turbo	5.73	4.38	2.55	–
Ours–Balanced	6.03	4.58	2.67	**14.28**

Table 14: Mean rendering speed (FPS) on the three standard-scale datasets.

Method	Mip-NeRF 360	Tanks & Temples	Deep Blending
3DGS	86	115	87
LightGaussian	165	217	174
Compact-3DGS	152	197	193
RadSplat	176	225	210
DashGaussian	106	148	106
Speedy-Splat	191	186	235
GHAP	316	354	333
Mini-Splatting	308	372	344
Taming 3DGS	160	231	226
FastGS	<u>327</u>	**389**	<u>359</u>
Ours–Turbo	305	251	320
Ours–Balanced	**341**	<u>378</u>	**391**
Ours–Quality	292	337	299

E.6 Reducer partition diagnostics

Reducer locality. Fig. 7 varies the evidence-balanced KD-tree depth. Shallow partitions increase runtime and reduce fidelity; depth ten gives the highest plotted PSNR, while further subdivision offers little runtime benefit and lowers PSNR. Tab. 15 compares partition strategies: spatially compatible, evidence-balanced blocks achieve higher PSNR and lower LPIPS than random grouping or median KD leaves.

Table 15: Spatial partitioning ablation on Mip-NeRF 360. Time is partition time in seconds, not total training time; counts are in millions.

Partition	Time↓	$N_{\mathbf{GS}}$ ↓	PSNR↑	SSIM↑	LPIPS↓
Random	**0.52**	0.33M	26.15	0.793	0.314
Median KD-tree	<u>3.95</u>	<u>0.28M</u>	<u>28.56</u>	<u>0.856</u>	<u>0.243</u>
Ours–Balanced (evidence-balanced KD-tree)	4.06	**0.26M**	**28.88**	**0.862**	**0.214**

E.7 Per-scene Mip-NeRF 360 results

The per-scene tables report the same presets and scene set as Tab. 3. The count table exposes how each preset distributes capacity among indoor and outdoor scenes, while the PSNR table provides the corresponding reconstruction values.

Table 16: Per-scene PSNR on Mip-NeRF 360.

Method	bicycle	bonsai	counter	garden	kitchen	room	stump
3DGS	25.22	**32.23**	**29.06**	27.37	31.36	31.59	26.61
DashGaussian	25.47	31.76	<u>28.95</u>	**27.56**	**31.45**	31.60	27.08
Speedy-Splat	25.57	31.59	28.69	<u>27.50</u>	30.56	<u>31.97</u>	26.36
GHAP	25.37	30.97	28.38	27.15	30.44	30.81	26.53
Mini-Splatting	<u>25.66</u>	31.41	28.59	27.24	31.34	31.41	**27.50**
Taming 3DGS	24.82	<u>31.84</u>	28.55	27.27	31.03	31.29	26.01
FastGS	25.47	31.50	28.59	27.12	31.18	31.67	26.87
Ours–Turbo	24.83	30.36	27.76	26.32	30.27	31.01	26.42
Ours–Balanced	25.46	31.31	28.60	27.12	31.15	31.66	26.85
Ours–Quality	**25.70**	31.52	28.90	27.45	<u>31.38</u>	**32.01**	<u>27.12</u>

Table 17: Per-scene Gaussian count (millions) on Mip-NeRF 360.

Method	bicycle	bonsai	counter	garden	kitchen	room	stump
3DGS	4.63	1.00	1.01	3.85	1.49	1.22	4.08
DashGaussian	4.69	0.86	0.84	3.23	1.22	1.17	3.52
Speedy-Splat	1.14	0.25	0.19	0.90	0.22	0.21	0.73
GHAP	0.93	0.21	0.21	0.72	0.31	0.25	0.81
Mini-Splatting	0.53	0.35	0.41	0.59	0.43	0.39	0.60
Taming 3DGS	0.81	0.41	0.31	1.90	0.48	0.23	0.48
FastGS	0.86	0.25	0.20	0.51	0.28	0.21	0.62
Ours–Turbo	**0.44**	**0.16**	**0.14**	**0.30**	**0.18**	**0.13**	**0.33**
Ours–Balanced	0.51	**0.16**	**0.14**	0.32	0.19	0.14	0.36
Ours–Quality	0.98	0.37	0.32	0.91	0.46	0.33	0.61

E.8 Per-scene Waymo results

The Waymo tables list the six scenes underlying Tab. 4.

Table 18: Per-scene PSNR on Waymo.

Method	100613	117240	130854	148697	152706	158686
3DGS	**31.64**	31.40	24.62	24.64	21.40	27.69
DashGaussian	28.29	30.59	24.36	23.20	20.10	27.25
Speedy-Splat	28.02	30.35	23.70	22.58	19.48	26.72
GHAP	30.60	30.52	24.39	24.29	21.52	27.21
Mini-Splatting	25.95	26.32	21.18	20.07	20.54	26.44
Taming 3DGS	31.07	30.38	24.75	24.58	**22.50**	27.39
FastGS	31.39	31.36	24.32	24.82	21.91	27.48
Ours–Balanced	31.19	31.25	24.52	24.97	22.09	27.47
Ours–Quality	31.52	**31.52**	**24.87**	**25.28**	22.38	**27.78**

Table 19: Per-scene Gaussian count (millions) on Waymo.

Method	100613	117240	130854	148697	152706	158686
3DGS	2.52	1.71	1.36	2.02	0.20	3.27
DashGaussian	1.34	1.36	1.34	1.21	0.16	2.31
Speedy-Splat	0.52	0.46	0.38	0.47	**0.03**	1.03
GHAP	0.49	0.36	0.31	0.43	0.04	0.65
Mini-Splatting	0.31	0.24	0.34	**0.21**	0.27	0.41
Taming 3DGS	0.48	0.29	0.31	0.37	0.13	0.59
FastGS	0.43	0.32	0.43	0.59	0.09	0.65
Ours–Balanced	**0.26**	**0.20**	**0.26**	0.33	0.06	**0.37**
Ours–Quality	0.52	0.45	0.53	0.69	0.14	0.81

E.9 Deep Blending and Tanks & Temples details

The following tables give the four scenes behind the Deep Blending and Tanks & Temples summaries.

Table 20: Per-scene training time (minutes), Gaussian count (millions), and FPS on Deep Blending and Tanks & Temples.

Method	drjohnson			playroom			train			truck		
	Time	$N_{GS}\downarrow$	FPS	Time	$N_{GS}\downarrow$	FPS	Time	$N_{GS}\downarrow$	FPS	Time	$N_{GS}\downarrow$	FPS
3DGS	34.13	3.07	73	26.23	1.85	101	16.05	1.09	107	19.55	2.05	123
DashGaussian	8.03	2.35	110	6.33	1.53	102	7.34	0.98	137	7.11	1.44	159
Speedy-Splat	20.48	0.60	232	16.72	0.36	238	12.47	0.17	129	14.54	0.25	243
GHAP	22.90	0.61	323	17.62	0.37	343	11.12	0.22	347	13.15	0.40	361
Mini-Splatting	23.26	0.61	336	20.42	0.51	352	16.43	0.28	**410**	16.83	0.32	334
Taming 3DGS	4.45	0.40	217	3.66	0.18	235	5.09	0.37	207	4.62	0.27	255
FastGS	3.17	0.39	345	2.80	0.25	373	2.81	0.22	377	2.88	0.28	**401**
Ours–Turbo	**2.00**	**0.22**	346	**1.76**	**0.16**	294	**1.78**	**0.14**	287	**1.88**	**0.18**	216
Ours–Balanced	3.11	0.25	**402**	2.83	0.18	**380**	2.81	0.16	368	2.93	0.19	387
Ours–Quality	4.23	0.52	305	3.83	0.40	293	3.94	0.33	349	4.13	0.36	325

Table 21: Per-scene PSNR, SSIM, and LPIPS on Deep Blending and Tanks & Temples.

Method	drjohnson			playroom			train			truck		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
3DGS	29.40	0.903	0.248	30.09	0.903	0.252	22.17	**0.819**	**0.208**	25.36	0.882	0.153
DashGaussian	29.18	0.900	0.252	30.05	0.904	0.246	22.10	0.813	0.211	25.79	0.882	0.152
Speedy-Splat	29.15	0.902	0.255	30.12	0.908	0.258	21.73	0.773	0.290	25.18	0.868	0.190
GHAP	29.38	0.902	0.253	30.11	0.903	0.257	21.43	0.791	0.253	24.95	0.865	0.181
Mini-Splatting	29.59	0.906	**0.246**	30.44	0.909	**0.240**	21.65	0.810	0.223	25.21	0.879	**0.139**
Taming 3DGS	29.47	0.900	0.270	30.20	0.898	0.279	22.49	0.803	0.238	25.12	0.863	0.186
FastGS	29.45	0.901	0.260	30.63	0.908	0.254	22.43	0.799	0.246	25.69	0.871	0.178
Ours–Turbo	29.31	0.894	0.281	30.55	0.904	0.270	21.97	0.783	0.272	25.27	0.867	0.186
Ours–Balanced	29.70	0.905	0.265	30.70	**0.910**	0.261	22.50	0.802	0.247	25.68	0.871	0.175
Ours–Quality	**29.79**	**0.908**	0.251	**30.74**	0.908	0.249	**22.61**	**0.819**	0.213	**25.81**	**0.886**	0.149

E.10 QUALITATIVE COMPARISONS

Fig. 4 in the main paper gives the selected cross-dataset comparison. The remaining figures are included here.

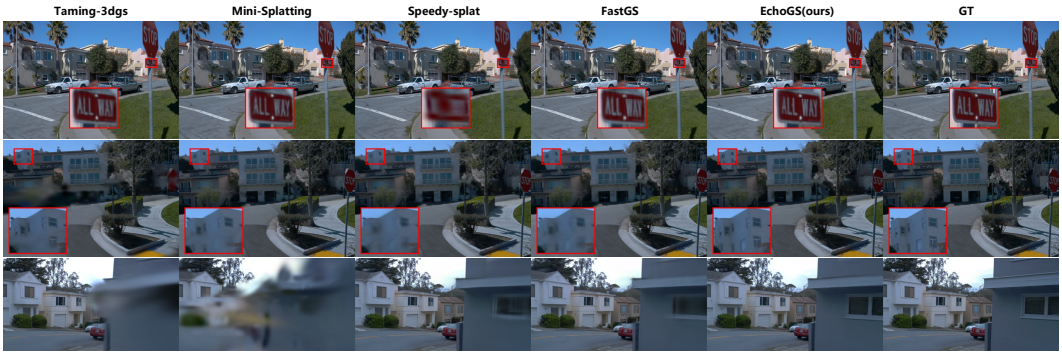

Figure 8: Waymo qualitative comparisons. Red boxes magnify traffic signs, building edges, and thin foreground/background boundaries.

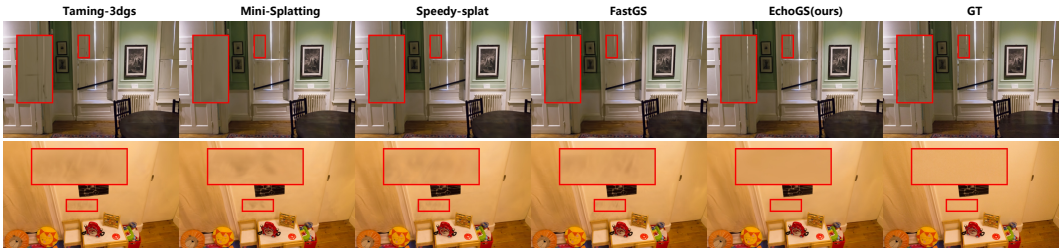

Figure 9: Further indoor and outdoor examples, showing small object markings, thin structures, distant windows, door edges, and wall detail.

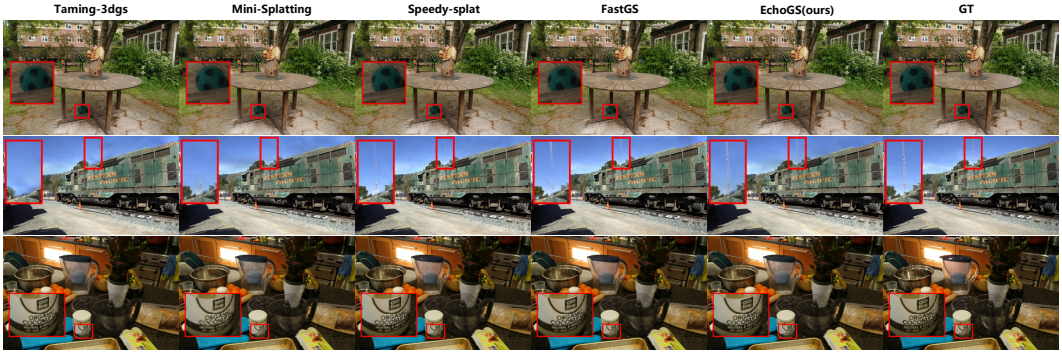

Figure 10: Additional qualitative comparisons, including foliage, thin poles, and small printed labels.

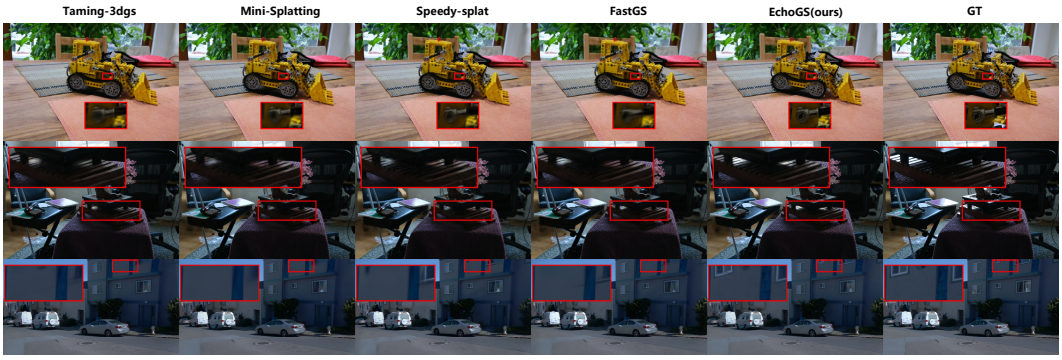

Figure 11: Additional qualitative comparisons, including foliage, thin poles, and small printed labels.