ALLOCSPLAT: SPATIAL ALLOCATION FOR BUDGET-ADAPTIVE FEED-FORWARD GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

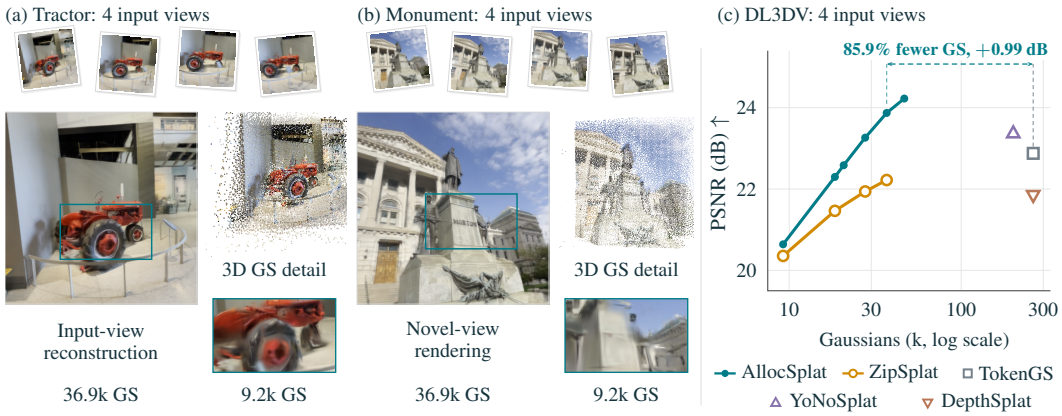

Figure 1: **AllocSplat adapts reconstruction capacity without retraining.** (a,b) Reconstructions with 36.9K Gaussians, their decoded centers in 3D, and corresponding cropped regions reconstructed with 9.2K Gaussians. Panel (a) shows an input view and (b) a held-out view. (c) Reconstruction quality versus Gaussian count on DL3DV-140 with four posed inputs and no test-time optimization. AllocSplat and ZipSplat vary capacity with fixed weights; other methods are shown at their evaluated capacities. All methods are scored on the same 46 non-input views at 252×252.

ABSTRACT

Feed-forward 3D Gaussian Splatting reconstructs renderable scenes from a few images with known camera poses. Different storage and rendering limits call for different numbers of Gaussians. The challenge is to distribute a limited number of Gaussians across the scene while preserving local visual detail. We introduce **AllocSplat**, which uses 3D anchor points to determine where Gaussian groups are placed and how their spatial scales are set. A joint geometry–appearance rule selects these anchors from depth-based 3D candidates, considering both scene coverage and differences in visual content. A density-adaptive local decoder generates a fixed number of Gaussians per anchor, so the anchor count sets the total Gaussian count. It adapts each group's spatial scale to the local anchor density, using distances to nearby selected anchors to limit Gaussian sizes and offsets from the anchor. When the requested budget changes, the model selects the corresponding number of anchors and recomputes local scales and information shared across anchors before decoding. The same anchor point can thus generate different Gaussian groups at different budgets without changing the network weights. Experiments on DL3DV-140 show improved quality–count trade-offs across multiple Gaussian budgets. With four input views, AllocSplat achieves 0.99 dB higher PSNR than TokenGS using 85.9% fewer Gaussians, and exceeds ZipSplat by 0.29–1.65 dB at matched primitive counts.

1 INTRODUCTION

Feed-forward 3D Gaussian Splatting predicts a renderable scene from a few images with known camera poses in a single network pass (Charatan et al., 2024; Chen et al., 2024b; Xu et al., 2025).

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.

The resulting Gaussians support efficient rendering (Kerbl et al., 2023), with their number affecting storage, transmission and rendering costs. Different devices may require different Gaussian budgets for the same scene. A single model should support these budgets while retaining scene coverage and local detail.

Recent feed-forward methods support adjustable Gaussian counts but differ in where the retained Gaussians are placed. ZipSplat clusters visual tokens in feature space and decodes each cluster into Gaussians with unconstrained positions (Veicht et al., 2026). EcoSplat conditions Gaussian attributes on the requested count and keeps the most opaque Gaussians (Park et al., 2026); AdaptiveSplat prunes low-texture clusters and adapts the surviving Gaussians (Singhal et al., 2026). In both, the centers remain on the pixel grid of the dense prediction. Anchor-based decoders, from per-scene Scaffold-GS (Lu et al., 2024) to feed-forward AnchorSplat (Zhang et al., 2026a), instead organize Gaussians around explicit 3D anchors. When the budget varies, the anchor set itself must change, and with it the extent each anchor should cover. We study how geometry and appearance can guide this allocation, and how the resulting distribution should shape local reconstruction.

Our central idea is to use the spatial layout of Gaussian groups to connect budget allocation with local decoding. Geometry and appearance provide complementary allocation cues: spatial distance encourages coverage, while appearance differences separate nearby locations with different visual content. The resulting layout also reflects local density: distances between neighboring groups provide a geometric reference for the spatial scales used in local decoding.

We introduce AllocSplat (Fig. 2) to realize this idea in feed-forward reconstruction. Anchor candidates are back-projected from the depth of a pretrained visual geometry model (Lin et al., 2026), which is initialized from its released weights and fine-tuned with the anchor model. A joint geometry–appearance allocation rule selects 3D anchors for local Gaussian groups from these candidates, and a density-adaptive local decoder uses their spacing to adapt Gaussian sizes and offsets.

As in farthest-point sampling, the selection order does not depend on the budget, so a smaller allocation reuses a prefix of a larger one's centers while its scales and Gaussian groups are recomputed. Controlled comparisons within AllocSplat show that appearance-aware allocation gives the largest PSNR gains over spatial selection at small budgets, and that density-adaptive scales outperform fixed and globally scaled footprints.

Our main contributions are:

- We introduce AllocSplat for budget-adaptive feed-forward reconstruction, in which joint geometry–appearance allocation places the 3D anchors of Gaussian groups and a density-adaptive local decoder derives each group's spatial scale from the selected-anchor spacing.
- We show that decoding Gaussian groups for the requested allocation improves reconstruction over retaining groups from a larger allocation, with further gains from updating local scales.
- Quantitative and qualitative evaluations show improved quality–count trade-offs on DL3DV-140 and generalization to RealEstate10K and Mip-NeRF 360, with controlled comparisons of each component.

2 Related Work

Feed-forward Gaussian reconstruction. Generalizable novel view synthesis predicts scenes from posed images instead of optimizing each one independently: first by conditioning radiance fields (Mildenhall et al., 2020) on image features, ray-wise aggregation or cost volumes (Yu et al., 2021; Wang et al., 2021; Chen et al., 2021), and since 3DGS (Kerbl et al., 2023) by predicting pixel-aligned Gaussians from probabilistic depth, plane-swept cost volumes or combined monocular and multi-view depth (Charatan et al., 2024; Chen et al., 2024b; Xu et al., 2025). Pixel alignment grounds predictions geometrically but couples primitive count to resolution and view count. AllocSplat keeps dense evidence internally and allocates output capacity through spatial anchors.

Geometry foundation models and pose-free prediction. Pointmap regressors provide depth, confidence and multi-level features from uncalibrated images (Wang et al., 2024; Leroy et al., 2024; Wang et al., 2025), and feed-forward Gaussian methods build on them directly, decoding Gaussians

on a frozen geometry, predicting them in a canonical frame without input poses, or estimating cameras and appearance jointly (Smart et al., 2024; Ye et al., 2025; Zhang et al., 2025; Ye et al., 2026). In AllocSplat, back-projected depth defines the candidate pool and backbone features initialize and condition the anchors. We evaluate the resulting reconstruction with known cameras.

Compact representations. Scaffold-GS (Lu et al., 2024) decodes neural Gaussians from sparse anchors, while pruning, quantization (Lee et al., 2024; Fan et al., 2024), and entropy coding (Chen et al., 2024a) compress scene-specific models. Feed-forward alternatives construct compact representations directly. TokenGS (Ren et al., 2026) predicts Gaussian attributes, including 3D means, from learnable tokens; GlobalSplat (Itkin et al., 2026) aggregates observations globally; C3G (An et al., 2026) uses learnable aggregation tokens; and ZipSplat (Veicht et al., 2026) clusters backbone features into a test-time-adjustable token set. Explicit spatial organization is also central to AnchorSplat (Zhang et al., 2026a), which grounds anchor-aligned Gaussians in 3D geometric priors. Concurrent work, LocusGS (Li et al., 2026), improves the spatial coherence of fixed-budget Gaussian queries by progressively refining learned anchor centers and radii. AllocSplat instead selects a budget-dependent set of depth-grounded anchors.

Adaptive and budget-controlled allocation. SparseSplat (Zhang et al., 2026b) turns image entropy into sampling probabilities for sparse 3D anchors, with a temperature controlling expected sparsity. ATSplat (Cho et al., 2026) expands uncertain anchor tokens across decoder blocks at fixed per-stage ratios, or at scene-dependent ones under optional thresholds; neither takes an exact requested count. EcoSplat (Park et al., 2026) and AdaptiveSplat (Singhal et al., 2026) address the problem we do, a user-specified primitive count, from the pixel-aligned side. EcoSplat conditions its opacity, covariance and color heads on the count, with the center head frozen, and keeps the Gaussians of highest opacity; AdaptiveSplat clusters the pooled point cloud in position and color, prunes low-texture clusters to meet the count, and re-predicts the survivors' scales, rotations, opacities and colors from per-view retention masks. Both adapt the survivors to the budget, the second locally, while their centers stay on the pixel grid. AllocSplat couples selected-anchor spacing with Gaussian decoding: the same neighbor-derived scale constrains both offsets and sizes, and is recomputed when the budget changes.

3 METHOD

AllocSplat turns a requested Gaussian budget into a spatial allocation and a set of local Gaussian groups (Fig. 2). A visual geometry model (VGM) first supplies 3D candidates and image features. Joint geometry–appearance selection determines where the anchors are placed; their neighbor spacing then sets the spatial scale for local decoding. Image features and cross-anchor context condition the Gaussians produced within this scale.

3.1 SPATIAL ANCHORS AND LOCAL GAUSSIAN GROUPS

Let $\mathcal{I} = \{(I_v, \mathbf{K}_v, \mathbf{T}_v)\}_{v=1}^{V}$ denote V posed input views, with camera intrinsics $\mathbf{K}_v$ and camera-to-world poses $\mathbf{T}_v$. We distinguish the anchor budget M from the Gaussian budget $B = KM$, where K is the fixed number of Gaussians per anchor. Given a feasible integer M, the model constructs an ordered set of spatial anchors

$$\mathcal{A}_M = (a_1^{(M)}, \dots, a_M^{(M)}), \qquad a_i^{(M)} = (\mathbf{p}_i, \rho_i^{(M)}, \mathbf{z}_i^{(M)}), \tag{1}$$

where $\mathbf{p}_i \in \mathbb{R}^3$ is the anchor position, $\rho_i^{(M)}$ is its local spatial scale, and $\mathbf{z}_i^{(M)}$ is its contextualized content token; the superscript marks the dependence on the requested budget. A shared decoder $\mathcal{D}$ maps every anchor to a local group of K Gaussian primitives,

$$\mathcal{G}_i^{(M)} = \mathcal{D}(\mathbf{z}_i^{(M)}, \mathbf{p}_i, \rho_i^{(M)}), \qquad |\mathcal{G}_i^{(M)}| = K, \tag{2}$$

and the scene representation concatenates these groups, $\mathcal{G}_M = \operatorname{concat}_{i=1}^{M} \mathcal{G}_i^{(M)}$. Differentiable splatting renders $\mathcal{G}_M$ for novel-view supervision. The budget M controls spatial allocation and K the capacity of each group, and changing M produces KM Gaussians with the same decoder and weights. With N candidates, the supported Gaussian counts are $K, 2K, \dots, KN$.

Both the local scale and the content token depend on the selected anchor set. The following steps determine the centers, derive their scales, and decode their budget-specific content.

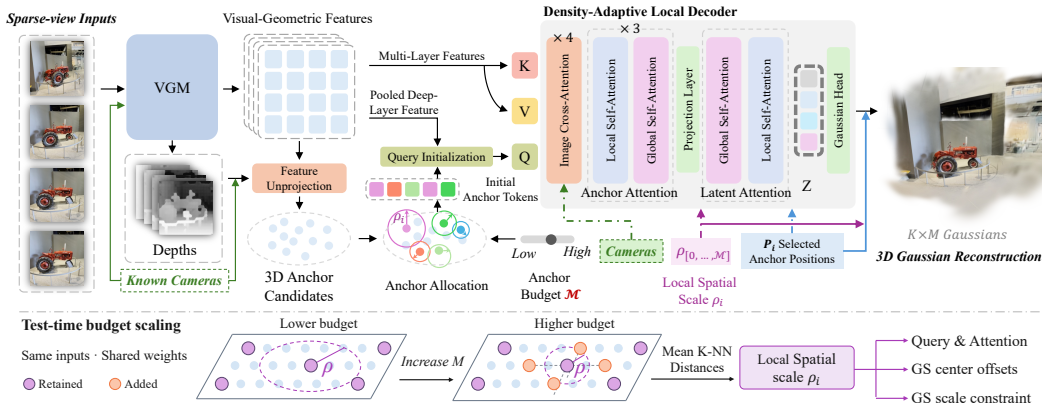

Figure 2: **AllocSplat.** Sparse views and their known cameras pass through a visual geometry model (VGM), whose multi-layer features and depths are unprojected into a pool of 3D anchor candidates. A joint geometry–appearance farthest-point rule allocates exactly M of them, and the resulting density sets each anchor's local spatial scale ρ_i, so the same centers are decoded differently at different budgets. The Gaussian head emits K primitives per anchor, KM in total. Normalization and residual paths are omitted; the Gaussian groups are schematic.

3.2 Spatially Grounded Candidate Evidence

To ground allocation in the observed scene, a pretrained VGM predicts per-view depth D_v, confidence C_v, and multi-level visual features. We resample these outputs on an $H_e \times W_e$ evidence grid and back-project each cell center into world coordinates,

$$\mathbf{x}_{v,u} = \mathbf{T}_v \, \pi^{-1}(\mathbf{u}, D_v(\mathbf{u}), \mathbf{K}_v), \tag{3}$$

where π^{-1} denotes pinhole-camera back-projection. Collecting the grid cells from all views produces a redundant candidate pool $\mathcal{C} = \{c_j\}_{j=1}^{N}$, $N = VH_eW_e$, that grows with the available observations; each candidate keeps a fixed index in it.

Each candidate $c_j = (\mathbf{x}_j, \mathbf{f}_j, q_j)$ combines its 3D position with deepest-layer VGM features, RGB appearance, viewing direction, and the confidence C_j read off C_v at the cell it came from; learned linear projections fuse these into the descriptor $\mathbf{f}_j$, which is used both to distinguish complementary scene content during allocation and to initialize the selected anchor. We define an evidence-quality term

$$q_j = \max(C_j, \epsilon)\,(\beta + \hat{\sigma}_j), \qquad \sigma_j = \left\| \mathbf{f}_j - \frac{1}{N}\sum_{k=1}^{N} \mathbf{f}_k \right\|_{\mathrm{RMS}}, \tag{4}$$

where $\hat{\sigma}_j$ is σ_j normalized by its maximum within the scene. Confidence downweights unreliable geometry, while feature deviation favors distinctive evidence over visually uniform regions.

3.3 Joint Geometry–Appearance Allocation

We rank new centers by the evidence they add to the selected set, combining spatial coverage with complementary feature content. The residuals below are computed from candidate positions and features before any Gaussian is decoded. Let $\mathcal{P}_t = (\mathbf{p}_1, \ldots, \mathbf{p}_t)$ be the first t selected centers. For every remaining candidate c_j, we measure the spatial coverage it adds by its distance to the nearest selected anchor,

$$d_j^{(t)} = \min_{1 \le i \le t} \|\mathbf{x}_j - \mathbf{p}_i\|_2, \qquad \hat{d}_j^{(t)} = \frac{d_j^{(t)}}{\max_k d_k^{(t)} + \epsilon}. \tag{5}$$

Spatial distance alone treats two nearby candidates with different content as redundant, so we also maintain a deep-feature residual

$$r_{j,\mathrm{sem}}^{(t)} = \min_{1 \le i \le t} \left[1 - \cos(\bar{\mathbf{f}}_j, \bar{\mathbf{f}}_i)\right], \tag{6}$$

where $\bar{\mathbf{f}}_i$ is the compact normalized signature of the candidate selected as anchor i. This term modulates the priority of spatially uncovered candidates according to their content complementarity.

A detail factor $\phi_j^{(t)} \geq 1$ increases priority where the shallow-feature residual exceeds the deep-feature residual. Its bounded form is specified in App. A.1.

The allocation score uses reliability-weighted spatial coverage as its base, with deep-feature and detail factors modulating the ranking:

$$s_j^{(t)} = q_j \, \hat{d}_j^{(t)} \left(1 + r_{j,\mathrm{sem}}^{(t)}\right) \phi_j^{(t)}. \tag{7}$$

The first block is ranked by evidence quality q_j, later blocks by Eq. (7). We allocate in blocks of four candidates under the current residual state, then refresh the residuals for the next block; already selected candidates are excluded, candidates within a block are ordered by decreasing score with ties broken by candidate index, and the final block contributes only what is needed to reach M.

The score depends only on the evidence and preceding selections, giving a budget-independent center order. This fixed rule operates on descriptors learned with the rest of the model. The selected centers now provide the geometry for local decoding.

3.4 Density-Adaptive Local Decoding

To connect the selected layout to Gaussian geometry, we derive each anchor's local spatial scale from its nearby selected centers. Let $\mathcal{N}_i^{(M)}$ contain the $k_M = \min(8, M-1)$ nearest other selected anchors. For $M > 1$, we set

$$\rho_i^{(M)} = \mathrm{clip}\left(\frac{1}{k_M} \sum_{j \in \mathcal{N}_i^{(M)}} \|\mathbf{p}_i - \mathbf{p}_j\|_2, \rho_{\min}, \rho_{\max}\right). \tag{8}$$

For $M = 1$, we use $\rho_{\min}$.

Each anchor token combines the selected candidate's descriptor and confidence with a prior encoding its allocation rank and preceding selections (App. A.1). Each anchor reads multi-level image features through cross-attention; its positional-bias MLP takes the image token's displacement from the projected anchor center, log camera-space depth and log local scale. Global and local attention exchange context across anchors (App. A.1). A scene code then modulates every token, $\tilde{\mathbf{z}}_i = (1+\boldsymbol{\gamma}) \odot \mathbf{z}_i + \boldsymbol{\beta}$ with $(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \mathrm{MLP}(\frac{1}{M}\sum_j \mathbf{z}_j)$, so each token incorporates context from the whole allocation before decoding.

A shared decoder predicts K anisotropic Gaussians from each modulated anchor token. For Gaussian k associated with anchor i, its center is parameterized locally as

$$\boldsymbol{\mu}_{ik}^{(M)} = \mathbf{p}_i + \rho_i^{(M)} \tanh(\Delta\boldsymbol{\mu}_{ik}^{(M)}). \tag{9}$$

This bounds each coordinate of the offset by $\rho_i^{(M)}$, giving $\|\boldsymbol{\mu}_{ik}^{(M)} - \mathbf{p}_i\|_2 \leq \sqrt{3}\rho_i^{(M)}$. Each Gaussian's principal-axis scales are capped by the same local scale, with an additional global cap specified in App. A.1. Rotation, opacity and color are decoded alongside the offset. The scale $\rho_i^{(M)}$ defines the local coordinate frame for decoding: closely spaced anchors use smaller offset and size bounds, while isolated anchors cover larger neighborhoods, within the clipping limits of Eq. (8).

Changing the budget. At inference, M is the only representation-size control. For fixed input evidence and allocation settings, the ordered centers satisfy $\mathcal{P}_M = \mathcal{P}_{M'}[:M]$ for $1 \leq M < M' \leq N$: completed blocks share the same preceding selections, and partial blocks use the same ranking (App. A.5). Several budgets can thus reuse the cached evidence and center order. At each budget, we recompute the neighbor-derived scales and contextualized decoding to construct the local Gaussian groups. Section 4.4 isolates the effects of these updates while holding the centers and Gaussian count fixed.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and protocol. We evaluate posed reconstruction on the 140 DL3DV test scenes (Ling et al., 2024). Cross-dataset generalization is tested on RealEstate10K (Zhou et al., 2018), with the pixelSplat and DepthSplat indices, and on Mip-NeRF 360 (Barron et al., 2022), training only on DL3DV (App. C.3). On DL3DV we use the released DepthSplat input indices, which match those of TokenGS and ZipSplat, and score the $50 - V$ non-input frames. The Gaussians of every method are re-rendered, unchanged, onto a common 252×252 target camera and scored against the same ground truth (App. A.2).

Metrics. We report PSNR, SSIM and VGG-LPIPS, averaged within each scene and then across scenes, together with the number of Gaussian primitives (#GS). The parameter count per primitive differs across methods (App. A.3).

Implementation details. We train one network on the official DL3DV training split, which excludes the 140 test scenes, for two, four and six input views with default budgets $M = 512/1024/1536$. Late in training, several budgets are sampled per view count (App. A.1); all budget sweeps change only M at inference. Each anchor decodes $K = 36$ Gaussians from a pool of $324V$ candidates. The VGM is DA3-Giant (Lin et al., 2026), fine-tuned with the anchor model. Its pre-training includes DL3DV; overlap with the 140 test scenes remains unverified (App. A.1). App. A.1–A.4 give the architecture, schedule and statistical settings.

Baselines. We compare with the released models in Tab. 1, and with ZipSplat at matched primitive counts.

4.2 RECONSTRUCTION QUALITY

At its default budgets, AllocSplat uses 18,432/36,864/55,296 Gaussians at 2/4/6 input views. It exceeds all evaluated feed-forward baselines in PSNR and SSIM at all three view counts, with fewer Gaussians than every baseline configuration (Tab. 1, Fig. 3).

At matched primitive counts with four views, AllocSplat exceeds ZipSplat by 0.29, 0.83 and 1.65 dB at 9,216, 18,432 and 36,864 Gaussians; the margin grows with the budget.

Decoding the full candidate pool with the same weights adds 26.6% more Gaussians and raises PSNR by 0.27/0.35/0.32 dB at 2/4/6 views, with higher SSIM and lower LPIPS. This full-pool setting has the lowest LPIPS at two and four views; at six views YoNoSplat is lower (0.154 against 0.156) with $4.3\times$ as many Gaussians. On the uncropped 16:9 frame, a variant trained at 252×448 and evaluated with its full pool has higher PSNR and SSIM than DepthSplat and TokenGS at every view count, while DepthSplat has lower LPIPS at four and six views (App. C.5).

The starred rows optimize each scene for 50 steps using only its context images. Optimizing anchor latents improves PSNR by 0.53/1.81/2.70 dB at 2/4/6 views, compared with 0.07/1.18/1.82 dB for TokenGS optimizing its Gaussian tokens (App. C.2). AllocSplat gains more PSNR from test-time optimization at every view count.

4.3 BUDGET SCALING AND LOCAL DETAIL

With the input views fixed, a larger anchor budget improves PSNR, SSIM and LPIPS without retraining: at four views, increasing from 18,432 to 46,656 Gaussians raises PSNR from 22.30 to 24.22 dB (App. B.1). Figure 4 shows how the local scales change. From 128 to 1,024 anchors, the median scale of the drawn anchors shrinks by a factor of five and the largest by less than two, so the ratio between the largest and smallest scale grows from 1.5 to 5.5. These anchors' local scales change by different proportions as the budget increases. Section 4.4 compares this local adaptation with fixed and globally scaled footprints.

Table 1: **DL3DV-140 novel-view synthesis at default and full-pool budgets.** Cells report PSNR↑ / SSIM↑ / LPIPS↓ under the common-canvas protocol of Sec. 4.1. AllocSplat rows differ only in the anchor budget M set at inference; the fixed-M row holds capacity constant across view counts and so coincides with the default row at two views. *These rows use 50 steps of per-scene test-time optimization, fitting only the context images, which are not among the scored frames (App. C.2).

Method	#GS (2/4/6)	2 views	4 views	6 views
MVSplat (Chen et al., 2024b)	131/262/393 K	15.67 / 0.413 / 0.499	17.78 / 0.517 / 0.370	18.42 / 0.554 / 0.346
DepthSplat (Xu et al., 2025)	131/262/393 K	18.24 / 0.572 / 0.337	21.85 / 0.740 / 0.194	22.86 / 0.783 / 0.161
TokenGS (Ren et al., 2026)	262/262/262 K	19.52 / 0.614 / 0.430	22.88 / 0.745 / 0.303	23.99 / 0.782 / 0.274
ZipSplat (Veicht et al., 2026)	21/41/62 K	19.86 / 0.612 / 0.334	22.35 / 0.708 / 0.230	23.28 / 0.742 / 0.198
YoNoSplat (Ye et al., 2026)	100/201/301 K	19.55 / 0.605 / 0.329	23.38 / 0.771 / 0.187	24.56 / 0.814 / **0.154**
AllocSplat (fixed M=512)	18/18/18 K	20.73 / 0.660 / 0.304	22.30 / 0.707 / 0.274	22.47 / 0.705 / 0.279
AllocSplat (default)	18/37/55 K	20.73 / 0.660 / 0.304	23.87 / 0.779 / 0.199	25.16 / 0.818 / 0.165
AllocSplat (full pool)	23/47/70 K	**21.00 / 0.675 / 0.288**	**24.22 / 0.794 / 0.185**	**25.48 / 0.830** / 0.156
TokenGS + TTO*	262/262/262 K	19.59 / 0.620 / 0.405	24.05 / 0.783 / 0.264	25.80 / 0.833 / 0.228
AllocSplat + TTO (default)*	18/37/55 K	**21.26 / 0.685 / 0.266**	**25.68 / 0.834 / 0.142**	**27.86 / 0.881 / 0.102**

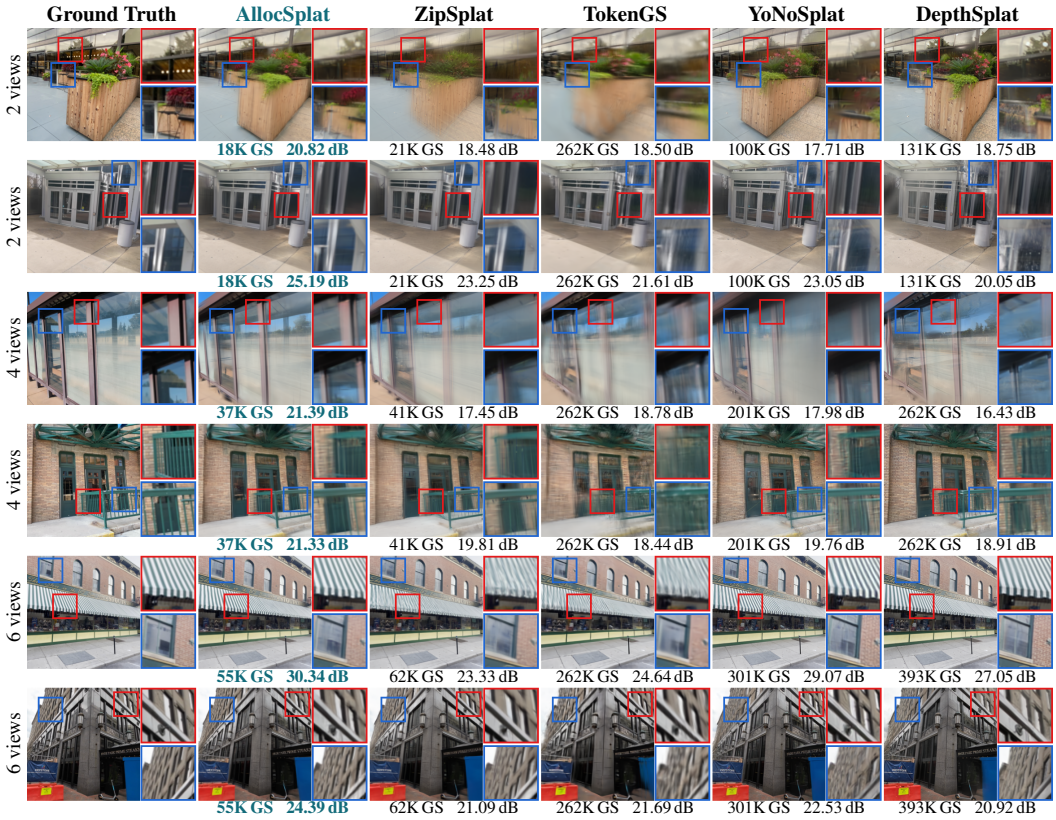

Figure 3: **DL3DV-140 reconstructions at two, four and six input views.** Two scenes per view count; each cell pairs a held-out view with two enlarged regions, and method names carry their Gaussian count and the frame's PSNR. The views shown are two to nine frames from the nearest input view.

4.4 Spatial Allocation and Local Decoding

Geometry–appearance allocation. We train variants that replace joint selection with Random, spatial FPS or position-space k-means++ selection (Tab. 2). Joint selection has the highest PSNR at every budget, 0.90–1.21 dB above Random. Its margin over spatial FPS falls from 1.73 dB at 256 anchors to 0.14 dB at 1,024. It also has the best SSIM at every budget and the best LPIPS at 256 and 512 anchors; FPS has the lowest LPIPS at 1,024 (Tab. 8). The PSNR gain over spatial FPS is largest at the smallest budget.

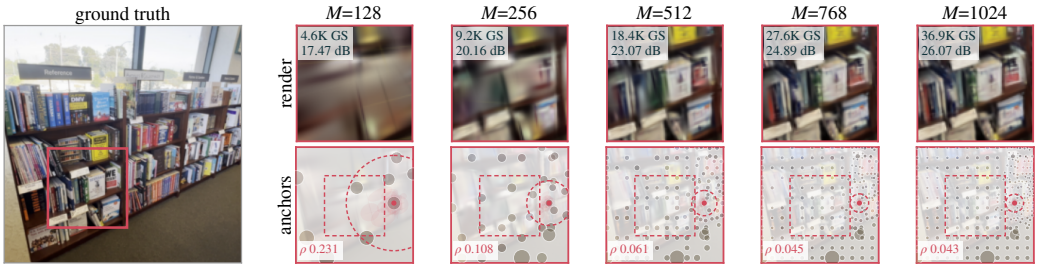

Figure 4: **Capacity, scale and detail move together.** AllocSplat reconstructs the same DL3DV region from four input views at 128/256/512/768/1,024 anchors (4.6K to 36.9K Gaussians), changing only M at inference. *Top*: the rendered crop; the number beside it is the PSNR of this whole target view at that budget. *Bottom*: the same region in anchor space. Gray discs are the nearby anchors, area giving the local spatial scale relative to the others on a log axis, so their number tracks the budget. Marked in red is one anchor center shared by all budgets, chosen to illustrate decreasing scale: the dashed circle is a reference for its projected local scale ρ, and the shading shows the 1σ extents of its K Gaussians. Their offsets and sizes are constrained by ρ; the circle is not a strict enclosing boundary.

Table 2: **Selection-rule ablation.** PSNR↑ on four-view DL3DV-140 with $K = 36$; each row is trained with the listed selection rule. Full PSNR/SSIM/LPIPS results are in Tab. 8.

Selection rule	$M = 256$	$M = 512$	$M = 1024$
Random	19.436	21.402	22.960
Position-space k-means++	18.419	20.006	22.955
Spatial FPS	18.909	21.577	23.728
Joint geometry–appearance	**20.641**	**22.297**	**23.870**

Budget-specific re-decoding. The anchor centers of a smaller allocation are a prefix of those of a larger one on all 140 scenes (App. A.5). With these centers and the Gaussian count fixed, we compare keeping the groups decoded at $M = 1024$, re-decoding with scales frozen at their $M = 1024$ values, and full re-decoding (Tab. 3). Keeping the groups is 5.09 and 3.61 dB below full re-decoding at $M = 256$ and 512. Re-decoding with frozen scales recovers 4.15 and 3.26 dB, and updating the scales adds 0.95 and 0.35 dB. Both re-decoding and scale updates improve reconstruction at the requested budget.

Table 3: **Budget-specific re-decoding at fixed centers and Gaussian counts.** Four-view DL3DV-140, scored on the 46 non-input frames. At each budget, all strategies use the same centers and primitive count. Keep-groups retains the first M groups from an M=1024 decode; freeze-scale re-decodes at M with each ρ_i held at its M=1024 value; full re-decoding also updates these scales. Metrics use float renders (App. A.3). Cells report PSNR↑ / SSIM↑ / LPIPS↓.

Strategy	9,216 Gaussians (M=256)	18,432 Gaussians (M=512)
Keep groups from M=1024	15.555 / 0.4730 / 0.5513	18.766 / 0.6130 / 0.3763
Re-decode, freeze scale ρ_i	19.701 / 0.5816 / 0.3890	22.027 / 0.6872 / 0.2793
Full re-decoding at M	**20.647 / 0.6108 / 0.3650**	**22.377 / 0.6976 / 0.2718**

Density-adaptive local scale. We train variants that replace the local scale with a fixed 18-pixel footprint or a global budget rule, $18(1024/M)^{0.67}$ pixels (App. A.1). Local adaptation exceeds the fixed footprint by 0.301–1.708 dB and global scaling by 0.114–0.980 dB, with the lowest LPIPS at every budget (Tab. 4). Replacing the local offset bound with its maximum constant value, while keeping the local size bound, lowers PSNR by 0.04–0.08 dB (App. C.1). Scales derived from local anchor spacing outperform both fixed and globally scaled footprints.

Group size. At matched primitive counts, a variant with $K = 16$ exceeds $K = 36$ by 0.60–1.34 dB PSNR at all seven tested operating points; at equal anchor counts, $K = 36$ is best among $K \in \{16, 36, 64\}$ (App. B.2). With the candidate pool fixed, $K = 16$ reaches at most 20,736 Gaussians and 23.34 dB at four views, whereas $K = 36$ reaches 24.22 dB with 46,656. Smaller groups use primitives more efficiently, while larger groups give a higher maximum capacity.

Table 4: Local spatial scale on DL3DV-140 with four input views. Cells report PSNR↑ / LPIPS↓. Each row is trained with the listed scale rule.

Scale rule	256	512	1024	1296
Fixed 18 px	19.375 / 0.3977	20.802 / 0.2864	22.162 / 0.2417	23.923 / 0.2085
Global budget scaling	20.486 / 0.3696	21.388 / 0.2780	22.890 / 0.2215	24.110 / 0.1977
Local adaptive	**20.641 / 0.3652**	**22.297 / 0.2736**	**23.870 / 0.1988**	**24.224 / 0.1850**

4.5 Generalization and Inference Cost

Cross-dataset generalization. Trained only on DL3DV, AllocSplat reaches 26.51 dB with 18K Gaussians on the RealEstate10K pixelSplat index (6,474 scenes, two views). It exceeds the strongest method run from DL3DV weights, YoNoSplat, by 0.88 dB with a fifth of its Gaussians. Against models trained on RealEstate10K, it exceeds YoNoSplat and is 0.31 and 1.27 dB below MVSplat and DepthSplat, which use about 7.1× as many Gaussians (Tab. 12). On the 199-scene DepthSplat-index subset, it has the best PSNR, SSIM and LPIPS among the compared methods at two, four and six views (Tab. 5). On the nine Mip-NeRF 360 scenes with six views, it reaches 21.62 dB with 55K Gaussians, compared with 20.95 dB for DepthSplat at 393K and 20.87 dB for YoNoSplat at 301K (App. C.4). The compact anchor representation transfers across datasets.

Table 5: **RealEstate10K generalization** on a 199-scene subset of the DepthSplat index, with DL3DV weights where available. *Has seen RealEstate10K in training (App. C.3). Cells report PSNR↑ / SSIM↑ / LPIPS↓.

Method	2 views	4 views	6 views
DepthSplat* (Xu et al., 2025)	17.100 / 0.624 / 0.371	21.397 / 0.797 / 0.213	23.514 / 0.856 / 0.159
TokenGS (Ren et al., 2026)	17.710 / 0.631 / 0.435	22.367 / 0.775 / 0.295	23.972 / 0.812 / 0.260
ZipSplat* (Veicht et al., 2026)	17.132 / 0.600 / 0.407	20.961 / 0.703 / 0.260	22.667 / 0.740 / 0.200
AllocSplat (default)	**18.873 / 0.678 / 0.333**	**23.903 / 0.824 / 0.194**	**26.196 / 0.871 / 0.150**

Inference cost. At four views, AllocSplat takes 136.7 ms at $M = 256$ and 382.3 ms at $M = 1024$ on an H100. Backbone time and peak memory stay nearly constant; the increase comes from sequential allocation and token initialization. Because smaller budgets reuse the shared center order, serving four budgets for one scene takes 966 ms instead of 1,205 ms for independent inference (App. C.6).

5 Conclusion

We present AllocSplat, a budget-adaptive feed-forward Gaussian splatting framework that links where Gaussian groups are placed to how each group is decoded. A joint geometry–appearance rule selects 3D anchors from depth-based candidates using scene coverage and differences in visual content. A density-adaptive local decoder uses the spacing between selected anchors to bound Gaussian offsets and sizes at each budget. One set of weights serves different budgets without retraining. Experiments show improved quality–count trade-offs on DL3DV-140 and generalization to RealEstate10K and Mip-NeRF 360. Controlled comparisons further show that decoding for the requested allocation outperforms reusing groups from a larger reconstruction, and that density-adaptive scales outperform fixed and globally scaled footprints. These findings highlight the selected spatial layout as an effective guide for local reconstruction under varying Gaussian budgets.

Limitations and future work. Latency grows with the budget because allocation and token initialization run sequentially (Sec. 4.5); extending the CUDA acceleration used for allocation to token initialization could reduce this cost. Smaller Gaussian groups use primitives more effectively at matched counts but cap the capacity of a fixed candidate pool (Sec. 4.4). The group size could instead grow with the budget, using small groups while unselected candidates remain and larger groups once the pool is exhausted; varying it across anchors with local appearance complexity could further concentrate primitives on detailed regions. Anchor candidates are also back-projected from backbone depth, so geometric errors can leave regions poorly represented; refining candidate positions with multi-view evidence could reduce this dependence.

REPRODUCIBILITY STATEMENT

The evaluation protocol, including the input indices, scored frames and common rendering canvas, is specified in Sec. 4.1 and App. A.2, with metric implementations in App. A.3. App. A.1 gives the architecture, loss and training settings, and App. C.6 the timing procedure. Code and trained models will be released.

ETHICS STATEMENT

This work reconstructs 3D scenes from posed images and introduces no new data collection. The DL3DV-140, RealEstate10K and Mip-NeRF 360 benchmarks are public and are used under their licenses. We do not attempt to identify people or places in them. Compact scene reconstruction has the dual-use character common to 3D capture: it lowers the cost of legitimate applications and, equally, of reconstructing spaces without the consent of their occupants. We do not release any capture tooling beyond the reconstruction model itself.

AI USE STATEMENT

We used a large language model assistant for literature retrieval and synthesis, manuscript drafting and editing, and reference formatting. The authors designed the experiments and are responsible for verifying AI-assisted content against the original sources and experimental records, and for all methodological decisions, results and conclusions.

REFERENCES

Honggyu An, Jaewoo Jung, Mungyeom Kim, Chaehyun Kim, Minkyeong Jeon, Jisang Han, Kazumi Fukuda, Takuya Narihira, Hyunah Ko, Junsu Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. Learning compact 3D representations from feed-forward novel view synthesis. In *CVPR*, 2026.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *CVPR*, 2024.

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-grid assisted context for 3D Gaussian splatting compression. In *ECCV*, 2024a.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D Gaussian splatting from sparse multi-view images. In *ECCV*, 2024b.

In Cho, Jeonghwan Cho, Mijin Yoo, Gim Hee Lee, and Seon Joo Kim. ATSplat: Compact feed-forward 3D Gaussian splatting with adaptive token expansion. *arXiv preprint arXiv:2607.20417*, 2026.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. In *NeurIPS*, 2024.

Roni Itkin, Noam Issachar, Yehonatan Keypur, Xingyu Chen, Anpei Chen, and Sagie Benaim. GlobalSplat: Efficient feed-forward 3D Gaussian splatting via global scene tokens. *arXiv preprint arXiv:2604.15284*, 2026.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):139:1–139:14, 2023.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian representation for radiance field. In *CVPR*, 2024.

Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3D with MASt3R. In *ECCV*, 2024.

Wenyu Li, Sidun Liu, Tongrui Hu, Peng Qiao, and Yong Dou. LocusGS: Spatially grounded tokens for feed-forward 3D Gaussian splatting. *arXiv preprint arXiv:2608.12825*, 2026.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *ICLR*, 2026.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A large-scale scene dataset for deep learning-based 3D vision. In *CVPR*, 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 *CVPR*, 2024.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. EcoSplat: Efficiency-controllable feed-forward 3D Gaussian splatting from multi-view images. In *CVPR*, 2026.

Jiawei Ren, Michal Jan Tyszkiewicz, Jiahui Huang, and Zan Gojcic. TokenGS: Decoupling 3D Gaussian prediction from pixels with learnable tokens. In *CVPR*, 2026.

Badrinath Singhal, Srihari K G, Sreehari Iyer, Ankit Dhiman, and R. Venkatesh Babu. AdaptiveSplat: Texture aware controllable 3D Gaussian allocation for feed-forward reconstruction. In *ECCV*, 2026.

Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3R: Zero-shot Gaussian splatting from uncalibrated image pairs. *arXiv preprint arXiv:2408.13912*, 2024.

Alexander Veicht, Sunghwan Hong, Dániel Baráth, and Marc Pollefeys. ZipSplat: Fewer Gaussians, better splats. *arXiv preprint arXiv:2606.05102*, 2026.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, 2025.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *CVPR*, 2021.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, 2024.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Dániel Baráth, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian splatting and depth. In *CVPR*, 2025.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3D Gaussian splats from sparse unposed images. In *ICLR*, 2025.

Botao Ye, Boqi Chen, Haofei Xu, Dániel Baráth, and Marc Pollefeys. YoNoSplat: You only need one model for feedforward 3D Gaussian splatting. In *ICLR*, 2026.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. FLARE: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *CVPR*, 2025.

Xiaoxue Zhang, Xiaoxu Zheng, Yixuan Yin, Tiao Zhao, Kaihua Tang, Michael Bi Mi, Zhan Xu, and Dave Zhenyu Chen. AnchorSplat: Feed-forward 3D Gaussian splatting with 3D geometric priors. In *CVPR*, 2026a.

Zicheng Zhang, Xiangting Meng, Ke Wu, and Wenchao Ding. SparseSplat: Towards applicable feed-forward 3D Gaussian splatting with pixel-unaligned prediction. In *CVPR*, 2026b.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM TOG*, 37(4):65:1–65:12, 2018.

A Implementation and Evaluation Details

A.1 Architecture and Training

Geometry backbone and anchor encoder. We use DA3-Giant-1.1 (Lin et al., 2026) and resize input images to 252×252. The VGM is initialized from its released weights and fine-tuned jointly with the anchor model during the first training stage below. Candidate descriptors use the deepest-layer features on an 18×18 grid per input view; layer 9 supplies the shallow selection signature. The anchor encoder has width 1024 and produces 256-dimensional anchor latents. Anchors read the full feature maps from layers $\{9, 19, 29, 39\}$ using 8-head image cross-attention. The network includes a six-layer anchor transformer and a two-layer global–local context module. The latter applies global attention in layer 0 and local attention in layer 1. Each local neighborhood contains the anchor itself and its $\min(16, M-1)$ nearest other centers in 3D; the relative-bias MLP takes the relative displacement divided by ρ and the distance divided by ρ. The image cross-attention bias is an MLP of the image-token coordinate minus the projected anchor center, log projected depth and log ρ. A two-layer MLP of hidden width 1024 decodes 36 anisotropic Gaussians per anchor.

Anchor geometry and allocation. The token-prior and evidence projections have width 1024. Candidate positions are retained as anchor centers. For DL3DV, poses are expressed relative to the first input camera and camera translations are multiplied by 0.15. The resulting world-to-camera matrices directly condition DA3. Both the trainable and frozen paths bypass DA3's inference-time extrinsic normalization; the camera encoder also applies no normalization. Predicted depths are clipped to $[0.05, 6.0]$ without explicit rescaling or post-hoc alignment. Adaptation to the supplied camera scale is learned during joint fine-tuning; depths are not multiplied by 0.15. The geometric constants below use these model coordinates. Equation (8) uses $\rho_{\min} = 0.03$ and $\rho_{\max} = 0.5$. We set the quality offset β in Eq. (4) to 0.25 and use blocks of four selections. A minimum-separation mask assigns score -1 to candidates within 0.005 of a selected anchor while sufficiently separated candidates remain available. Previously selected indices are assigned $-\infty$ and cannot be selected again; unselected nearby candidates can fill a block once the preferred candidates are exhausted. Gaussian scales are additionally capped at 0.075. The anchor transformer exchanges context through 96 learned register tokens in addition to the anchor tokens. The rank code $\mathbf{e}(i)$ of Eq. (11) is a sinusoidal encoding of $\log(1+i)$. The deep and shallow signatures used in Eq. (6) and the detail term are 16-dimensional average-pooled, ℓ_2-normalized features. Stable sorting breaks score ties by candidate index.

For the detail term, let $r^{(t)}_{j,\mathrm{sh}}$ be the residual of Eq. (6) computed on the shallow layer-9 signature instead of the deep one, and let $e^{(t)}_j = [r^{(t)}_{j,\mathrm{sh}} - r^{(t)}_{j,\mathrm{sem}}]_+$ be the part of it the deep signature does not already account for. We use

$$\phi^{(t)}_j = 1 + \lambda_{\mathrm{det}} \,\mathrm{clip}\left(\frac{e^{(t)}_j}{r^{(t)}_{j,\mathrm{sem}} + e^{(t)}_j + \epsilon}, 0, 0.75\right) e^{(t)}_j, \qquad \lambda_{\mathrm{det}} = 1. \tag{10}$$

Anchor token initialization. Writing $\mathbf{f}_i$ and C_i for the selected candidate descriptor and confidence, each anchor token is initialized from its evidence and a prior on its place in the allocation,

$$\mathbf{z}^{(0)}_i = \mathrm{LN}\big(\boldsymbol{\pi}_i + \sigma(\mathbf{W}_g \boldsymbol{\pi}_i) \odot (\mathbf{W}_f \mathbf{f}_i + \mathbf{W}_x[\mathbf{p}_i, C_i])\big), \qquad \boldsymbol{\pi}_i = \mathrm{MLP}\left(\mathbf{e}(i), \mathbf{s}_{<i}\right), \tag{11}$$

where $\mathbf{e}(i)$ encodes the allocation rank. The prior is evaluated sequentially: $\mathbf{s}_{<i}$ contains the preceding anchors' mean descriptor, position mean and standard deviation, and the detached residual carried from the previous anchor. For the first anchor, the feature and position statistics are zero and the residual is one. History advances after each anchor, including within a selection block.

Optimization. We train on the official DL3DV training split with the 140 benchmark scenes held out. Training has two stages: joint optimization of the VGM and reconstruction network for approximately 250K steps, followed by approximately 90K steps with the VGM frozen. Training begins with four-view inputs; the final run of the frozen-VGM stage mixes two, four and six input views and samples multiple anchor budgets for each view count. In the first stage, the VGM uses one tenth of the learning rate of the reconstruction network. The final part of the frozen-VGM stage uses a filtered subset of the training samples. Ablation variants (selection rule, scale rule, offset bound,

group size K and wide-frame input) are trained as separate models under the same training schedule as AllocSplat.

We use AdamW with bfloat16 mixed precision on eight H100 GPUs and optimize $\mathcal{L} = \mathcal{L}_1 + 0.05\mathcal{L}_{\text{LPIPS}}$ on rendered target views. The final run of the frozen-VGM stage uses cosine learning-rate decay from 5×10^{-5} to 10^{-6}.

Backbone pre-training provenance. DA3 lists DL3DV among the training sources for its geometry model (Lin et al., 2026). Its reported exclusion of the 140 benchmark scenes applies to its separately trained NVS model. Overlap between these scenes and the pre-training data of the DA3-Giant-1.1 geometry model used here remains unverified.

Scale controls. For a selected candidate originating in input view v, let z_i be its depth and $f_{x,v}$ the horizontal focal length in pixels on the model input canvas. The one-pixel footprint in 3D is $h_i = z_i/f_{x,v}$. A fixed footprint of w pixels sets $\rho_i = \text{clip}(wh_i, \rho_{\min}, \rho_{\max})$. The fixed variant uses $w = 18$. The global-budget variant uses $w(M) = 18(1024/M)^{0.67}$ with the same clipping, which gives about 46/29/18/15 pixels at $M = 256/512/1024/1296$. Before clipping, these rules hold the projected footprint fixed within a budget while allowing its 3D size to depend on depth. The local-adaptive rule instead computes ρ_i from selected-neighbor spacing.

A.2 Input Protocols and Budgets

The main DL3DV comparison uses the existing per-scene DepthSplat input indices, which also match the supplied TokenGS and ZipSplat evaluation indices. We use the released DL3DV models of DepthSplat (ViT-B, two scales), YoNoSplat and TokenGS, and the MVSplat DL3DV model released with DepthSplat (Xu et al., 2025). ZipSplat uses its official model trained jointly on DL3DV and RealEstate10K. All methods receive ground-truth poses and intrinsics, with pose alignment disabled for YoNoSplat and ZipSplat. The released index lists all 50 sequence frames as targets. Standard DL3DV quantitative results exclude the inputs and score the remaining $50 - V$ frames. AllocSplat and ZipSplat predict from 252×252 inputs, DepthSplat, MVSplat and TokenGS from 256×256, and YoNoSplat from 224×224; their unchanged Gaussians are rendered onto the common 252×252 target canvas. In both RealEstate10K tables ZipSplat runs at its full token budget, $324V$ tokens for V input views. ZipSplat decodes 32 Gaussians per cluster. For the matched-count DL3DV comparison, we set 288/576/1,152 clusters, giving exactly the 9,216/18,432/36,864 Gaussians that AllocSplat emits at $M = 256/512/1024$.

Image and target-camera construction. All DL3DV methods start from the same 540×960 source frames, resize the short side and center-crop to their native square input sizes. TokenGS scales camera translations by 0.15. The target camera uses resolution-independent normalized intrinsics; TokenGS's pixel-space intrinsics are converted from 256 to 252 pixels by the factor $252/256$. The default crop applies no half-pixel principal-point correction for odd crop remainders. Ground truth comes from the standard-protocol AllocSplat evaluation harness, using PIL bilinear resizing followed by center cropping. These images are hard-linked into each baseline's ground-truth directory and checked by hash for byte equality. The predicted Gaussians are re-rendered unchanged on the common target camera; four-view scores use the 46 non-input frames.

The larger-view study extends the released indices as described in App. C.7; RealEstate10K and Mip-NeRF 360 use their separately specified protocols.

The common-target study retains these context indices and evaluates only frames outside their union. Because the released contexts nest, that union is the six-view context, leaving 44 identical target frames for all three view counts in every scene.

The 18×18 evidence grids give $N = 324V$ candidates, so feasible requests satisfy $1 \le M \le N$. The full-pool rows of Table 1 use $M = N = 648/1296/1944$ at two, four and six views; only M differs from the default rows.

A.3 Metric Pipelines

We report PSNR, SSIM and VGG-LPIPS. Metrics are computed per frame, averaged within each scene, and then averaged equally across scenes. Reconstruction studies score 8-bit prediction–target

Table 6: **Four-view budget sweep under the standard DL3DV protocol.** Rows differ only in the anchor budget M set at inference and use the same 46 novel-view targets per scene as Table 1. $M = 1024$ is the default budget and $M = 1296$ uses the full four-view candidate pool.

AllocSplat (4 views)	#GS	PSNR↑	SSIM↑	LPIPS↓
$M = 256$	9,216	20.64	0.627	0.365
$M = 512$	18,432	22.30	0.707	0.274
$M = 768$	27,648	23.26	0.752	0.226
$M = 1024$	36,864	23.87	0.779	0.199
$M = 1296$	46,656	**24.22**	**0.794**	**0.185**

pairs (PNG) on the specified common canvas with fused SSIM, except the re-decoding analysis in Table 3, which scores render tensors directly (float) with skimage-style SSIM. VGG-LPIPS uses normalized image inputs. All methods within each study use the same metric pipeline. Paired comparisons use the same scene set, with statistical settings for the common-target study given in App. A.4.

Each method spends a different number of parameters per Gaussian: 14 for AllocSplat, TokenGS and YoNoSplat, 23 for ZipSplat, and 38 for DepthSplat and MVSplat. At an equal primitive count the methods with the larger per-Gaussian budget therefore store more scalar attributes per primitive.

A.4 Paired Statistical Comparisons

Cross-view comparisons pair scene scores at fixed M; budget comparisons pair scores at fixed V. The common-target study uses percentile bootstrap intervals from 10,000 resamples of the 140 paired scene differences, with random seed 0. These intervals quantify variation across test scenes, not across training runs.

A.5 Ordered Prefixes and Local Reconstruction

Any rule whose per-step criterion reads only the centers chosen so far, including spatial farthest-point sampling, produces one sequence that all budgets truncate. On 140 four-view scenes, the joint and spatial FPS rules both reproduce their M=1024 selection exactly at M=128/256/512, while per-budget clustering agrees on no scene. Both rules rank the first block by q_j and differ from the second block onward (Sec. 4.4). Retaining the first M centers of a larger allocation and recomputing their scales and contextualized decoding reproduces direct inference at M exactly, on 24 scenes at $M = 256/512/1024$ and on five scenes at 11 budgets that include partial blocks. Table 3 separately quantifies re-decoding against retaining previously decoded groups on all 140 scenes.

B Extended Capacity and Selection Studies

B.1 Input Views and Capacity: Full Results

Standard-protocol budget sweep. Table 6 extends the four-view comparison in the main table from $M = 256$ to the full candidate pool, with the same weights and input indices. PSNR, SSIM and LPIPS improve at each tested increment: going from 256 to 512 anchors adds 1.66 dB, and going from 1,024 to 1,296 adds 0.35 dB. Figure 4 shows a fixed target region across budgets: additional centers refine local boundaries and thin structures without changing the input images.

Common held-out targets across view counts. Input-view count and anchor count control different resources: the available observations and the capacity used to represent them. Comparing across view counts needs a common target set, since each released index reserves different frames as context; removing the union of the three leaves 44 frames per scene that all of them score. On those frames we sweep $V \in \{2, 4, 6\}$ and $M \in \{256, 384, 512, 640\}$ (Tab. 7); the 12 settings change only V and M at inference. The sweep includes both trained view–budget pairs and intermediate budgets.

Increasing M improves all three mean metrics at each fixed view count. For four views, successive budget increments add 0.97, 0.69 and 0.54 dB, showing diminishing returns over this range. The

Table 7: Input views and output capacity on DL3DV-140 with 44 common held-out targets per scene, obtained by removing the union of the released two-, four- and six-view context indices, so all three columns are scored on one identical target set. Each cell reports PSNR↑ / SSIM↑ / LPIPS↓. Settings differ only in V and M; each anchor decodes 36 Gaussians.

Anchors M	2 views	4 views	6 views
256	19.716 / 0.6013 / 0.3776	20.661 / 0.6270 / 0.3648	20.679 / 0.6244 / 0.3702
384	20.406 / 0.6399 / 0.3300	21.635 / 0.6737 / 0.3113	21.703 / 0.6703 / 0.3164
512	20.819 / 0.6644 / 0.3016	22.329 / 0.7076 / 0.2732	22.473 / 0.7053 / 0.2790
640	21.093 / 0.6784 / 0.2857	22.865 / 0.7331 / 0.2453	23.044 / 0.7307 / 0.2511

benefit of additional input views depends on the output budget: the PSNR gain from two to four views grows from 0.944 dB at $M = 256$ to 1.772 dB at $M = 640$, with paired 95% confidence intervals $[0.811, 1.079]$ and $[1.590, 1.950]$ dB.

Adding the fifth and sixth views gives a smaller PSNR gain. At $M = 256$ they add 0.018 dB over four views, an interval of $[-0.043, 0.078]$ that does not exclude zero, and only as the budget grows does the gain separate from it: 0.068 dB at 384, 0.143 at 512 and 0.179 at 640, the last with interval $[0.096, 0.263]$. These gains are specific to PSNR: SSIM and LPIPS favor four views at all four budgets. Additional observations and output capacity therefore have different, metric-dependent effects.

B.2 Selection Controls and Group Size

Selection rules. Table 8 expands the PSNR summary in Tab. 2 with SSIM and LPIPS for the same four rules. Joint selection gives the highest PSNR and SSIM at every tested budget and the lowest LPIPS at $M = 256/512$; at $M = 1024$, FPS and Random have lower LPIPS (0.193 and 0.197 against 0.199).

Table 8: Selection-rule ablation on DL3DV-140 with four input views and $K = 36$; each row is trained with the listed selection rule. Bold marks the best metric within each budget.

Anchors M	Selection rule	PSNR↑	SSIM↑	LPIPS↓
256	Joint geometry–appearance	**20.641**	**0.627**	**0.365**
	Random	19.436	0.587	0.407
	Spatial FPS	18.909	0.569	0.418
	Position-space k-means++	18.419	0.563	0.436
512	Joint geometry–appearance	**22.297**	**0.707**	**0.274**
	Random	21.402	0.688	0.280
	Spatial FPS	21.577	0.680	0.293
	Position-space k-means++	20.006	0.642	0.348
1024	Joint geometry–appearance	**23.870**	**0.779**	0.199
	Random	22.960	0.776	0.197
	Spatial FPS	23.728	0.777	**0.193**
	Position-space k-means++	22.955	0.752	0.227

The two capacity axes. A Gaussian budget can be distributed over more anchors or larger groups per anchor. At matched primitive counts, $K = 16$ outperforms $K = 36$ on all three metrics at the seven tested operating points, with a PSNR advantage of 0.60–1.34 dB (Tab. 9a). At fixed anchor counts, $K = 36$ is best among the three tested group sizes, including $K = 64$, which emits $1.78\times$ more primitives (Tab. 9b).

The choice of K therefore changes both efficiency at a given count and the maximum capacity available from the candidate pool. At four views, $K = 16$ reaches at most 20,736 Gaussians and 23.340 dB. With the same pool, $K = 36$ reaches 23.870 dB at the default 36,864 Gaussians and 24.224 dB at the full 46,656.

Table 9: **Group size under two capacity controls.** (a) Fixed Gaussian count: M varies with K to keep KM fixed; smaller K allocates more centers. The pool is $324V$ candidates regardless of K. (b) Fixed center count: four-view comparisons at equal M, so Gaussian count scales with K. Both panels use 140 scenes and the $50 - V$ non-input frames; cells report PSNR↑ / SSIM↑ / LPIPS↓.

(a) Fixed Gaussian count

Views	#GS	K=16	K=36 (ours)	K=64
2	9,216	**20.339 / 0.637 / 0.345**	19.656 / 0.599 / 0.379	18.535 / 0.534 / 0.465
2	10,368	**20.448 / 0.644 / 0.338**	19.853 / 0.610 / 0.365	18.724 / 0.544 / 0.451
4	9,216	**21.843 / 0.682 / 0.313**	20.641 / 0.627 / 0.365	19.069 / 0.547 / 0.461
4	18,432	**23.203 / 0.750 / 0.238**	22.297 / 0.707 / 0.274	20.587 / 0.621 / 0.371
4	20,736	**23.340 / 0.757 / 0.231**	22.585 / 0.721 / 0.259	20.869 / 0.635 / 0.355
6	9,216	**22.018 / 0.680 / 0.319**	20.679 / 0.624 / 0.370	19.040 / 0.544 / 0.467
6	18,432	**23.642 / 0.757 / 0.233**	22.473 / 0.705 / 0.279	20.687 / 0.619 / 0.375

(b) Fixed center count

Centers M	K=16	K=36 (ours)	K=64
256	20.038 / 0.590 / 0.424 4,096 GS	**20.641 / 0.627 / 0.365** 9,216 GS	20.339 / 0.609 / 0.386 16,384 GS
512	21.572 / 0.668 / 0.329 8,192 GS	**22.297 / 0.707 / 0.274** 18,432 GS	21.919 / 0.689 / 0.295 32,768 GS
1,024	23.021 / 0.741 / 0.247 16,384 GS	**23.870 / 0.779 / 0.199** 36,864 GS	23.413 / 0.762 / 0.219 65,536 GS

C Additional Analyses

C.1 Bounding Offsets as Well as Sizes

The local spatial scale enters the Gaussian decoder twice: it bounds how far each Gaussian may be offset from its anchor, and it caps the Gaussian's scale. To ask whether the positions need to follow the local density or only the sizes do, we replace the bound on offsets with the constant 0.5, the upper clamp of the local scale, so that the Gaussians can reach at least as far as in the full model while their sizes stay locally bounded. The variant is evaluated under the protocol of Tab. 4. It has lower PSNR and SSIM and higher LPIPS at every tested budget (Tab. 10).

Table 10: Offset bound on DL3DV-140 with four input views. Both rows cap Gaussian sizes by the local spatial scale; the second replaces the bound on offsets with a budget-independent constant. Cells report PSNR↑ / SSIM↑ / LPIPS↓. The last row gives the scenes on which the full model has the higher PSNR and the 95% paired bootstrap interval of the PSNR difference in dB.

Offset bound	M=256	M=512	M=1024	M=1296
Local scale	**20.641 / 0.627 / 0.365**	**22.297 / 0.707 / 0.274**	**23.870 / 0.779 / 0.199**	**24.224 / 0.794 / 0.185**
Constant 0.5	20.599 / 0.625 / 0.368	22.228 / 0.704 / 0.278	23.795 / 0.776 / 0.203	24.151 / 0.791 / 0.189
Scenes, interval	90/140, [0.025, 0.060]	109/140, [0.054, 0.086]	130/140, [0.065, 0.087]	126/140, [0.063, 0.084]

C.2 Test-Time Optimization

To evaluate per-scene refinement, we freeze the network, the anchor positions and the local spatial scales, and optimize the 256-dimensional anchor content latents against the context images, which are disjoint from the target views, using AdamW with a learning rate of 10^{-3}. We run TokenGS's released Gaussian-token optimization procedure unchanged on the same input index, with its network weights frozen.

Table 11 sweeps the number of refinement steps. Both methods gain more as context views accumulate. For AllocSplat, 20 steps recover 72–86% of the PSNR gain at 100 steps, and 50 steps recover approximately 92% or more, slightly exceeding the 100-step result at two views. At the matched 50 steps reported in Tab. 1, AllocSplat gains 0.53/1.81/2.70 dB against TokenGS's 0.07/1.18/1.82, and it is 1.67/1.63/2.05 dB higher in absolute terms.

Table 11: Test-time optimization on DL3DV-140, on the released DepthSplat index and scored on the $50 - V$ frames that are not inputs, as in Tab. 1. Network weights are frozen throughout; AllocSplat optimizes anchor latents and TokenGS its Gaussian tokens. Cells report PSNR/SSIM/LPIPS.

Method	Steps	2 views	4 views	6 views
TokenGS	0	19.521 / 0.614 / 0.430	22.877 / 0.745 / 0.303	23.986 / 0.782 / 0.274
	20	19.556 / 0.618 / 0.414	23.768 / 0.774 / 0.278	25.320 / 0.821 / 0.244
	50	19.589 / 0.620 / 0.405	24.053 / 0.783 / 0.264	25.804 / 0.833 / 0.228
AllocSplat	0	20.729 / 0.660 / 0.304	23.870 / 0.779 / 0.199	25.158 / 0.818 / 0.165
	20	21.183 / 0.682 / 0.272	25.361 / 0.825 / 0.150	27.289 / 0.869 / 0.112
	50	**21.262** / **0.685** / 0.266	25.685 / 0.834 / 0.142	27.858 / 0.881 / 0.102
	100	21.255 / 0.684 / **0.265**	**25.796** / **0.837** / **0.139**	**28.098** / **0.885** / **0.099**

Budgets: $M = 512/1024/1536$ at 2/4/6 views for AllocSplat; TokenGS uses 4,096 tokens throughout. All cells are re-scored from dumped PNGs with the same metric pipeline as the main tables.

C.3 Generalization to RealEstate10K

Tab. 12 gives the two-view comparison summarized in the main text, on the pixelSplat index, which places both context views at the ends of the clip. It compares the DL3DV-trained AllocSplat with two references: the other methods run from their own DL3DV weights, and the RealEstate10K models those methods release, which were trained on the evaluation dataset. To see whether the allocation still generalizes when more observations are available, we repeat the experiment on the DepthSplat index at two, four and six views. That index is a video index over 1,600 scenes common to all three view counts; we evaluate a fixed subset, drawn by shuffling those scenes with a seed and taking the first 200, with the same subset reused at every view count; one of them is absent from the DepthSplat renders at all three view counts, so all methods are scored on the remaining 199. As in the main text, every method runs from DL3DV weights where they exist and keeps the center-crop convention. This index reserves every non-context frame as a target, between 200 and 279 per scene, and all of them are scored. AllocSplat uses its default budget for each view count (Tab. 5). Two baselines in that table have seen RealEstate10K: DepthSplat's released DL3DV weights were pre-trained on it, and ZipSplat ships a single model trained on both datasets.

Table 12: **RealEstate10K, public pixelSplat index.** All 6,474 test scenes, two input views, GT cameras, no test-time optimization, the same ground truth for every row. The upper block uses weights trained with RealEstate10K; the lower block uses DL3DV weights. ZipSplat uses a single model trained jointly on DL3DV and RealEstate10K. *DepthSplat's released DL3DV weights include RealEstate10K pre-training. Bold marks the best entry within each block. #GS follows the pixel-aligned convention of Tab. 1.

Method	#GS	PSNR↑	SSIM↑	LPIPS↓
RealEstate10K in training				
DepthSplat (Xu et al., 2025)	131 K	**27.774**	**0.899**	**0.111**
MVSplat (Chen et al., 2024b)	131 K	26.821	0.882	0.123
YoNoSplat (Ye et al., 2026)	100 K	26.097	0.865	0.136
ZipSplat (Veicht et al., 2026)	21 K	22.745	0.736	0.203
DL3DV weights, evaluated on RealEstate10K				
DepthSplat* (Xu et al., 2025)	131 K	25.220	0.868	**0.141**
TokenGS (Ren et al., 2026)	262 K	25.550	0.854	0.202
YoNoSplat (Ye et al., 2026)	100 K	25.632	0.855	0.147
AllocSplat (M=512)	**18 K**	**26.507**	**0.873**	0.147

C.4 Generalization to Mip-NeRF 360

Fig. 5 carries the same DL3DV weights to the nine Mip-NeRF 360 scenes (Barron et al., 2022), with six posed input views and eight held-out targets per scene shared by every method, and no test-time optimization. Over the 72 targets, AllocSplat reaches 21.62 dB from 55K Gaussians, against 20.95 for DepthSplat and 16.85 for MVSplat at 393K and 20.87 for YoNoSplat at 301K.

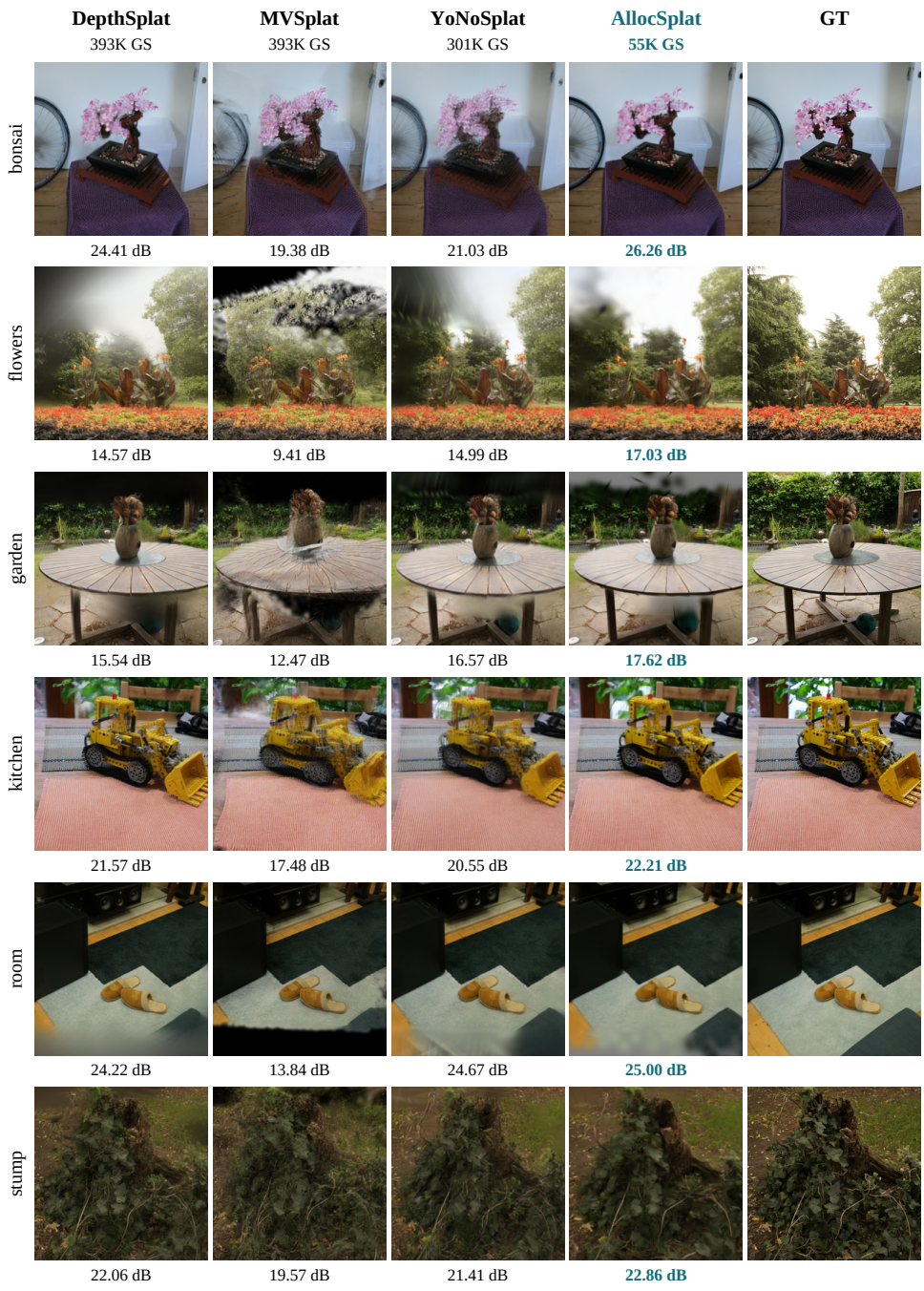

Figure 5: **Generalization to Mip-NeRF 360** with DL3DV weights, six posed input views, no test-time optimization. Every method is rendered on a common 252×252 canvas and scored against one ground truth. Labels below the images give PSNR, with the best in each row in bold; headers give the Gaussian count. Views are held out, 2 to 38 frames from the nearest input.

C.5 Evaluating on the Uncropped Frame

The main comparison uses the central square of each DL3DV frame, retaining 56.25% of the width of a 16:9 image. We also evaluate the same scenes on an uncropped 252×448 canvas (Tab. 13). DepthSplat and TokenGS use their released 256×448 DL3DV models. AllocSplat uses a separate variant trained on 252×448 inputs, with 576 candidates per view. Its full-pool result exceeds TokenGS by 1.38/1.48/1.76 dB at 2/4/6 views.

The square-input pipelines used for ZipSplat and YoNoSplat do not reconstruct the full width under this evaluation setup. For example, ZipSplat reaches 16.11 dB at four views with empty bands near the image edges, compared with 22.35 dB on the square canvas.

Table 13: **The same comparison on the uncropped 16:9 frame.** Scored on the same 140 scenes, the same released indices and the same $50 - V$ non-input frames as Tab. 1. DepthSplat and TokenGS use their released 256×448 models. AllocSplat uses a separate variant trained at 252×448 (App. C.5), and its candidate pool grows to 576 per view. The full pool is $M = 1152/2304/3456$. Cells report PSNR↑ / SSIM↑ / LPIPS↓.

Method	#GS (2/4/6)	2 views	4 views	6 views
DepthSplat (Xu et al., 2025)	229/459/688 K	18.42 / 0.572 / 0.323	20.45 / 0.661 / **0.215**	20.89 / 0.678 / **0.192**
TokenGS (Ren et al., 2026)	262/262/262 K	19.00 / 0.600 / 0.435	20.88 / 0.674 / 0.343	21.34 / 0.690 / 0.324
AllocSplat (default)	18/37/55 K	19.59 / 0.605 / 0.382	21.59 / 0.682 / 0.300	22.36 / 0.711 / 0.268
AllocSplat (full pool)	41/83/124 K	**20.38 / 0.651 / 0.318**	**22.36 / 0.725** / 0.244	**23.10 / 0.751** / 0.221

C.6 Inference Cost

Table 14 resolves one forward pass into stages. Across the measured budgets, the geometry backbone takes about 51 ms and peak memory stays at 7.3–7.4 GiB. The budget-dependent work grows with M, dominated by two sequential loops: the allocation advances one block of four candidates at a time and refreshes the residuals in between, and the prior builds tokens one anchor at a time. We implement the allocation loop as a CUDA-based operator, which accelerates that stage by 5.7–$6.1\times$; the prior runs without it and becomes the largest budget-dependent stage at 281 ms for the full pool.

The profile uses the $K = 36$ architecture with 1,296 four-view candidates. Measurements use an H100 80GB with bf16 autocast and a probe list of 80 four-view scenes: 20 for warmup and 60 for timing. Inputs are staged on CPU and copied to GPU before each timed region; model construction, disk I/O and these copies are excluded. GPU synchronization brackets each measured stage. The timed pipeline includes DA3, candidate construction, allocation, local-scale computation, prior and token construction, anchor/latent processing, Gaussian decoding, and rasterization of four 252×252 target views. Rasterization takes about 1 ms.

At $M = 1024$, the resulting representation contains 36.9K Gaussians and 0.52 M scalar attributes, compared with 262K Gaussians and 3.67 M attributes for TokenGS or 9.96 M for DepthSplat. These are output-representation counts; they exclude network weights.

C.7 Input-View Extrapolation

Figure 6 extends evaluation to as many as 16 input views, beyond the mixed 2/4/6-view training range, with 256 anchors allocated per view. Both input evidence and output capacity increase in this experiment. It complements the fixed-budget study in App. B.1, which separates these two resources.

The released index supplies contexts only up to six views, and its contexts nest: on all 140 scenes the two-view context is contained in the four-view one and that in the six-view one. We therefore keep the released contexts at two, four and six views and build the larger sets on top of the six-view one, adding at each step the frame whose index is farthest from those already selected, so that the six sets nest all the way to sixteen. Every view count is scored on the same 34 frames, which no set uses as input. The target set is fixed while both the number of input views and the anchor budget increase along the curve.

Table 14: **AllocSplat architecture profiling at four input views.** $K = 36$, H100 80GB, bf16 autocast, 20 warmup and 60 timed probe scenes. Timings include four 252×252 target renders, with GPU synchronization at measurement boundaries. Model construction, disk I/O and host-to-device copies are excluded. Only the allocation loop uses the CUDA-based operator. Stage columns report separately measured medians; the total column reports the end-to-end median (App. C.6).

		stage median (ms)				
Method / budget M	#GS	geometry	allocation	tokens	decode + render	total (ms)
AllocSplat, M=256	9.2 K	51.1	18.3	55.2	23.0	136.7
AllocSplat, M=512	18.4 K	51.3	35.0	110.5	32.2	219.9
AllocSplat, M=1024	36.9 K	51.0	68.4	222.2	50.5	382.3
AllocSplat, M=1296	46.7 K	51.0	86.2	280.9	62.6	470.2

Serving the four budgets for one scene with the CUDA-based operator costs 1,205 ms when each is inferred independently and 966 ms with candidate-order reuse, a $1.25\times$ speedup (19.8% less time). Without it, the paired run takes 2,318/1,431 ms ($1.62\times$). Without it, the single-budget medians for the four rows above are 240.2/412.1/745.1/928.8 ms.

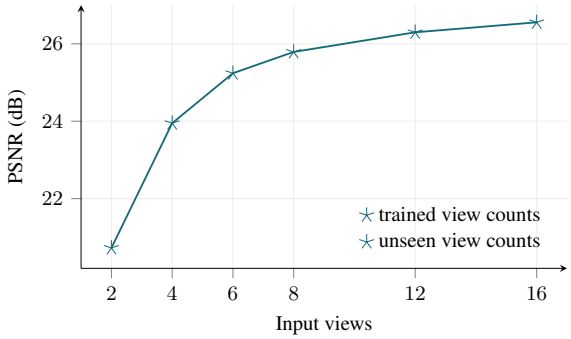

Figure 6: PSNR vs. input view count on DL3DV-140, on the 34 targets that no view count uses as input. Filled stars: view counts seen during training (2/4/6); open stars: unseen counts. The anchor budget follows the $\times 1.0$ rule (256 anchors per view), so capacity grows with the view count.

D Additional Qualitative Comparisons

Figures 7–12 provide qualitative reconstruction comparisons in twenty-nine examples, with two plates per input-view count. Each scene occupies a row of renderings followed by a row of per-pixel squared error. Within each row, all methods use the same input images and render the same camera view.

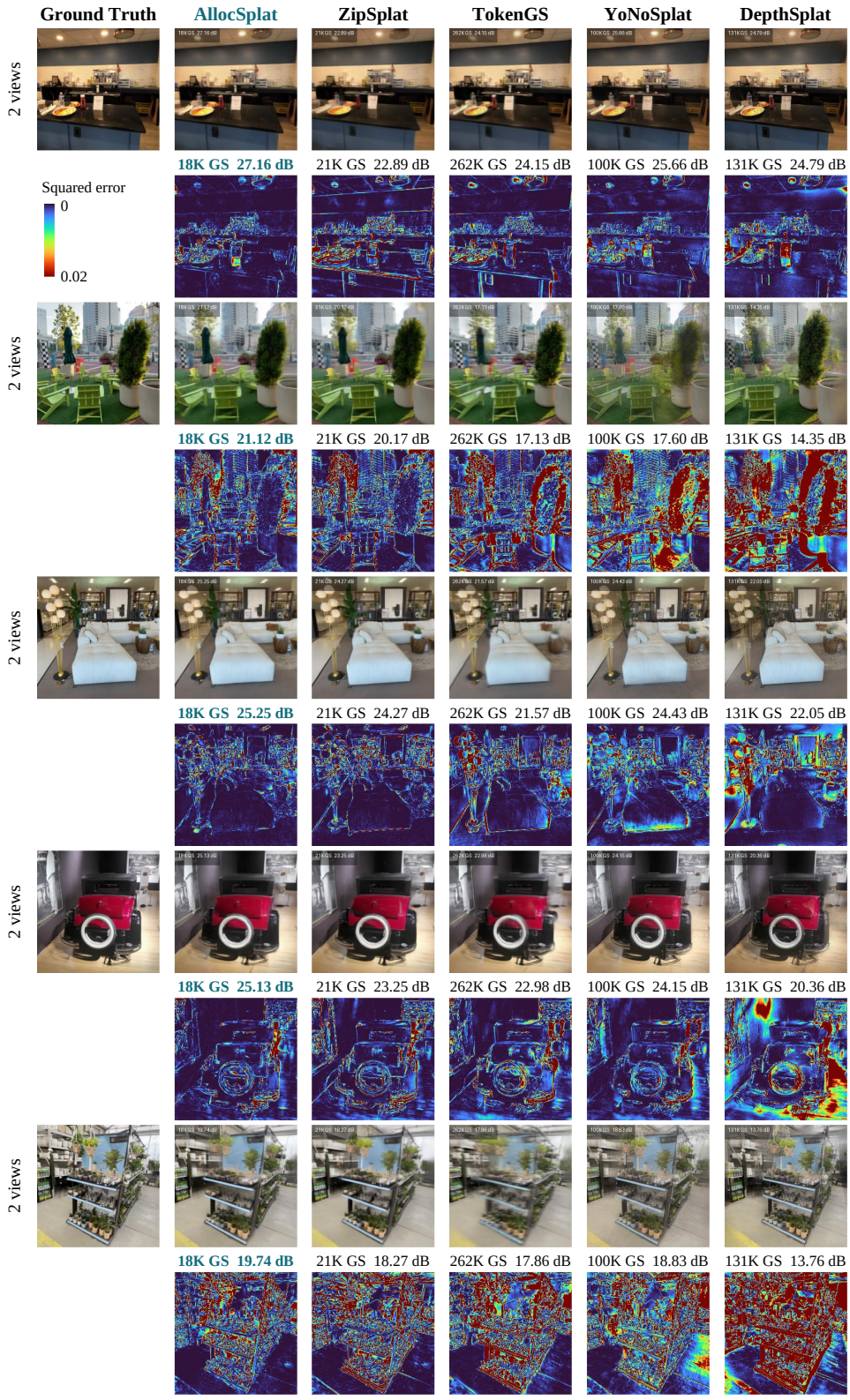

Figure 7: **Further DL3DV-140 qualitative comparisons at 2 input views.** 5 scenes; each occupies a row of renderings and a row of per-pixel squared error (turbo, clipped at 0.02). Labels between the rows give the Gaussian count and frame PSNR.

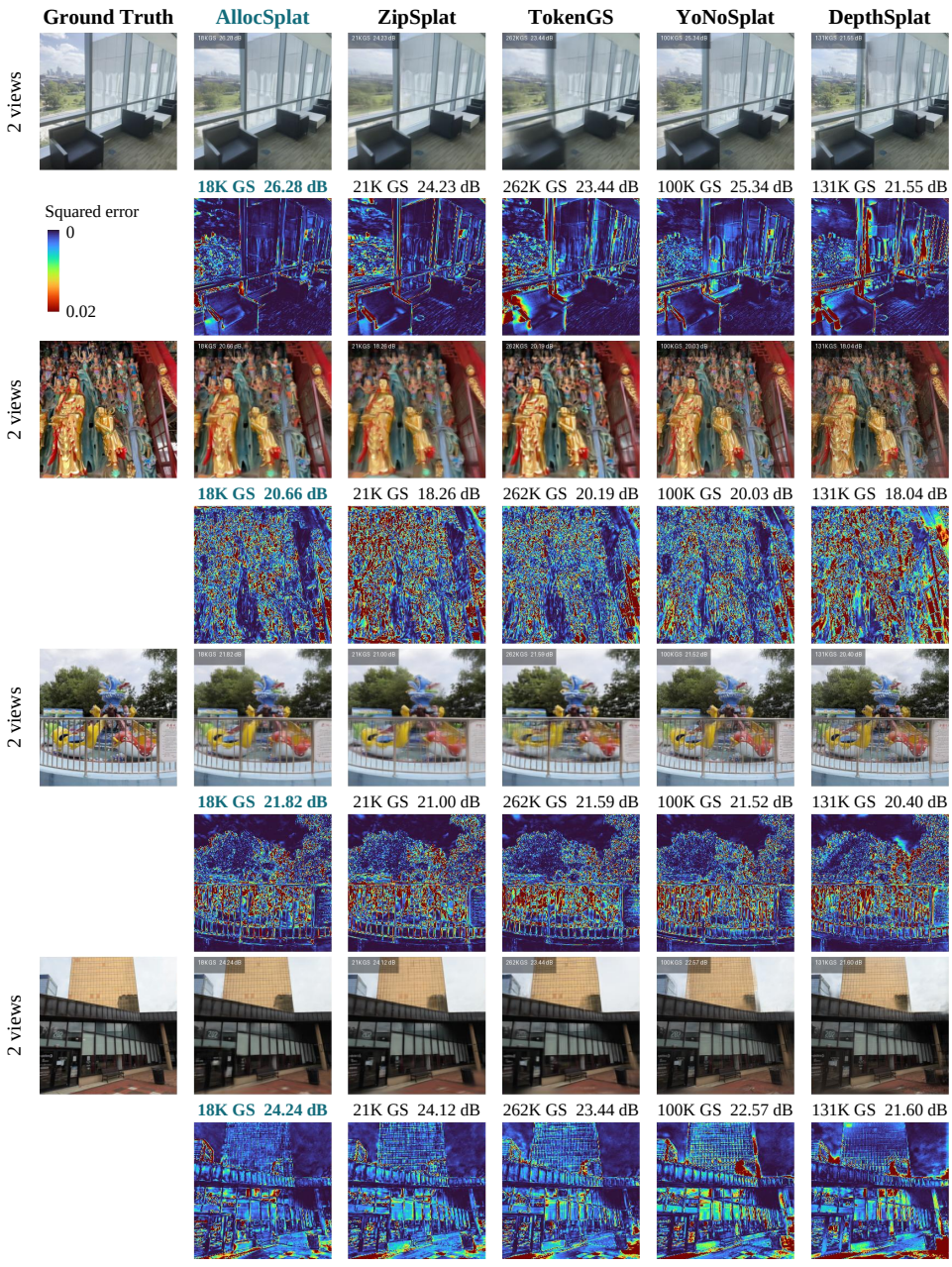

Figure 8: **Further DL3DV-140 qualitative comparisons at 2 input views.** 4 scenes; each occupies a row of renderings and a row of per-pixel squared error (turbo, clipped at 0.02). Labels between the rows give the Gaussian count and frame PSNR.

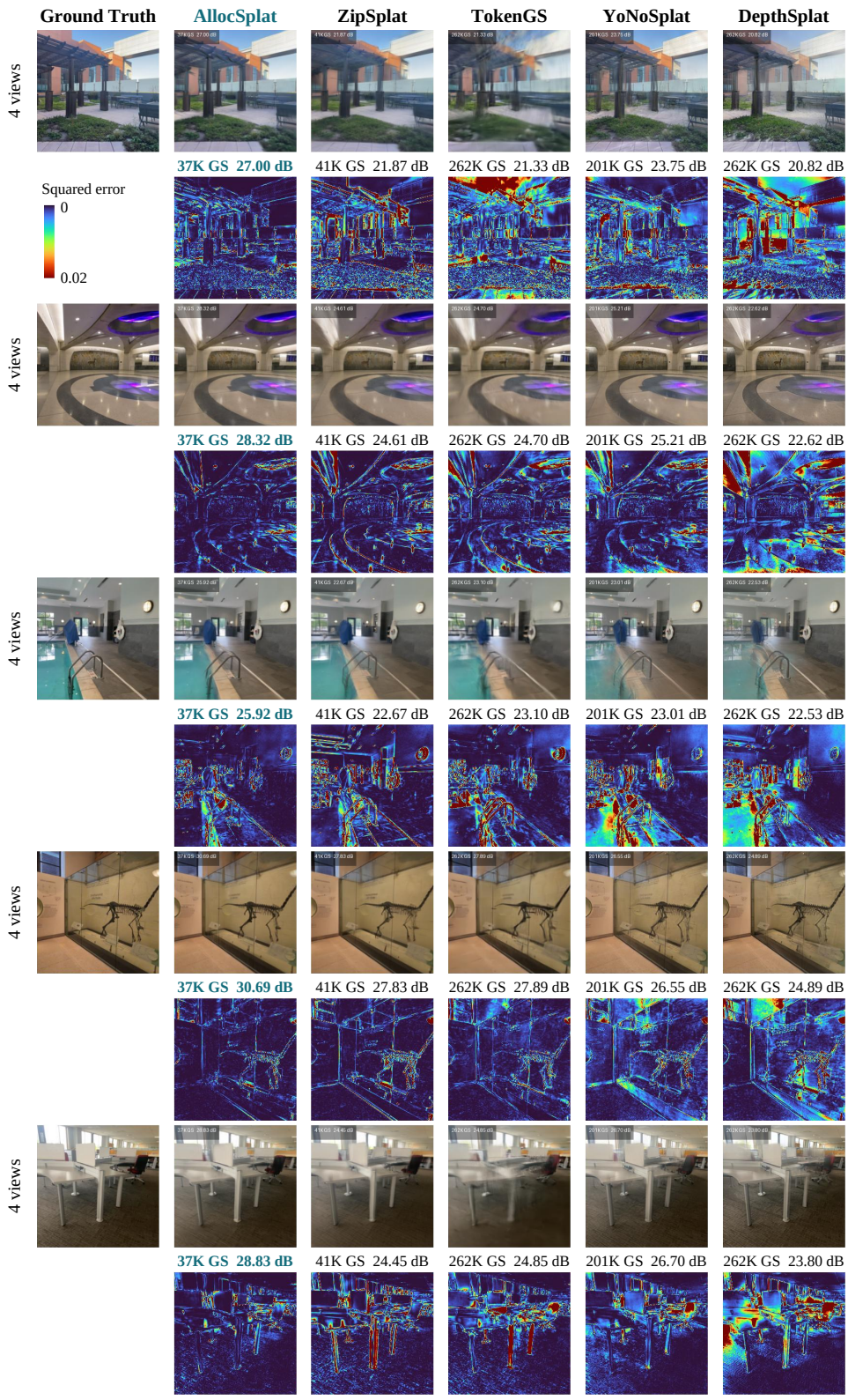

Figure 9: **Further DL3DV-140 qualitative comparisons at 4 input views.** 5 scenes; each occupies a row of renderings and a row of per-pixel squared error (turbo, clipped at 0.02). Labels between the rows give the Gaussian count and frame PSNR.

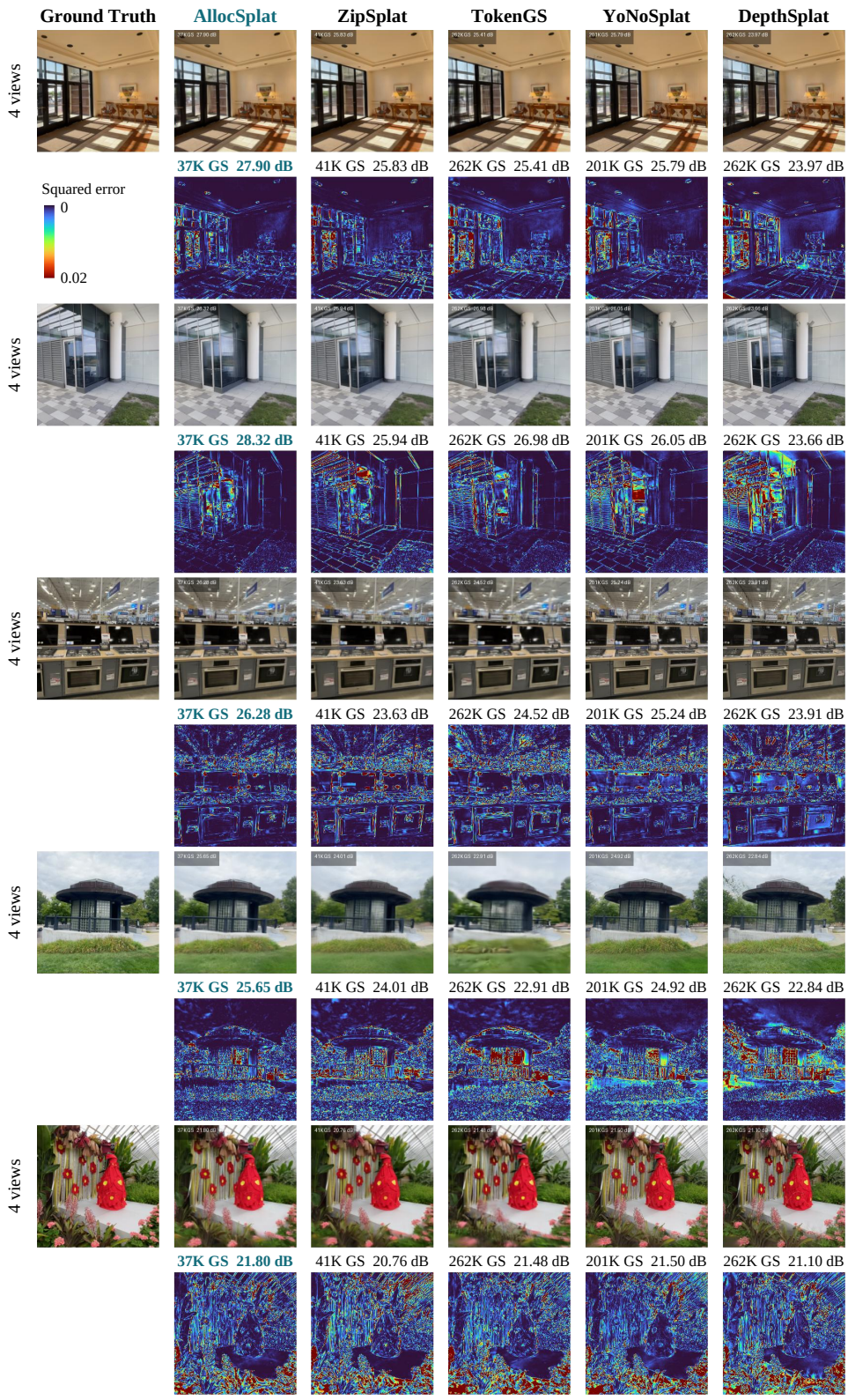

Figure 10: **Further DL3DV-140 qualitative comparisons at 4 input views.** 5 scenes; each occupies a row of renderings and a row of per-pixel squared error (turbo, clipped at 0.02). Labels between the rows give the Gaussian count and frame PSNR.



Figure 11: **Further DL3DV-140 qualitative comparisons at 6 input views.** 5 scenes; each occupies a row of renderings and a row of per-pixel squared error (turbo, clipped at 0.02). Labels between the rows give the Gaussian count and frame PSNR.

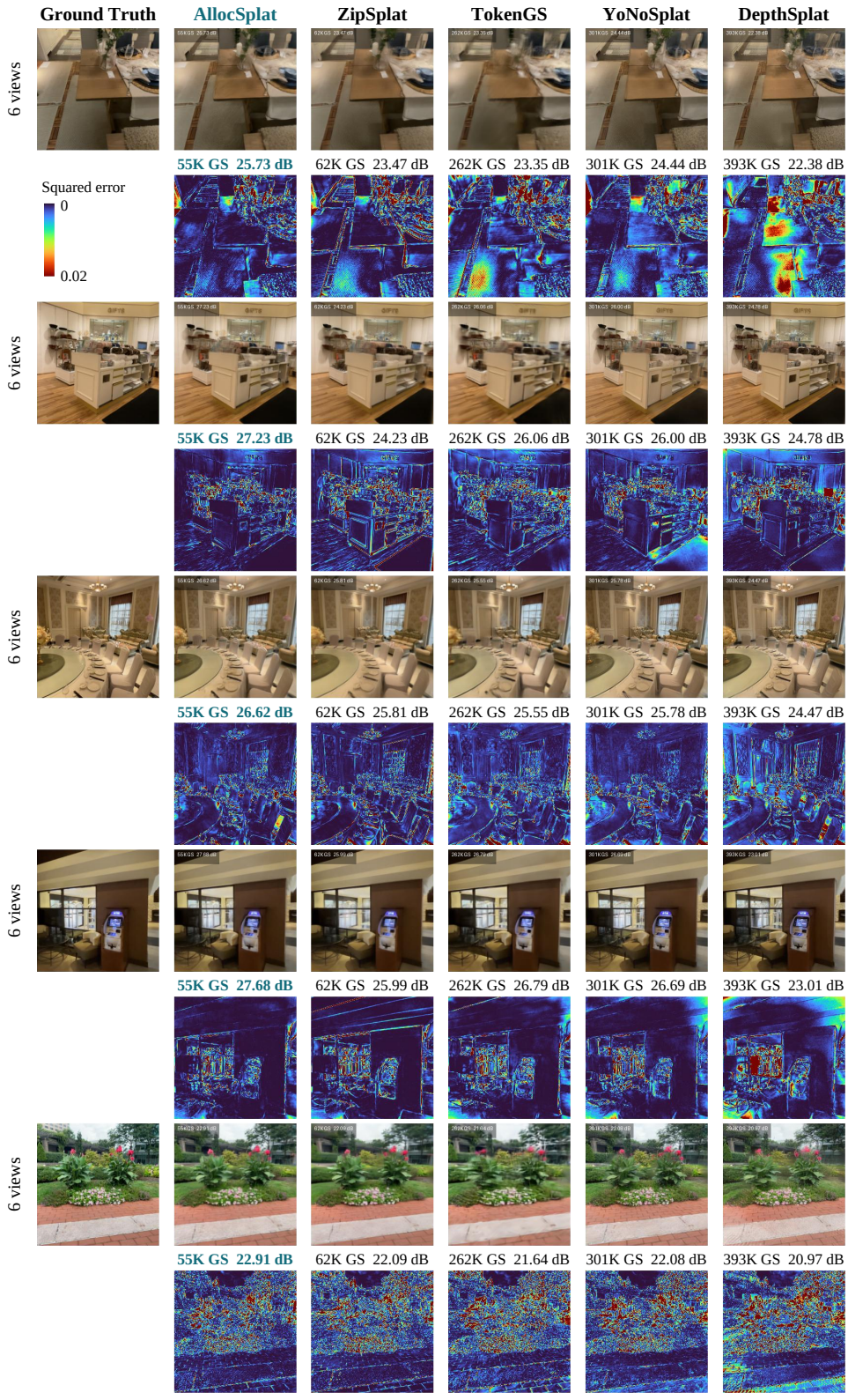

Figure 12: **Further DL3DV-140 qualitative comparisons at 6 input views.** 5 scenes; each occupies a row of renderings and a row of per-pixel squared error (turbo, clipped at 0.02). Labels between the rows give the Gaussian count and frame PSNR.