LoCoSplat: Real-Time Feed-Forward 3D Gaussian Splatting with Minimal 3D Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Feed-forward 3D Gaussian Splatting (3DGS) increasingly aggregates multi-view evidence with heavy learned 3D networks. We propose LoCoSplat (**Lo**cal-**Co**ntext **Splat**ting), motivated by the observation that a Gaussian is a local primitive: once depth is predicted, what the 3D stage must add (scale, rotation, opacity) depends on the point cloud around each anchor, and a fixed local average of that neighbourhood is enough to supply it, no heavy network required. LoCoSplat realises exactly this average: it splats a 16-d linear projection of the point features into a fine and a coarse grid and reads both back at each anchor with a 0.14M-parameter pointwise MLP; with no learned 3D network and no dynamic sparse computation, its whole encoder runs as one fp16 CUDA graph. On RealEstate10K, LoCoSplat outperforms every prior feed-forward method on PSNR, SSIM, and LPIPS at 6, 12, and 24 views, with a margin that widens as views densify ($+3.3$ PSNR over VolSplat, the prior voxel-aligned state of the art, at 24 views) and grows further under zero-shot transfer to ACID and fine-tuning on ScanNet. It reconstructs a 6-view scene in 33 ms on one NVIDIA RTX PRO 6000 GPU, the fastest of seven feed-forward methods and $4.2\times$ faster than the previous state of the art, trains $2.7\times$ faster ($5.9\times$ at 24 views), and uses $6.7\times$ less inference memory.

1 Introduction

Feed-forward 3D Gaussian Splatting (Charatan et al., 2024; Chen et al., 2024; Xu et al., 2025) predicts the Gaussian primitives of 3DGS (Kerbl et al., 2023) from a handful of posed images in a single forward pass, replacing the per-scene optimization of NeRF and 3DGS (Mildenhall et al., 2020; Kerbl et al., 2023) with a learned prior. Pixel-aligned prediction, the dominant approach, suffers from multi-view misalignment and resolution-bound Gaussian budgets, and the field's answer has been to reason with ever-heavier learned modules, mostly by moving prediction into 3D: sparse 3D U-Nets and CNNs (Wang et al., 2026a; Zhang et al., 2022), 3D cost-volume networks, or long-sequence Mamba/transformer models over image tokens (Ziwen et al., 2025). VolSplat (Wang et al., 2026a), the state of the art of the depth-guided voxel-aligned family, predicts depth, voxelizes the unprojected pixels into anchors, and then *refines* the sparse volume with a 17M-parameter sparse 3D U-Net before decoding Gaussians. That U-Net is its largest inference cost ($\sim$7 scenes per second): a dynamic-shape sparse computation (Choy et al., 2019) that cannot be captured in a CUDA graph and barely benefits from mixed precision, and the dominant training memory as views grow (24-view training takes 159 hours). Behind this trajectory lies an implicit assumption: that aggregating multi-view evidence in 3D requires substantial learned 3D capacity.

Our motivation is a local view of the 3D stage: what a Gaussian needs from its neighbours is the kind of low-order local information that a grid computes for free, not a learned 3D network. LoCoSplat (**Lo**cal-**Co**ntext **Splat**ting, Fig. 1) keeps the depth-guided voxel-aligned pipeline (Section 2.1) up to the anchors: a hybrid monocular–multi-view depth branch and a DPT feature head place one anchor per occupied voxel. Its 3D stage is a fixed grid average: each anchor's feature, linearly projected to 16 dimensions, is averaged into a fine and a coarse grid, and both grids are read back at the anchor by a 0.14M-parameter pointwise MLP and a pointwise head. With no

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.

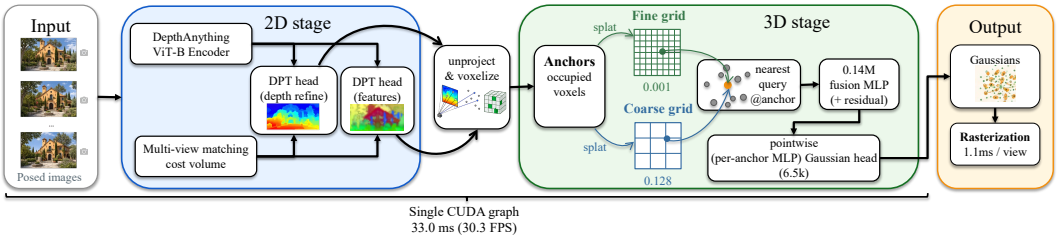

Figure 1: **LoCoSplat overview.** Predicted depth and per-pixel features are unprojected and voxelized into anchors; the 3D stage averages anchor features in a fine and a coarse grid and decodes each anchor pointwise. The whole encoder runs as one CUDA graph.

learned 3D network and no dynamic sparse computation, the whole encoder runs as one fp16 CUDA graph.

The motivation is structural: **a 3D Gaussian is a local primitive**. Once depth is predicted, each anchor's colour and position are determined by its own pixel. The 3D stage must add scale, rotation, and opacity, the layout of the surface around the anchor; no single pixel observes this layout, but it is determined by the neighbouring points, which a *fixed* average over a coarser neighbourhood, computed for free by a grid, can summarize. The anchors are also so sparse at the working voxel size (99.6% have no occupied neighbour at 6 views) that a 3^3 sparse convolution over them is already pointwise (Section 2.4), so any context must come from a coarser scale, which the coarse grid supplies. This intuition led to the design.

We then measure what the 3D stage of this pipeline actually needs (Section 3): very little. A fixed grid average read by a pointwise MLP of under 0.2M parameters beats the 17M-parameter U-Net by $+0.70$ PSNR (Table 9). LoCoSplat has the best PSNR, SSIM, and LPIPS of all feed-forward methods on RealEstate10K at 6, 12, and 24 views, with a margin that *grows* on ACID and ScanNet, and reconstructs a 6-view scene in **33.0 ms** (**30.3 FPS**) on one GPU, $4.2\times$ faster than VolSplat ($2.4\times$ at equal fp32 precision) with 85% less peak inference memory, and trains $2.7\times$ faster ($5.9\times$ at 24 views; Table 2).

Our contributions are:

- **A 3D stage with no learned 3D network,** motivated by the locality of a Gaussian, with controlled experiments showing that the 3D stage of this pipeline needs very little: a fixed grid average read by a pointwise MLP of under 0.2M parameters beats a 17M-parameter sparse 3D U-Net, and the minimal coarse context it keeps needs neither learning nor appearance content (Section 3.5).
- **LoCoSplat, a state-of-the-art real-time feed-forward 3DGS method,** leading at 6, 12, and 24 views, in and out of domain, whose fp16 CUDA-graph encoder reconstructs at 30 FPS on one GPU, $4.2\times$ faster than VolSplat with $6.7\times$ less inference memory and $2.7\times$ faster training (Section 3.3).
- **A predictive account of the dense-view collapse** of depth-guided reconstruction: an optimization failure of the depth branch governed by the per-candidate plane-sweep disparity; LoCoSplat does not collapse and leads VolSplat by $+3.3$ PSNR at 24 views (Section 3.5).

2 Method

LoCoSplat reconstructs a set of 3D Gaussians from V posed images in one forward pass (Fig. 1). Its 2D stage (Section 2.1) is a depth-guided pipeline in the VolSplat/DepthSplat family (Wang et al., 2026a; Xu et al., 2025): per-view depth and per-pixel features, unprojected and voxelized into anchors (Section 2.2). Its 3D stage (Section 2.3) has no learned 3D network: a fixed local average of the point features is read by a pointwise MLP, and a pointwise head decodes each anchor's Gaussian (Section 2.4); Section 2.5 gives the training objective.

2.1 Depth and Feature Prediction

The input is V images $\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3}$ with intrinsics $\boldsymbol{K}_i$ and camera-to-world poses $(\boldsymbol{R}_i, \boldsymbol{t}_i)$. A frozen DepthAnythingV2 ViT-B backbone (Yang et al., 2024) provides monocular features, mapped by a trainable linear *interface* from 768 to 384 channels (ViT-S width); a CNN with a local-window

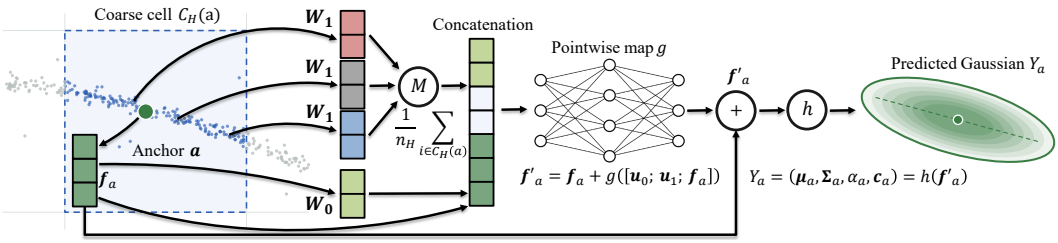

Figure 2: **Local context as a fixed average.** Points in the coarse cell $C_H(a)$ are projected by $\boldsymbol{W}_1$ and averaged, with no parameters, into $\boldsymbol{u}_1(a)$; the anchor's feature gives $\boldsymbol{u}_0(a) = \boldsymbol{W}_0 \boldsymbol{f}_a$. The pointwise map g fuses $[\boldsymbol{u}_0; \boldsymbol{u}_1; \boldsymbol{f}_a]$ residually and the head h decodes Y_a (Eqs. (3) to (5)).

cross-view transformer over the two nearest views (Xu et al., 2023) provides matching features, from which a plane-sweep cost volume over D=64 inverse-depth candidates $\{d_k\}$ is built. A depthwise-separable, half-width UniMatch-style regressor turns it into matching probabilities $p_i[k, \boldsymbol{p}]$ and the coarse inverse depth $\tilde{d}_i(\boldsymbol{p}) = \sum_k p_i[k, \boldsymbol{p}]\, d_k$; one DPT head (Yang et al., 2024) refines it with the monocular features into the full-resolution depth $\boldsymbol{D}_i$, and a second DPT head yields the per-pixel feature $\boldsymbol{f}_{i,\boldsymbol{p}} \in \mathbb{R}^{132}$ (image, depth, and peak matching probability appended) that enters the 3D stage.

2.2 Anchors

Every pixel is unprojected along its depth and voxelized at the fine voxel size v=0.001:

$$\boldsymbol{x}_{i,\boldsymbol{p}} = \boldsymbol{R}_i\, \boldsymbol{D}_i(\boldsymbol{p})\, \boldsymbol{K}_i^{-1} \begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix} + \boldsymbol{t}_i, \qquad c_v(\boldsymbol{x}) = \mathrm{rnd}(\boldsymbol{x}/v) \in \mathbb{Z}^3. \tag{1}$$

This gives the point cloud $P = \{(\boldsymbol{x}_i, \boldsymbol{f}_i)\}_{i \le n}$, $n = V{\cdot}H{\cdot}W$, and the anchor set $\mathcal{A} = \{c_v(\boldsymbol{x}_i)\}$ of occupied voxels, one Gaussian per anchor; anchor a carries the mean position $\boldsymbol{x}_a$ and mean feature $\boldsymbol{f}_a$ of its points. At $v = 0.001$ almost every occupied voxel holds a single point (Table 7) and at 6 views only 0.42% of anchors have an occupied 3^3 neighbour, so anchors are points and the fine lattice carries no neighbourhood structure: any context the 3D stage uses must come from a coarser scale.

2.3 Local-Context Aggregation

The role of the 3D stage. Write $Y_a = (\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a, \alpha_a, \boldsymbol{c}_a)$ for the Gaussian of anchor a. Its colour and position are functions of the anchor's own pixel, $(\boldsymbol{f}_a, \boldsymbol{x}_a)$. Scale, rotation, and opacity describe the surface *around* a, which no single pixel observes; they are what the 3D stage has to supply, and they depend on the layout of P around a. Since anchors are isolated at the fine voxel size (Section 2.2), this information has to be gathered at a coarser scale H, and the simplest fixed operator that does so is a local average. With $w_i(a) = K_H(\boldsymbol{x}_a, \boldsymbol{x}_i)$ for a kernel of bandwidth H and a linear projection $\boldsymbol{W}$ of the point features, splatting $\boldsymbol{W}\boldsymbol{f}_i$ into a grid of cell size H and reading the cell at a computes the Nadaraya–Watson estimate (Nadaraya, 1964; Watson, 1964) in its binned form,[1]

$$\boldsymbol{u}_H(a) = \frac{\sum_i w_i(a)\, \boldsymbol{W}\boldsymbol{f}_i}{\sum_i w_i(a)}. \tag{2}$$

Because $\boldsymbol{f}_i$ carries each point's depth and peak matching probability alongside its image features, $\boldsymbol{W}$ can route geometric information about the neighbourhood (such as its mean depth) into $\boldsymbol{u}_H$ as well as appearance. The motivation for LoCoSplat is that this fixed average, together with the anchor's own feature, gives a pointwise map enough context to predict the Gaussian without a learned 3D network (Fig. 2). Section 3.5 measures what this context is worth, whether it needs to be learned or to carry appearance (by replacing $\boldsymbol{u}_H$ with an untrained low-pass or with geometric statistics of the cell), and how well a learned 3D refiner trains in this pipeline.

[1] Binning here is a computational device, the same grid accumulation used by binned kernel estimators (Hall & Wand, 1996). That analysis concerns density estimation, and we do not rely on it for a guarantee on the regression estimate in Eq. (2); what the average is worth is measured empirically in Section 3.5.

Two grids and a pointwise map. LoCoSplat instantiates Eq. (2) with the box kernel at two scales, with a learned 16-d projection per scale, $\boldsymbol{W}_0\boldsymbol{f}_i$ and $\boldsymbol{W}_1\boldsymbol{f}_i$. The *fine* level is the base voxelization of Section 2.2 itself: its grid is the anchor lattice c_v, and $\boldsymbol{u}_0(a) = \boldsymbol{W}_0\boldsymbol{f}_a$ is the projection of the anchor's own mean feature. The *coarse* level is a grid of cell size $H = 0.128$ with its own cell index $c_H(\boldsymbol{x}) = \mathrm{rnd}(\boldsymbol{x}/H) \in \mathbb{Z}^3$, the same rounding rule as c_v so that cell centres sit at integer multiples of H; writing $n_H(c)$ for the number of points in coarse cell c,

$$\text{project \& splat:} \qquad \boldsymbol{f}_i \mapsto \boldsymbol{W}_1\boldsymbol{f}_i, \qquad \boldsymbol{u}_1(c) = \frac{1}{n_H(c)} \sum_{i:\, c_H(\boldsymbol{x}_i)=c} \boldsymbol{W}_1\boldsymbol{f}_i, \tag{3}$$

$$\text{query \& fuse:} \qquad \boldsymbol{u}_1(a) = \boldsymbol{u}_1\big(c_H(\boldsymbol{x}_a)\big), \qquad \boldsymbol{f}'_a = \boldsymbol{f}_a + g\big([\boldsymbol{u}_0(a); \boldsymbol{u}_1(a); \boldsymbol{f}_a]\big), \tag{4}$$

with $\boldsymbol{W}_0, \boldsymbol{W}_1 \in \mathbb{R}^{16\times 132}$, $g : \mathbb{R}^{164} \to \mathbb{R}^{256} \to \mathbb{R}^{256} \to \mathbb{R}^{132}$, and $\boldsymbol{u}_1(c) = \boldsymbol{0}$ for empty cells. The splat in Eq. (3) is $\boldsymbol{u}_H$ of Eq. (2) with $w_i(a) = \mathbf{1}[c_H(\boldsymbol{x}_i) = c_H(\boldsymbol{x}_a)]$. The fine level needs no splat: a nearest-cell splat of $\boldsymbol{W}_0\boldsymbol{f}_i$ at cell size v under the same rounding rule, read back at the anchor, returns exactly the per-voxel mean $\boldsymbol{W}_0\boldsymbol{f}_a$ (a linear map commutes with the mean, and each voxel's mean point lies inside its voxel), so $\boldsymbol{u}_0$ is taken from the base voxelization directly and only the coarse level is splatted. The aggregation has no parameters and no data-dependent shape; the 3D stage's only learned parameters are $(\boldsymbol{W}_0, \boldsymbol{W}_1)$ (4.3k) and g (0.14M).

2.4 Gaussian Prediction

A two-layer pointwise MLP $h : \mathbb{R}^{132} \to \mathbb{R}^{38} \to \mathbb{R}^{38}$ (6.5k parameters) maps $\boldsymbol{f}'_a$ to raw parameters $(\bar\alpha_a, \bar{\boldsymbol{o}}_a, \bar{\boldsymbol{s}}_a, \bar{\boldsymbol{q}}_a, \bar{\boldsymbol{c}}_a) \in \mathbb{R}^{1+3+3+4+27}$, decoded as

$$\begin{aligned} \boldsymbol{\mu}_a &= \boldsymbol{x}_a + 3v\,\big(\sigma(\bar{\boldsymbol{o}}_a) - \tfrac{1}{2}\big), & \alpha_a &= \sigma(\bar\alpha_a), \\ \boldsymbol{s}_a &= \mathrm{clip}\,\big(\mathrm{softplus}(\bar{\boldsymbol{s}}_a - 4),\, s_{\min},\, s_{\max}\big), & \boldsymbol{\Sigma}_a &= \boldsymbol{R}(\bar{\boldsymbol{q}}_a/\|\bar{\boldsymbol{q}}_a\|)\,\mathrm{diag}(\boldsymbol{s}_a)^2\,\boldsymbol{R}(\bar{\boldsymbol{q}}_a/\|\bar{\boldsymbol{q}}_a\|)^\top, \end{aligned} \tag{5}$$

where σ is the sigmoid, $\boldsymbol{R}(\cdot)$ the rotation matrix of a unit quaternion, and the offset is confined to ± 1.5 fine voxels around the anchor. The colour $\boldsymbol{c}_a$ is a degree-2 spherical-harmonics vector whose DC term is set from the mean RGB of the anchor's pixels; h predicts the higher orders. A pointwise head suffices because 99.6% of anchors have no occupied 3^3 neighbour (Section 2.2), so a sparse convolution would reduce to its centre tap at almost every anchor (Remark 1); it also keeps the pipeline free of sparse-tensor operations.

2.5 Training Objective

Following DepthSplat and VolSplat (Xu et al., 2025; Wang et al., 2026a), the network is trained end to end from RGB supervision alone. Writing $\hat{\mathbf{I}}^{(m)}_{\mathcal{G}}$ for the m-th of M target views rendered from a Gaussian set $\mathcal{G}$ with a differentiable rasterizer (Kerbl et al., 2023),

$$\mathcal{L} = \sum_{m=1}^{M} \Big[\ell\big(\hat{\mathbf{I}}^{(m)}_{\mathcal{G}(\boldsymbol{D})}, \mathbf{I}^{(m)}\big) + \beta\,\ell\big(\hat{\mathbf{I}}^{(m)}_{\mathcal{G}(\tilde{\boldsymbol{D}})}, \mathbf{I}^{(m)}\big)\Big], \qquad \ell(\hat{\mathbf{I}}, \mathbf{I}) = \|\hat{\mathbf{I}} - \mathbf{I}\|_2^2 + \lambda\,\mathrm{LPIPS}(\hat{\mathbf{I}}, \mathbf{I}), \tag{6}$$

with $\lambda = 0.05$ and $\beta = 0.9$. $\mathcal{G}(\boldsymbol{D})$ is the Gaussian set built from the final depth and $\mathcal{G}(\tilde{\boldsymbol{D}})$ the one built from the coarse depth of Section 2.1 through the same anchors, 3D stage, and head (shared weights); no depth ground truth is used. All models train for 150k steps with AdamW under VolSplat's training protocol (Appendices B.1 and B.2). Nothing in Sections 2.2 to 2.4 has a data-dependent shape, so at inference the entire encoder runs as one fp16 CUDA graph (Section 3.3).

3 Experiments

3.1 Experimental Setup

Datasets and protocol. We train on RealEstate10K (Zhou et al., 2018) (256×256) and evaluate on VolSplat's official indices ($\sim$700 test scenes, 8 target views each) at 6, 12, and 24 context views; zero-shot generalization uses ACID (Liu et al., 2021) (173 scenes), and indoor transfer follows the FreeSplat protocol (Wang et al., 2024b) on ScanNet (Dai et al., 2017), fine-tuning every RealEstate10K checkpoint for 150k steps and evaluating on VolSplat's official 72-sequence index. All our models and our retrained VolSplat follow VolSplat's official training protocol (Section 2.5 and Appendix B.2; $4\times$ NVIDIA RTX PRO 6000).

Table 1: **RealEstate10K (Zhou et al., 2018), 6/12/24 views.** pixelSplat–GGN are quoted from VolSplat's retrainings (Wang et al., 2026a); our retraining uses VolSplat's official protocol (Section 3.1). ‡Pose-free, released checkpoints with ground-truth cameras (different training corpora and resolutions, no retraining). OOM: cannot infer on 96 GB.

Method	6 views			12 views			24 views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
pixelSplat (Charatan et al., 2024)	28.95	0.900	0.163	OOM	OOM	OOM	OOM	OOM	OOM
MVSplat (Chen et al., 2024)	29.13	0.924	0.091	26.97	0.912	0.101	26.23	0.903	0.108
TranSplat (Zhang et al., 2025)	29.62	0.928	0.084	28.00	0.920	0.089	26.65	0.884	0.115
DepthSplat (Xu et al., 2025)	30.52	0.931	0.079	28.54	0.919	0.088	26.26	0.880	0.115
GGN (Zhang et al., 2024)	26.68	0.879	0.133	25.83	0.870	0.142	23.86	0.840	0.171
AnySplat (Jiang et al., 2025)‡	25.37	0.811	0.140	25.55	0.815	0.136	27.23	0.882	0.107
WorldMirror (Liu et al., 2026)‡	29.20	0.914	0.106	29.56	0.921	0.101	29.37	0.920	0.103
VolSplat (reported)	31.30	0.941	0.075	29.40	0.928	0.085	27.21	0.896	0.111
VolSplat (retrained)	30.93	0.938	0.078	30.86	0.939	0.077	27.24	0.881	0.127
LoCoSplat (ours)	**31.51**	**0.942**	**0.074**	**31.47**	**0.942**	**0.073**	**30.57**	**0.934**	**0.082**

Table 2: **Six-view reconstruction efficiency** (RealEstate10K; same GPU and timing protocol for all methods, Appendix A). Baselines run in plain fp32 as released; LoCoSplat on its deployed path (fp16, CUDA graph) and in plain fp32. Methods as in Table 1. Training: 4 GPUs, batch 1/GPU, wall-clock extrapolated from the measured step rate to 150k steps. ‡Pose-free, released defaults, not retrainable here.

Method	PSNR↑	Params train. / total	Inference (ms)	Recon. FPS	Peak mem. (MB)	Train (it/s)	Train wall-clock
pixelSplat	28.95	119.0 / 119.0 M	238.5	4.2	31,869	1.05	∼39.6 h
MVSplat	29.13	12.0 / 12.0 M	93.8	10.7	3,528	2.92	∼14.3 h
DepthSplat	30.52	120.7 / 120.7 M	81.2	12.3	7,204	2.44	∼17.1 h
GGN	26.68	12.1 / 12.1 M	177.0	5.6	3,489	2.25	∼18.5 h
AnySplat‡	25.37	886.4 / 1,190.7 M	85.5	11.7	3,774	—	—
WorldMirror‡	29.20	1,263.3 / 1,263.3 M	90.8	11.0	5,673	—	—
VolSplat (retrained), fp32	30.93	51.4 / 138.0 M	137.0	7.3	7,271	1.18	∼35.4 h
LoCoSplat, fp32 eager	**31.51**	12.5 / 99.1 M	56.9	17.6	3,873	**3.17**	**∼13.2 h**
LoCoSplat, deployed	**31.51**	12.5 / 99.1 M	**33.0**	**30.3**	**1,081**	**3.17**	**∼13.2 h**

Baselines and metrics. We compare against pixel-aligned (pixelSplat, MVSplat, TranSplat, DepthSplat), post-processed (GGN), and pose-free (AnySplat, WorldMirror; released checkpoints) methods, FreeSplat on ScanNet, and VolSplat as published and retrained, the anchor for all paired analyses (Appendix B.2). We report PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018); all methods are timed with their released code on the same GPU under the same protocol (Appendix A).

3.2 Main Results on RealEstate10K

Table 1 reports quality across view counts. LoCoSplat has the best PSNR, SSIM, and LPIPS at every view count: $+0.58$ PSNR over our retrained VolSplat at 6 views and $+0.61$ at 12 (paired, 68% of scenes). Our retrained VolSplat is the stronger 12-view baseline (30.86 vs. the published 29.40) and anchors every paired comparison (Appendix B.2). At 24 views every RealEstate10K-trained method degrades as the context densifies, VolSplat most; LoCoSplat does not share the drop: 30.57 PSNR, $+3.3$ paired over VolSplat (98% of scenes) and $+1.2$ over WorldMirror, which is pretrained on a large multi-dataset corpus and essentially flat in the view count; Section 3.5 traces the difference to one hyper-parameter of the depth branch. Qualitatively, pixel-aligned methods blur and duplicate thin structures at 6 views (Fig. 3); at 24 views VolSplat streaks and the pose-free baselines blur branches and text (Fig. 8; 12 views in Fig. 7).

3.3 Real-Time Reconstruction

Table 2 summarizes efficiency at 6 views. The 3D stage has no data-dependent shape (a sort-based voxel group-by padded to a static $V{\cdot}H{\cdot}W$ row count, fixed-table gathers and scatters, pointwise g and h), so the whole encoder is captured as one fp16 CUDA graph (Appendix B.1). LoCoSplat reconstructs $4.2\times$ faster than VolSplat on this path and $2.4\times$ faster at equal fp32 precision (56.9 vs.

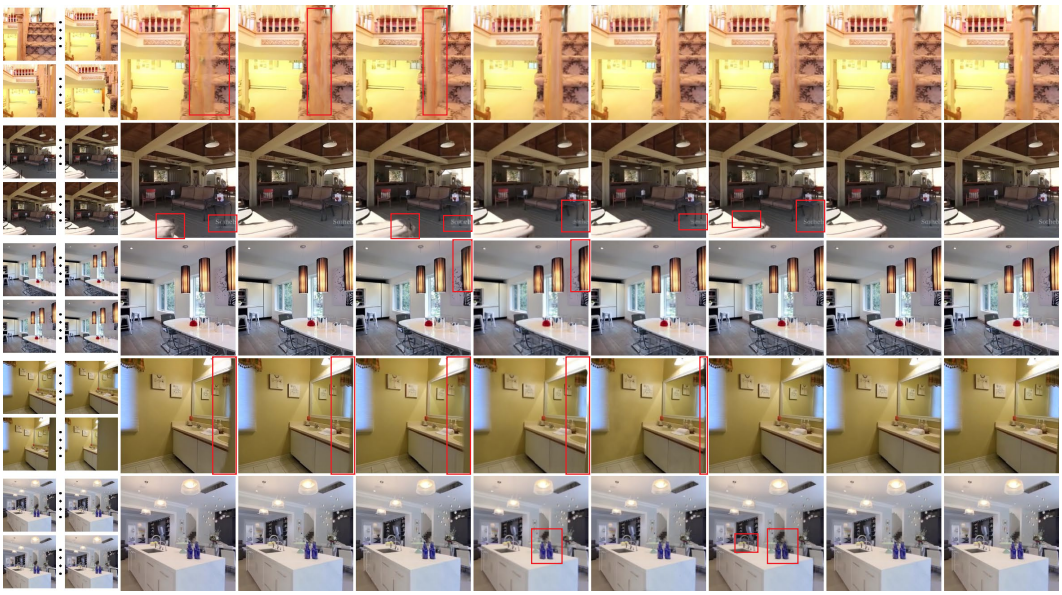

Figure 3: **RealEstate10K (Zhou et al., 2018), 6 input views.** LoCoSplat ("Ours") preserves fine detail and thin structures with less ghosting; red boxes highlight local differences. Checkpoints follow Table 1 (VolSplat: our retraining; AnySplat and WorldMirror: released checkpoints with ground-truth cameras).

137.0 ms), trains 2.7× faster (13.2 vs. 35.4 h; 27 vs. 159 h at 24 views), and uses 85% less peak inference memory with 76% fewer trainable parameters; VolSplat gains only 1.4% from the same optimizations, as its sparse U-Net resists mixed precision, static shapes, and graph capture. Most of the speedup is architectural: removing the U-Net takes the fp32 eager encoder from 137.0 to 94.4 ms *while improving quality* (Tables 8 and 9). Against the released baselines, LoCoSplat is the fastest at every view count, at the lowest memory and the highest PSNR (2.5× faster than DepthSplat, the next fastest; 2.9 vs. 27.3 GB for VolSplat at 24 views; Tables 6 and 7); MVSplat and GGN match its trainable-parameter count but run 2.8–5.4× slower and lose 2.4 and 4.8 PSNR.

3.4 Generalization: Zero-Shot ACID and ScanNet Fine-Tuning

Table 3 evaluates RealEstate10K-trained models on ACID without fine-tuning. The margin over VolSplat does not shrink out of domain: it grows to +0.87 PSNR paired (77% of scenes) against +0.58 in domain, and LoCoSplat also exceeds the PSNR and LPIPS VolSplat reports for its own checkpoint. In Fig. 4, VolSplat's refinement smears thin structures such as a radio tower and the pixel-aligned baselines ghost or dissolve them, which LoCoSplat preserves. Table 4 fine-tunes LoCoSplat and VolSplat from their RealEstate10K checkpoints under VolSplat's ScanNet fine-tuning protocol; the ordering carries over with a wider margin: LoCoSplat gains +0.92 PSNR over the fine-tuned VolSplat (on all 8 scenes) and lies above every published entry (Fig. 5, Appendix E.3).

3.5 Ablations and Analysis

Statistical protocol. All ablations retrain under VolSplat's exact 150k-step training protocol with identical losses and seeds, changing one component at a time. Gaps between our own models are per-scene *paired* differences; a gap within the replicate noise of repeated training runs ($\leq$0.03 PSNR, Appendix B.2) is reported as a tie. Full ablation tables are in Appendix C.3.

Ablation models. The coarse-source ablations run on the *two-grid model*: VolSplat's pipeline with the two grids and fusion MLP of Section 2.3 as its 3D stage (trilinear splat and query, 0.16M-parameter g), keeping its 164-d features, sparse-convolutional head, and D=128 (Table 11). The level-count controls (Table 12) run on the final LoCoSplat configuration.



Figure 4: **Zero-shot ACID (Liu et al., 2021), 6 input views.** Models trained on RealEstate10K, checkpoints as in Table 3; LoCoSplat preserves thin structures and road markings with less blur and ghosting (red boxes).

Table 3: **Zero-shot RE10K→ACID (Liu et al., 2021), 6 views.** MVSplat, DepthSplat, and GGN: VolSplat's 6-view releases; only VolSplat (reported) is quoted, other rows are measured under its official protocol.

Method	PSNR↑	SSIM↑	LPIPS↓
MVSplat	30.88	0.911	0.105
DepthSplat	31.93	0.917	0.097
GGN	27.45	0.828	0.188
VolSplat (reported)	32.65	**0.932**	0.092
VolSplat (retrained)	32.29	0.924	0.097
LoCoSplat	**33.16**	0.931	**0.091**

Table 4: **ScanNet (Dai et al., 2017), 6 input views** (72 sequences; a *fine-tuning* experiment). Baselines and VolSplat (reported) quoted from VolSplat (Wang et al., 2026a); the last two rows use its 150k-step fine-tuning protocol; paired analysis in Section 3.4.

Method	PSNR↑	SSIM↑	LPIPS↓
FreeSplat	27.45	0.829	0.222
FreeSplat++	27.45	0.829	0.223
AnySplat	19.45	0.626	0.344
WorldMirror	25.83	0.819	0.136
VolSplat (reported)	28.41	0.906	0.127
VolSplat (retrained)	28.24	0.905	0.122
LoCoSplat	**29.16**	**0.911**	**0.115**

Removing the U-Net helps under a fixed training protocol. Without the U-Net, fixed grids fused by g *improve* PSNR by $+0.71$ with eight levels (31.64 vs. 30.93, 6 views) and $+0.70$ with the two levels of LoCoSplat, with 33% fewer trainable parameters and $1.67\times$ higher training throughput (Table 9). The gap holds at every matched checkpoint and widens with training: the U-Net's test PSNR plateaus after $\sim$80k steps while the grid model keeps improving. The 3D stage still needs only minimal context: dropping the coarse level costs LoCoSplat 0.13 PSNR (Table 12), four times the replicate noise. At 12 views the full pipeline gains $+0.61$ over our retrained VolSplat (paired, 68% of scenes; Table 10).

The coarse context needs neither learning nor appearance. In place of the learned coarse level (Table 11), (i) a *fixed, untrained* low-pass of the fine field (box pooling or Gaussian smoothing) ties it on all three metrics, and (ii) a detached 10-channel geometric descrip-

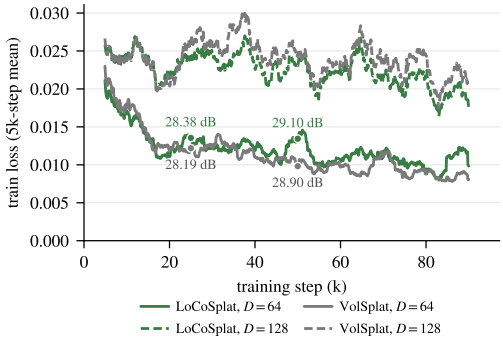

Figure 6: **24-view training loss** (5k-step mean). Both D=128 models plateau at twice the loss; the original VolSplat with D=64 follows LoCoSplat. Dots: PSNR on the 24-view index at 25k and 50k steps. Curves cover the first 90k steps of the 150k-step schedule.

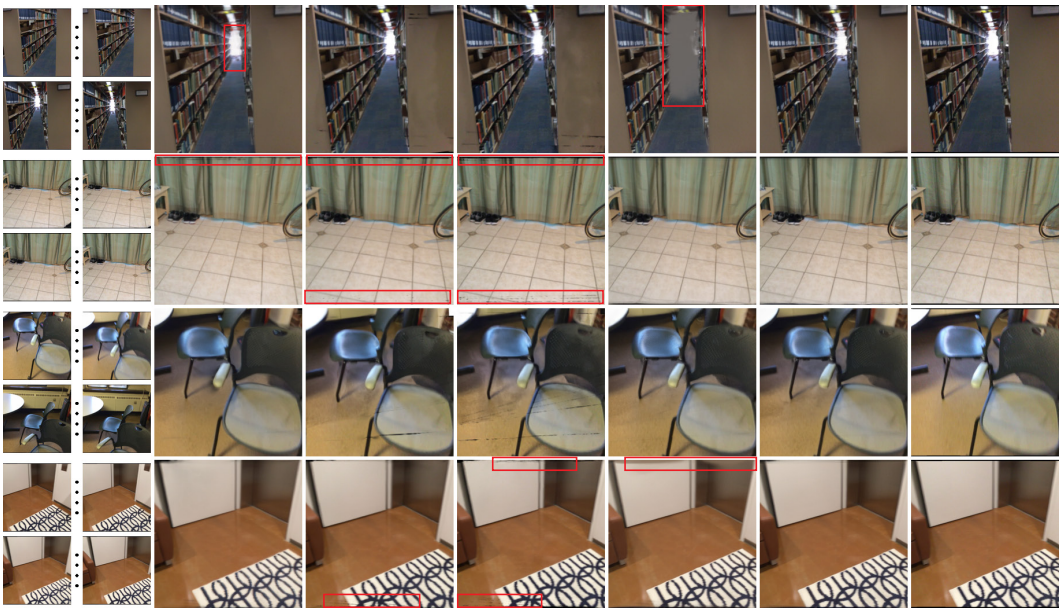

Figure 5: **ScanNet (Dai et al., 2017), 6 input views.** LoCoSplat and VolSplat are fine-tuned under VolSplat's protocol, the other methods use released checkpoints (Table 4); LoCoSplat preserves shelves and rug patterns with fewer artifacts.

Table 5: **Plane-sweep candidate count and the dense-view collapse.** Full-budget retrainings of LoCoSplat with one component changed. Δu: adjacent-candidate disparity relative to 6 views, D=128 (Eq. (7)); paired ΔPSNR vs. LoCoSplat at the same view count. D=256 and D=32 were pre-registered predictions; the 12-view reference is re-evaluated for this table (31.49 vs. 31.47 in Table 1).

Model	D	Δu (rel.)	PSNR↑	SSIM↑	LPIPS↓	Paired ΔPSNR vs. LoCoSplat	LoCoSplat better
12 views							
LoCoSplat, D=256 (pre-reg.)	256	0.23	28.39	0.895	0.109	−3.10	99%
LoCoSplat	64	0.92	**31.49**	**0.943**	**0.073**	—	—
24 views							
VolSplat (retrained)	128	0.22	27.24	0.881	0.127	−3.3	98%
LoCoSplat, D=128	128	0.22	28.08	0.888	0.114	−2.49	98%
LoCoSplat, trilinear fine level	64	0.45	30.45	0.932	0.084	−0.12	65%
LoCoSplat, D=32 (pre-reg.)	32	0.91	30.39	0.932	0.083	−0.18	55%
LoCoSplat	64	0.45	**30.56**	**0.934**	**0.082**	—	—

tor with no appearance content (density, position offset, per-axis spread, log-depth moments; Appendix B.1) matches it (+0.03, a tie). On LoCoSplat, two levels tie four levels within $[0.001, 0.008]$ and eight reaching 0.128; only a wider window adds (+0.12 at 2.048, Table 12).

The plane-sweep candidate count, not the 3D stage, decides the 24-view drop. Retraining LoCoSplat at 24 views with one change reverted at a time (Table 5), trilinear fine splatting costs 0.1 PSNR, whereas VolSplat's D=128 plane-sweep candidates (instead of 64) reproduce its failure (28.08 PSNR, worse on 98% of scenes) with every 3D-stage component in place; every degrading 24-view model runs 128 candidates and every healthy one 64. Denser official indices shrink the matching baseline B, and the per-candidate disparity $\Delta u \propto B/(D-1)$ of the failing configuration is half that of the healthy ones (Appendix D). Collapsed models have a switched-off multi-view depth head (the same one-hot candidate at every pixel, depth carried by the monocular path), whereas healthy ones are peaked and input-dependent (Table 13). The account passed both pre-registered tests: D=256 at 12 views ($\Delta u = 0.23$) collapses (−3.10 paired, 99% of scenes) and D=32 at 24 views ($\Delta u = 0.91$) stays healthy (−0.18). The converse holds on the baseline itself: retraining the original VolSplat at 24 views with D=64 and nothing else changed puts it on the healthy loss

trajectory of LoCoSplat (Fig. 6), so the drop is a property of the depth branch's candidate spacing, not of the 3D stage.

4 Related Work

Per-scene novel view synthesis. NeRF (Mildenhall et al., 2020) and its successors (Barron et al., 2021; Fridovich-Keil et al., 2022; Müller et al., 2022; Barron et al., 2023) and 3D Gaussian Splatting (Kerbl et al., 2023; Lu et al., 2024; Fan et al., 2024; Ren et al., 2025) optimize each scene; we target the feed-forward regime, where reconstruction itself must be fast.

Feed-forward 3DGS. Generalizable NVS networks (Yu et al., 2021; Wang et al., 2021; Chen et al., 2021) amortize reconstruction across scenes; pixelSplat (Charatan et al., 2024) and MVSplat (Chen et al., 2024) predict pixel-aligned Gaussians from epipolar or cost-volume features; TranSplat (Zhang et al., 2025), DepthSplat (Xu et al., 2025), and FreeSplat (Wang et al., 2024b; 2026b) improve depth estimation and cross-view fusion; GGN (Zhang et al., 2024) and AnySplat (Jiang et al., 2025) post-process pixel-aligned Gaussians, and WorldMirror (Liu et al., 2026) predicts them jointly with geometry. A pose-free line (Ye et al., 2025; Huang & Mikolajczyk, 2025), to which AnySplat and WorldMirror also belong, builds on DUSt3R- and VGGT-style geometry transformers (Wang et al., 2024a; 2025) pretrained on large multi-dataset corpora, while SelfSplat (Kang et al., 2025) does so without 3D priors; these models change the inputs and the training data rather than the 3D stage, so we report them as released, for reference. VolSplat (Wang et al., 2026a) unprojects per-pixel features along predicted depth, voxelizes them, refines the sparse volume with a 3D U-Net (Çiçek et al., 2016; Choy et al., 2019), and decodes one Gaussian per occupied voxel; related systems likewise lean on sparse 3D CNNs (Zhang et al., 2022) or volumetric fusion (Miao et al., 2025), while Long-LRM (Ziwen et al., 2025) instead scales a long-sequence Mamba/transformer model over image tokens.

Multi-resolution grids and efficient inference. Instant-NGP (Müller et al., 2022) popularized multi-resolution feature grids with tiny MLPs for per-scene optimization, on a long line of grid and octree structures (Meagher, 1982; Riegler et al., 2017; Koneputugodage et al., 2023; Fridovich-Keil et al., 2023). Our grids store no learned scene content and are rebuilt from the input images at every forward pass: they are a splat-and-query aggregation structure that computes binned local means of point features, in the sense of binned kernel estimators (Hall & Wand, 1996), not a learned representation.

5 Conclusion

We have presented LoCoSplat, a real-time feed-forward 3DGS method whose 3D stage has no learned 3D network, only a fixed average that a grid computes for free. Its premise is that a Gaussian is local: once depth is known, the 3D stage has only to supply the surface layout around each anchor. Controlled ablations show that this 3D stage needs very little: a fixed low-pass context read by a tiny pointwise map beats the sparse 3D U-Net under an identical training protocol and trains faster, and the minimal coarse context that remains useful needs neither learning nor appearance content. LoCoSplat outperforms VolSplat on every metric at 6, 12, and 24 views, in and out of domain, and runs as a single fp16 CUDA graph at 30 FPS, $4.2\times$ faster than VolSplat and the fastest of seven feed-forward methods. A second finding lies outside the 3D stage: for this pipeline the 24-view collapse is decided by the plane-sweep candidate count, and a disparity-resolution account of it predicted, before training, both a new collapse and a healthy dense-view model.

Limitations. The 3D-stage ablations run only at 6 views and only against VolSplat's U-Net, not smaller 3D CNNs or local attention, and we have not evaluated beyond the posed 256×256 regime: at 512^2, where $H = 0.128$ may not transfer, on wider-baseline data such as DL3DV, or under noisy depth or pose error. In principle the geometry is depth-dominated: the head moves each centre at most $1.5v$ from its anchor (Eq. (5)), so the 3D layout inherits the quality of the depth prediction, and the 24-view collapse indeed occurs in the depth branch. The plain mean is optimal only for a smooth field with homoscedastic noise (Appendix C.2) and blurs across surfaces at occlusion boundaries, and the pointwise head relies on anchor isolation (Remark 1), which denser inputs may erode.

AI USE STATEMENT

We used LLM-based assistants in three ways. First, for writing: to draft sections of the paper from the authors' notes, results, and outlines, to polish the text, and to check compliance with the conference template. Second, for retrieval and discovery: to help find related work. Third, as coding assistants: part of the implementation, evaluation, and plotting code was generated by LLM-based tools under the authors' supervision. They were not used to generate research ideas, hypotheses, theory, proofs, or methods, and every claim and number in the paper originates from the authors' own work. All drafted or edited text and all generated code were reviewed by the authors, who take full responsibility for the content of this paper.

ETHICS STATEMENT

We use only public research datasets (RealEstate10K, ACID, ScanNet) under their licenses; no new data were collected and no human subjects were involved. Our contribution lowers the cost of existing feed-forward reconstruction rather than enabling new capabilities, and carries the same dual-use considerations as prior work in the area. We are not aware of any conflicts of interest.

REPRODUCIBILITY STATEMENT

All models, including our retrained VolSplat baseline, follow VolSplat's public training protocol and official evaluation indices (Section 3.1). Architecture and optimizer details are in Appendix B.1, the timing protocol and hardware in Appendix A, the paired-comparison protocol in Section 3.5 and Appendix B.2, and the selection procedure for qualitative figures in Appendices E.1 and E.2. Code, configurations, evaluation indices, and checkpoints will be released upon publication.

REFERENCES

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Int. Conf. Comput. Vis.*, pp. 5855–5864, 2021.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *Int. Conf. Comput. Vis.*, pp. 19697–19705, 2023.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 19457–19467, 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 *Int. Conf. Comput. Vis.*, pp. 14124–14133, 2021.

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 *Eur. Conf. Comput. Vis.*, pp. 370–386. Springer, 2024.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 3075–3084, 2019.

Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 424–432. Springer, 2016.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 5828–5839, 2017.

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 *Adv. Neural Inform. Process. Syst.*, volume 37, pp. 140138–140158, 2024.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 5501–5510, 2022.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit radiance fields in space, time, and appearance. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 12479–12488, 2023.

Peter Hall and Matthew P. Wand. On the accuracy of binned kernel density estimators. *Journal of Multivariate Analysis*, 56(2):165–184, 1996.

Ranran Huang and Krystian Mikolajczyk. No pose at all: Self-supervised pose-free 3D Gaussian splatting from sparse views. In *Int. Conf. Comput. Vis.*, pp. 27947–27957, 2025.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. AnySplat: Feed-forward 3D Gaussian splatting from unconstrained views. *ACM Trans. Graph.*, 44(6):257:1–257:16, 2025.

Gyeongjin Kang, Jisang Yoo, Jihyeon Park, Seungtae Nam, Hyeonsoo Im, Sangheon Shin, Sangpil Kim, and Eunbyung Park. SelfSplat: Pose-free and 3D prior-free generalizable 3D Gaussian splatting. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 22012–22022, 2025.

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

Chamin Hewa Koneputugodage, Yizhak Ben-Shabat, and Stephen Gould. Octree guided unoriented surface reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 16717–16726, 2023.

Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Int. Conf. Comput. Vis.*, pp. 14458–14467, 2021.

Yifan Liu, Zhiyuan Min, Zhenwei Wang, Junta Wu, Tengfei Wang, Yixuan Yuan, Yawei Luo, and Chunchao Guo. WorldMirror: Universal 3D world reconstruction with any-prior prompting. In *Int. Conf. Mach. Learn.*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Int. Conf. Learn. Represent.*, 2019.

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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 20654–20664, 2024.

Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

Sheng Miao, Jiaxin Huang, Dongfeng Bai, Xu Yan, Hongyu Zhou, Yue Wang, Bingbing Liu, Andreas Geiger, and Yiyi Liao. EVolSplat: Efficient volume-based Gaussian splatting for urban view synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 11286–11296, 2025.

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 *Eur. Conf. Comput. Vis.*, pp. 405–421. Springer, 2020.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022.

Elizbar A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.

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

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 3577–3586, 2017.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 5294–5306, 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 4690–4699, 2021.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 20697–20709, 2024a.

Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Feng Chen, Jia-Wang Bian, Zheng Zhu, Donny Y. Chen, and Bohan Zhuang. VolSplat: Rethinking feed-forward 3D Gaussian splatting with voxel-aligned prediction. In *Eur. Conf. Comput. Vis.*, 2026a.

Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. FreeSplat: Generalizable 3D Gaussian splatting towards free view synthesis of indoor scenes. In *Adv. Neural Inform. Process. Syst.*, volume 37, pp. 107326–107349, 2024b.

Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. FreeSplat++: Generalizable 3D Gaussian splatting for efficient indoor scene reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 48(7):7749–7765, 2026b.

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

Geoffrey S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, 26(4):359–372, 1964.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45 (11):13941–13958, 2023.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian splatting and depth. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 16453–16463, 2025.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. In *Adv. Neural Inform. Process. Syst.*, volume 37, pp. 21875–21911, 2024.

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 *Int. Conf. Learn. Represent.*, 2025.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 4578–4587, 2021.

Chuanrui Zhang, Yingshuang Zou, Zhuoling Li, Minmin Yi, and Haoqian Wang. TranSplat: Generalizable 3D Gaussian splatting from sparse multi-view images with transformers. In *AAAI Conf. Artif. Intell.*, volume 39, pp. 9869–9877, 2025.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 586–595, 2018.

Shengjun Zhang, Xin Fei, Fangfu Liu, Haixu Song, and Yueqi Duan. Gaussian graph network: Learning efficient and generalizable Gaussian representations from multi-view images. In *Adv. Neural Inform. Process. Syst.*, volume 37, pp. 50361–50380, 2024.

Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. NeRFusion: Fusing radiance fields for large-scale scene reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 5449–5458, 2022.

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

Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-LRM: Long-sequence large reconstruction model for wide-coverage Gaussian splats. In *Int. Conf. Comput. Vis.*, pp. 4349–4359, 2025.

APPENDIX

A Efficiency Measurements **14**

A.1 Benchmark Protocol 14

A.2 Training Throughput 14

A.3 Scaling and Speedup Decomposition 15

A.4 Training at High View Counts 15

B Implementation and Protocol Details **15**

B.1 Implementation Details 15

B.2 Experimental Protocol 17

C Analysis of the 3D Stage **18**

C.1 The Path to the Two-Grid Model 18

C.2 Design Notes 18

C.3 Ablation Tables 19

D The 24-View Drop in Detail **21**

D.1 The Drop Lies in the Depth Branch, Not the 3D Stage 21

D.2 A Disparity-Resolution Account 21

D.3 The Collapsed Depth Head 22

E Additional Qualitative Results **22**

E.1 Qualitative Results at 12 Input Views 22

E.2 24-View Evaluation and Example Selection 22

E.3 Qualitative Results on ScanNet 22

E.4 More Qualitative Comparisons on RealEstate10K 23

A Efficiency Measurements

A.1 Benchmark Protocol

All timings (Tables 2, 6 and 7) are measured on a single idle RTX PRO 6000 (96 GB; PyTorch 2.8, CUDA 12.8) with one common protocol: each method alone on the GPU, the same 100 RealEstate10K test scenes, 10 warm-up scenes, batch size 1, 256^2 inputs, 8 target views, per-scene medians of synchronized wall-clock time. "Encoder" maps context images (plus cameras) to Gaussians; "render" rasterizes all 8 targets with the method's own renderer. FPS is $1000/t_{\text{enc}}$ (or $1000/(t_{\text{enc}} + t_{\text{render}})$); memory is peak allocated. Baselines run as released (fp32; AnySplat/WorldMirror with a bf16 ViT at 252^2, WorldMirror with ground-truth camera priors); LoCoSplat runs its deployed path (Appendix B.1), and its fp32 eager path gives the same ordering. Checkpoints: the 6-view releases of the VolSplat authors for pixelSplat/MVSplat/DepthSplat/GGN (their cost is shape-only, so reuse at 12/24 views leaves timing unchanged; pixelSplat's is 6-view-specific), authors' weights for AnySplat/WorldMirror, and our retrained VolSplat and LoCoSplat. Code changes were limited to dependency fallbacks and two one-line fixes. TranSplat is not timed (its code hard-codes 2–4 views). "Trainable" parameters are those each method's training protocol updates; "total" adds frozen backbones.

A.2 Training Throughput

All six retrainable methods run their own code under our training protocol (4 GPUs, DDP, batch 1/GPU, 6 context + 8 target views, MSE + LPIPS), timed over steps 50–350 and extrapolated to 150k steps (MVSplat/GGN received pixelSplat's multi-view sampler). LoCoSplat trains at 3.17 steps/s vs. 1.18 for VolSplat, with 20 GB peak memory per GPU vs. 37 (VolSplat), 66 (pixelSplat), and 23–28 GB (others); our full 150k run including validation took 17.6 h. AnySplat and WorldMirror fine-tune billion-parameter backbones on dataset mixtures and cannot be retrained under this protocol.

Table 6: **Reconstruction speed, 6/12/24 views** (RealEstate10K, 100-scene medians; protocol above). FPS counts scenes; "+ render" includes 8 targets. Checkpoints: MVSplat, DepthSplat, and GGN are timed at 12 and 24 views with their released 6-view checkpoints (their cost is shape-only, so timing is unaffected); VolSplat at 24 views with our 12-view checkpoint (§); LoCoSplat with its own 6-, 12-, and 24-view checkpoints. Baselines run in plain fp32 as released, LoCoSplat on its deployed fp16 CUDA-graph path (fp32 eager: 56.9 ms at 6 views, the same ordering). ‡Pose-free, released defaults: 252^2, bf16 backbone, multi-dataset pretraining; WorldMirror uses ground-truth camera priors.

Method	Encoder (ms) ↓			Encoder-only FPS ↑			Encoder + render, FPS ↑		
	6v	12v	24v	6v	12v	24v	6v	12v	24v
pixelSplat	238.5	—	—	4.2	—	—	4.0	—	—
MVSplat	93.8	303.9	1111.7	10.7	3.3	0.9	9.6	3.1	0.9
DepthSplat	81.2	149.8	306.4	12.3	6.7	3.3	11.1	6.1	3.0
GGN	177.0	483.8	1451.7	5.6	2.1	0.7	5.4	2.0	0.7
AnySplat‡	85.5	116.6	209.0	11.7	8.6	4.8	10.5	7.6	4.3
WorldMirror‡	90.8	142.0	259.0	11.0	7.0	3.9	10.7	6.8	3.7
VolSplat (retrained)	137.0	263.8	548.5§	7.3	3.8	1.8§	6.9	3.6	1.7§
LoCoSplat (deployed)	**33.0**	**62.9**	**137.4**	**30.3**	**15.9**	**7.3**	**23.7**	**12.9**	**6.1**

Table 7: **Rendering, memory, and Gaussian counts.** Protocol and ‡ as in Table 6.

Method	Render 8 views (ms) ↓			Peak memory (MB) ↓			#Gaussians (mean)		
	6v	12v	24v	6v	12v	24v	6v	12v	24v
pixelSplat	10.4	—	—	31,869	—	—	393,216	—	—
MVSplat	9.9	17.2	31.7	3,528	7,277	24,341	393,216	786,432	1,572,864
DepthSplat	8.4	13.4	23.8	7,204	13,912	27,330	393,216	786,432	1,572,864
GGN	9.2	9.3	9.8	3,489	7,158	24,010	176,837	238,445	319,885
AnySplat‡	10.4	14.9	23.8	3,774	4,199	5,106	343,347	640,670	1,145,219
WorldMirror‡	2.9	5.0	8.7	5,673	6,497	8,152	326,098	577,272	972,238
VolSplat (retrained)	8.6	14.6	25.5	7,271	13,964	27,349	393,175	786,288	1,572,369
LoCoSplat	9.1	14.8	26.5	**1,081**	**1,695**	**2,925**	393,216	786,432	1,572,864

A.3 Scaling and Speedup Decomposition

Scaling. Deployed LoCoSplat is fastest at every view count, and the ordering follows how views are aggregated: pixelSplat's epipolar transformer is slowest already at 6 views; MVSplat and GGN attend across view pairs and scale near-quadratically ($11.9\times$ / $8.2\times$ from 6 to 24 views); DepthSplat and VolSplat scale linearly from a high constant; the pose-free models scale sub-linearly but start 2.6–2.7$\times$ slower. LoCoSplat scales linearly from the lowest constant (its $4.2\times$ from 6 to 24 views reflects the static stage padded to $V{\cdot}256^2$ rows) and is the only encoder under 150 ms at 24 views, with 1.1/1.7/2.9 GB peak memory vs. $\geq$24 GB for every pixel-aligned baseline at 24 views. VolSplat's time varies most across scenes (s.d. 20–40 ms), since its sparse U-Net cost follows voxel occupancy.

Speedup decomposition. With no dynamic sparse computation left, the whole encoder is one static-shape graph, so every systems optimization applies end-to-end. Of the $4.2\times$ speedup (Tables 8 and 9), the architecture accounts for most: dropping the U-Net and pruning to two levels (137.0→94.4 ms, *while improving quality*), the graph-native retrain (58.9 ms), and two backbone-side levers (56.9 ms in fp32, still $+0.58$ PSNR over VolSplat). fp16, type-check removal, whole-encoder CUDA graphs, and stream parallelism give the rest (33.0 ms; 42 ms including 8 rendered views). Checkpoint-compatible steps are metric-identical ($\leq 10^{-4}$ PSNR); the retrain steps cost $31.63 \rightarrow 31.51$ in total. The same optimizations speed VolSplat up by only 1.4%: its dominant sparse U-Net benefits from none of these optimizations.

A.4 Training at High View Counts

Without recomputation, 24-view VolSplat needs ∼126 GB of fp32 activations. With exact activation recomputation (deltas $\leq 6 \times 10^{-6}$) its 24-view run peaked at 53 GB and took 159 h, with per-GPU memory growing from 48 to 62 GB, versus 27 h at flat memory for LoCoSplat-24v.

Table 8: **Cumulative speedups.** Top: checkpoint-compatible optimizations; bottom: retrained models with deployed stacks.

Configuration	Encoder (ms)	PSNR
Two-grid model (Section 3.5), original code	94.4	31.63
+ fused ops (SDPA, packed voxel unique, fused gathers)	87.4	31.63
+ fp16 image-side autocast	83.8	31.63
+ runtime type-checking removed	74.4	31.63
+ CUDA-graphed 2D image side	67.4	31.63
graph-native retrain (pointwise head, fine-from-base, D=64)	52.0	31.60
+ stream parallelism (mono ‖ multi-view)	47.0	31.60
+ ViT-B backbone, ViT-S-width interface	37.7	31.58
+ DWConv half-width regressor (= LoCoSplat, full stack)	**33.0**	31.51

B Implementation and Protocol Details

B.1 Implementation Details

Grids. Both grids store the feature mean $\boldsymbol{u}_H$ (Eq. (2)) of a linear projection of the point features; no density, position, or covariance channel. For points $(\boldsymbol{x}_i, \boldsymbol{f}_i)$, $\boldsymbol{f}_i \in \mathbb{R}^{132}$ (164 for the ablation models of Section 3.5), and levels $\ell \in \{0, 1\}$ with $h_0 = 0.001$, $h_1 = 0.128$:

1. *Project*: $\boldsymbol{W}_\ell \in \mathbb{R}^{16\times 132}$ per level (4.3k parameters, the only learned part of the grids).
2. *Splat*: each point goes to its nearest node $c_\ell(\boldsymbol{x}_i) = \mathrm{rnd}(\boldsymbol{x}_i/h_\ell)$ (box kernel) and each occupied cell stores the mean of $\boldsymbol{W}_\ell \boldsymbol{f}_i$; the ablation models use weight-normalized trilinear splatting (tent kernel). Empty cells hold zero.
3. *Query*: anchor a reads $\boldsymbol{u}_\ell(c_\ell(\boldsymbol{x}_a))$ (ablation models: trilinear, renormalized over occupied corners). Since $h_0 = v$, the fine level equals the per-voxel mean $\boldsymbol{W}_0 \boldsymbol{f}_a$ exactly and is taken directly from the base voxelization.
4. *Fuse*: $\boldsymbol{f}'_a = \boldsymbol{f}_a + g([\boldsymbol{u}_0; \boldsymbol{u}_1; \boldsymbol{f}_a])$, with g an MLP $164 \rightarrow 256 \rightarrow 256 \rightarrow 132$ (0.14M).



Figure 7: **RealEstate10K (Zhou et al., 2018), 12 input views** (four shown). LoCoSplat ("Ours") preserves masonry, branches, and distant structures where baselines blur or introduce floaters. Red boxes highlight local differences. VolSplat is retrained at 12 views; pose-free baselines use released checkpoints with ground-truth cameras, as in Fig. 3.

Nearest-cell splat/query is quality-neutral against trilinear. The geometry-only coarse level of Table 11 (two-grid model only, not part of LoCoSplat) replaces steps 2–3 at the coarse level with a detached 10-channel descriptor, zero-padded to 16: offset of the cell's mean point from the anchor over H (3), per-axis s.d. of the positions in the cell over H (3), mean and s.d. of log depth (2), $\log(1 + n_H)/5$ with n_H the cell's point count (1), and query occupancy (1).

Head and parameters. The two-grid model keeps VolSplat's two 3^3 sparse convolutions ($164 \to 38 \to 38$, 0.21M); LoCoSplat uses a pointwise MLP ($132 \to 38 \to 38$, 6.5k). Trainable / total parameters (86.6M of them the frozen ViT-B): VolSplat 51.4 / 138.0M, two-grid model 34.5 / 121.1M, LoCoSplat 12.5 / 99.1M.

Deployed inference stack. An exact sort-based voxel group-by padded to a static $B{\cdot}V{\cdot}H{\cdot}W$ row count (dead rows get zero opacity and are moved out of view) removes all data-dependent shapes and host syncs, so the entire encoder is captured as one CUDA graph. fp16 autocast covers the image side, with depth values and pose/warp math pinned to fp32 and the 3D stage in fp32 (68.8 dB agreement with fp32 renders; benchmark metrics identical to four decimals). Further steps: SDPA with cached masks, fused cost-volume gathers, no runtime type-checking, and stream parallelism between the monocular and multi-view branches.

Backbone-side levers. A ViT-S-width interface to the retained ViT-B backbone (four trainable $768 \to 384$ projections) recovers $\sim$88% of the quality a full ViT-S swap loses, so the pretrained prior,

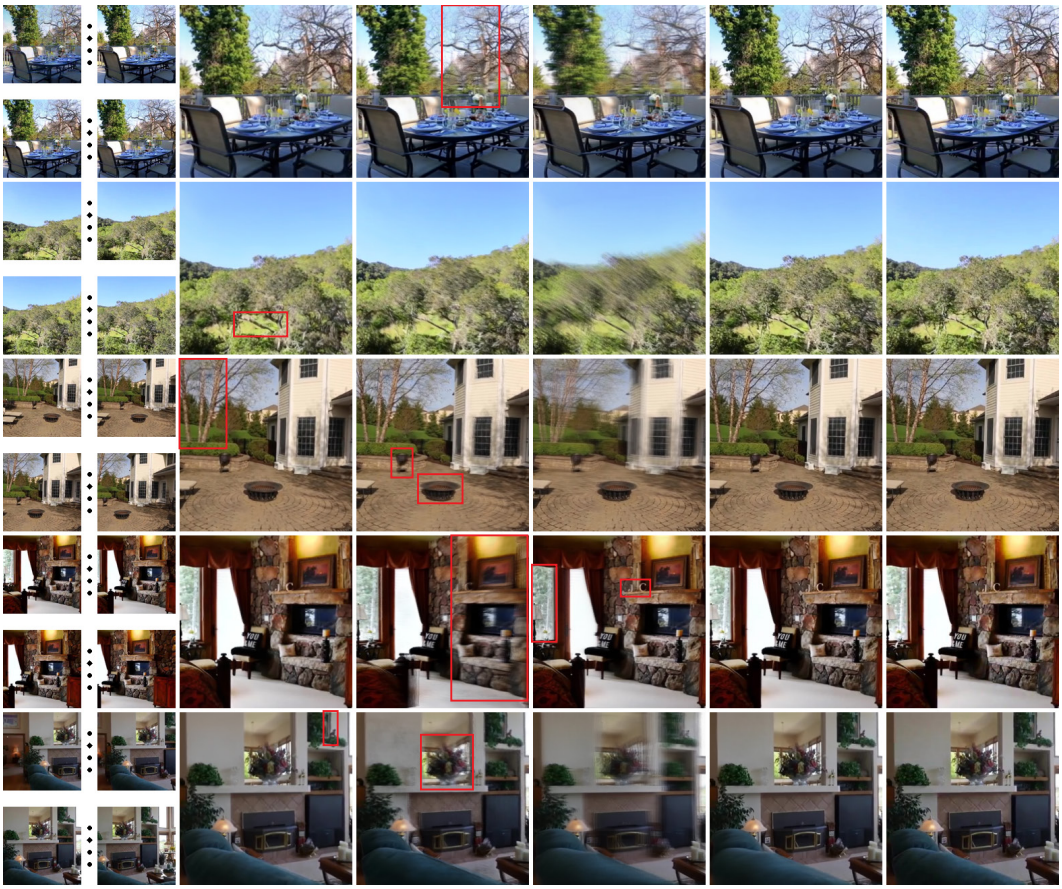

Figure 8: **RealEstate10K (Zhou et al., 2018), 24 input views** (four shown). In these selected examples, LoCoSplat ("Ours") preserves branches, text, and fine structure where VolSplat streaks and pose-free baselines blur. Red boxes highlight local differences. Checkpoints, evaluation paths, and example selection are detailed below.

not the width, is what matters. A depthwise-separable half-width cost-volume regressor is quality-neutral, whereas a plain-conv width cut loses 0.45 PSNR.

B.2 Experimental Protocol

Data and training. We train on RealEstate10K (Zhou et al., 2018) (256^2) and evaluate on VolSplat's official indices (∼700 scenes, 8 targets) at 6/12/24 views, zero-shot on ACID (Liu et al., 2021) (173 scenes), and on ScanNet (Dai et al., 2017) following FreeSplat (Wang et al., 2024b): each RealEstate10K checkpoint is fine-tuned for 150k steps with VolSplat's training protocol and evaluated on its 72-sequence index. All our models and our retrained VolSplat share this protocol: 150k steps, $4\times$ RTX PRO 6000, batch 1/GPU, AdamW (Loshchilov & Hutter, 2019) with a one-cycle schedule, lr 2×10^{-4} (2×10^{-6} for the backbone), MSE + LPIPS + intermediate depth supervision, identical seeds; 12/24-view runs use exact activation recomputation.

Baselines. Pixel-aligned methods (pixelSplat (Charatan et al., 2024), MVSplat (Chen et al., 2024), TranSplat (Zhang et al., 2025), DepthSplat (Xu et al., 2025)), GGN (Zhang et al., 2024), the pose-free AnySplat (Jiang et al., 2025) and WorldMirror (Liu et al., 2026), and FreeSplat (Wang et al., 2024b; 2026b) on ScanNet; numbers are quoted from VolSplat (Wang et al., 2026a) unless noted. Our retrained VolSplat anchors all paired analyses. It is within 0.4 PSNR of the published model at 6 views (30.93 vs. 31.30) and 24 views (27.24 vs. 27.21) and stronger at 12 views (30.86 vs. 29.40), so every 12-view gain is measured against the retrained model.

Table 9: **Cumulative evolution to LoCoSplat** (150k steps, RealEstate10K, 6 views, 709 scenes). Enc.: encoder time on one RTX PRO 6000; intermediate models run their original code, LoCoSplat its full inference stack (Table 8).

Stage	PSNR↑	LPIPS↓	Enc. (ms)
VolSplat (sparse 3D U-Net, 17M)	30.93	0.078	137.0
→ 8-level grid + MLP	**31.64**	**0.073**	118
→ 2 levels $\{0.001, 0.128\}$	31.63	**0.072**	94.4
→ pointwise head, D=64	31.60	0.073	58.9
→ ViT-B/narrow iface, DWConv (= LoCoSplat)	31.51	0.074	**33.0**

Metrics and ties. PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018); efficiency as in Appendix A. Comparisons between our models use per-scene paired differences. Three seed draws of LoCoSplat give 31.51 / 31.50 / 31.50 (max paired difference 0.02, s.d. < 0.01), and three seed draws of the two-grid model lie pairwise within 0.03 (s.d. 0.01), so differences $\leq$0.03 PSNR are reported as ties.

C Analysis of the 3D Stage

C.1 The Path to the Two-Grid Model

An eight-level grid without a 3D network. The starting hypothesis was that VolSplat's single-resolution voxelization loses both fine detail and context, and that an Instant-NGP-style (Müller et al., 2022) multi-resolution grid, rebuilt from the images at every forward pass rather than learned, could keep both without a 3D network. Eight trilinear levels ($0.001 \cdot 2^{\ell}$) plus a 0.18M fusion MLP beat the 17M U-Net by $+0.71$ PSNR with 33% fewer trainable parameters and $1.67\times$ training throughput; the gap held at every checkpoint and widened with training, while the U-Net's test PSNR plateaued after $\sim$80k steps.

The coarse scale, not the level count, drives the gain. Varying only the grid levels of LoCoSplat at full budget (Table 12) shows that removing the coarse level is the one change that hurts (-0.13, $4\times$ replicate noise). Level count and coarsest scale within $[0.008, 0.128]$ do not matter: 2, 4, and 8 levels tie (31.49–31.51), as 2 vs. 8 levels did on the two-grid model (31.63 vs. 31.64, at 20% faster encoding). Only a *wider* window helps: 12 levels reaching 2.048 gain $+0.12$.

The coarse level is a fixed geometric low-pass. In Table 11, untrained box pooling or Gaussian smoothing of the fine field ties the learned re-splat, so the context needs neither learning nor smooth interpolation (hence the nearest-cell grids of LoCoSplat). A geometry-only descriptor with no appearance content also ties ($+0.030$): low-order geometric moments of the cell carry the coarse context. Only the gridless radius mean trails (-0.062), a kernel-choice effect. Removing the fusion MLP costs -0.044; we keep it for its negligible cost.

The two-grid model. It keeps VolSplat's depth, features, anchors, head, and renderer, and its 3D stage is a fine 0.001 grid, a coarse 0.128 grid, and a 0.16M fusion MLP, with no U-Net; its only remaining sparse convolution is the retained head. It reaches 31.63 PSNR at 94.4 ms (top row of Table 8); three training seeds lie pairwise within 0.03 PSNR (Appendix B.2).

C.2 Design Notes

Box (nearest-cell) and tent (trilinear) grids correspond to the simple and linear binning rules of binned kernel estimators (Hall & Wand, 1996), here applied to Eq. (2).

Receptive field of the U-Net. Applying $\mathrm{RF}_{\mathrm{out}} = \mathrm{RF}_{\mathrm{in}} + (k-1)j$ through VolSplat's sparse U-Net (three stride-2 encoder blocks, two bottleneck blocks, four transposed-conv decoder stages) gives 93 fine voxels, i.e. ± 0.046 around each anchor, whereas a 0.128 cell extends at least ± 0.064. The U-Net thus sees about a third of the context the coarse grid uses, and approximates the normalization by the cell's point count only through strided sums.

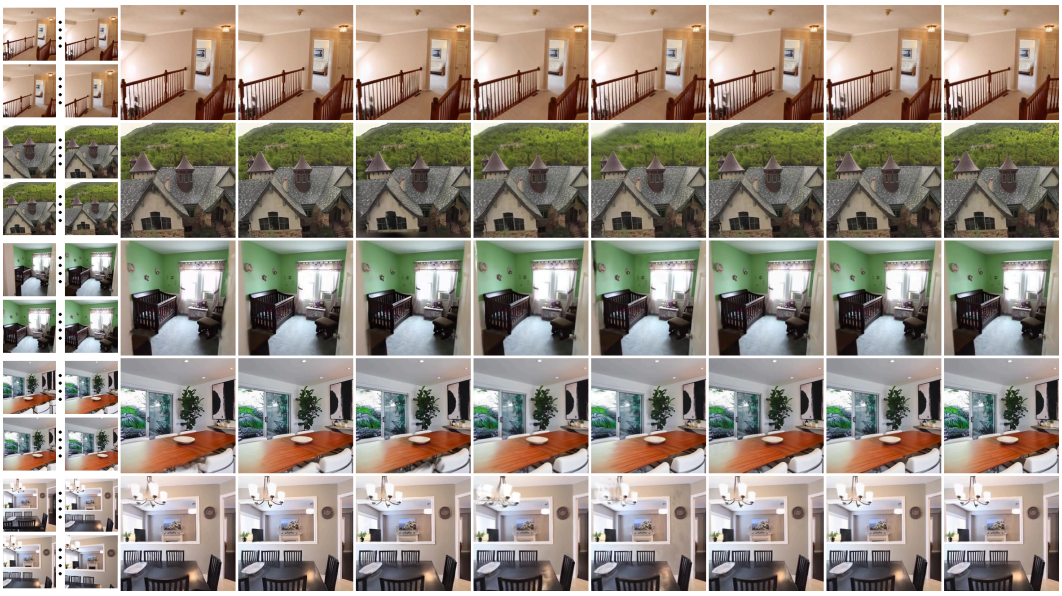

Figure 9: **More qualitative comparisons on RealEstate10K (Zhou et al., 2018), 6 input views** (four shown). Checkpoints and evaluation follow Fig. 3; selection is described in Appendix E.4. Red boxes highlight local differences.

The plain mean as default aggregator. Under a smooth-field-plus-homoscedastic-noise model, the cell mean in Eq. (3) is the minimum-variance unbiased linear estimate of the low-pass (Gauss–Markov). A reduced-budget pilot on the two-grid model (50k steps, one GPU, single seed) found no alternative aggregator that improved on it: confidence weighting by the detached matching probability hurt (-0.23 PSNR), and learned per-point weights, a per-cell variance channel, and a 3^3-cell neighbourhood query all tied within the seed noise of that budget.

Remark 1 (Sparse convolutions on isolated anchors). A sparse convolution $(\boldsymbol{W} * \boldsymbol{f})(a) = \sum_o \boldsymbol{W}_o \boldsymbol{f}_{a+o} \mathbf{1}[a + o \in \mathcal{A}]$ reduces to its centre tap $\boldsymbol{W}_0 \boldsymbol{f}_a$ whenever the 3^3 neighbourhood of a in the anchor set $\mathcal{A}$ is empty, so a stack of such layers interleaved with pointwise operations is a pointwise function at such anchors. Sparse-conv and pointwise heads therefore differ only on the fraction p of anchors with an occupied neighbour, empirically $p = 0.42\%$ at $v = 0.001$ and 6 views (Section 2.2). Hence LoCoSplat's pointwise head, which is quality-neutral at full budget (Table 9).

C.3 Ablation Tables

Table 12 (level count vs. scale range) and Table 11 (coarse source) are discussed in Appendix C.1; Table 10 gives the paired 12-view comparison against our retrained VolSplat.

Table 10: **LoCoSplat vs. VolSplat at 12 views.** RealEstate10K, 12 input views. Our retraining of VolSplat and LoCoSplat (which also carries the backbone-side changes of Section 2.1) on the official 12-view index; Δ is the per-scene paired difference against the retrained VolSplat. The full LoCoSplat pipeline, whose 3D stage is the two-level grid and a pointwise MLP, gains $+0.61$, a whole-pipeline gain (the isolated 3D-stage comparison is Table 9 at 6 views). Our 12-view retraining (30.86) is stronger than the published model (29.40, Appendix B.2).

Model (12 views)	3D stage	PSNR↑	SSIM↑	LPIPS↓	ΔPSNR
VolSplat (retrained)	sparse 3D U-Net	30.86	0.939	0.077	—
LoCoSplat	two-level grid + MLP	**31.47**	**0.942**	**0.073**	**+0.61** vs. retrained (paired)

Table 11: **Coarse-source ablation on the two-grid model** (Section 3.5: VolSplat's 164-d features, trilinear grids, sparse-convolutional head, $D{=}128$; full 150k budget; RealEstate10K, 6 views, 709 scenes). Its reference is therefore the two-grid model (31.60 / 31.63), not LoCoSplat (31.51, Table 12). The reference row (learned re-splat of the 16-d projected features) is the coarse level LoCoSplat ships. Paired ΔPSNR vs. the reference run. The reference row lists two training runs at the same seed (this code tree / earlier run); three training seeds of the two-grid model lie pairwise within 0.03 PSNR (Appendix B.2), which bounds the replicate noise (three draws of LoCoSplat give 0.02, Table 12), and differences within it are marked (tie).

Coarse source	Learned?	PSNR↑	Paired Δ vs. ref.
Learned re-splat (reference, 2 runs, same seed)	✓	31.60 / 31.63	—
Fine field, hard box pooling	×	31.58	-0.013 (tie)
Fine field, Gaussian low-pass	×	31.59	-0.005 (tie)
Fine field, anchor-centered radius mean	×	31.53	-0.062
Geometry-only statistics (no appearance)	×	**31.63**	$+0.030$ (tie)

Table 12: **Level count vs. scale range on the final LoCoSplat configuration at full budget** (only the grid levels change; RealEstate10K, 6 views, 709 scenes; the reference is LoCoSplat itself, 31.51, unlike Table 11). ΔPSNR is the difference of the 709-scene means vs. LoCoSplat, which on a common scene set equals the paired mean. (tie): within replicate noise, bounded by three seed draws of LoCoSplat (rows 2–4: largest paired difference 0.02 PSNR, sample s.d. < 0.01) and by three seed draws of the two-grid model of Table 11 ($\sim$0.03). Removing the coarse levels is the one change that hurts (1 level vs. 4 / 8 levels, paired: -0.123 / -0.113); four narrow levels, eight wide levels in two seed draws, and the two levels of LoCoSplat in three seed draws tie (8 vs. 4, paired: -0.010; 8 vs. 2 at the same seed: -0.006); extending the coarsest scale from 0.128 to 2.048 is the only change that helps.

Config	Scales	PSNR↑	SSIM↑	LPIPS↓	ΔPSNR vs. LoCoSplat
1 level (fine only)	0.001	31.38	0.941	0.075	-0.13
2 levels (LoCoSplat)	{0.001, 0.128}	31.51	0.942	0.074	—
2 levels (LoCoSplat, second seed)	{0.001, 0.128}	31.50	0.941	0.074	-0.01 (tie)
2 levels (LoCoSplat, third seed)	{0.001, 0.128}	31.50	0.941	0.074	-0.01 (tie)
4 levels (×2)	0.001–0.008	31.50	0.941	0.074	-0.01 (tie)
8 levels (×2)	0.001–0.128	31.49	0.941	0.074	-0.02 (tie)
8 levels (×2, second seed)	0.001–0.128	31.50	0.942	0.074	-0.01 (tie)
12 levels (×2)	0.001–2.048	**31.62**	**0.942**	**0.073**	$\mathbf{+0.12}$

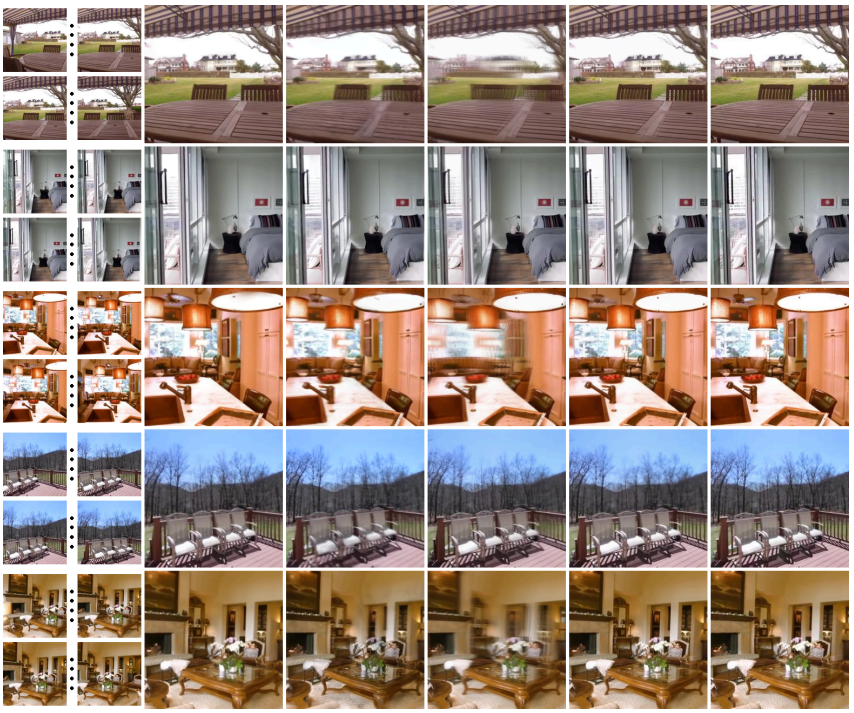

Figure 10: **More qualitative comparisons on RealEstate10K (Zhou et al., 2018), 24 input views** (four shown). Checkpoints and evaluation follow Fig. 8; selection is described in Appendix E.4. Red boxes highlight local differences.

D THE 24-VIEW DROP IN DETAIL

D.1 THE DROP LIES IN THE DEPTH BRANCH, NOT THE 3D STAGE

We retrained LoCoSplat at 24 views with one change reverted at a time (Table 5). Reverting the fine level to trilinear costs only 0.12 PSNR, but reverting the plane-sweep from D=64 to VolSplat's D=128 reproduces VolSplat's failure (28.08 vs. its 27.24 PSNR; -2.49 paired, worse on 98% of scenes), with the same loss plateau at twice the healthy level from the first 15k steps, even with the grids and pointwise head in place. Every degrading 24-view model uses D=128 and every healthy one D=64: the drop is an optimization failure of the multi-view depth branch, which LoCoSplat avoided incidentally through a change made for speed.

D.2 A DISPARITY-RESOLUTION ACCOUNT

With D candidates uniform in inverse depth on $[i_{\min}, i_{\max}]$, adjacent candidates are separated in a source view at baseline B by

$$\Delta u \;=\; f\,B\,\frac{i_{\max} - i_{\min}}{D-1}, \tag{7}$$

with f the focal length in feature pixels. The official indices pack more views into a fixed $\sim$48-frame span, so the neighbour spacing (and with it B) falls from 9.5 frames at 6 views to 4.4 and 2.1. Relative to 6 views with D=128, Δu is 0.46 (12v, D=128), 0.45 (24v, D=64), and 0.22 (24v, D=128): the two healthy dense settings share Δu, the failing one has half of it. We then pre-registered two predictions, both confirmed (Table 5): D=256 collapses the 12-view model ($\Delta u =$ 0.23; -3.10 paired PSNR, worse on 99% of scenes, same loss signature), while D=32 keeps the 24-view model healthy ($\Delta u = 0.91$; only -0.18, loss trajectory identical to D=64). Frame spacing is a faithful proxy: exact disparities from the evaluation cameras (0.21–0.91) match the spacing-based ones (0.22–0.92). Δu thus predicts the collapse across all trained configurations.

D.3 The Collapsed Depth Head

Collapsed models have a *switched-off* multi-view depth head (Table 13): all matching mass sits on one candidate, the same one at every pixel of every scene, so the head ignores its input, its depth is a constant 36–48% off the final depth, and the monocular path carries depth. Healthy models are peaked and input-dependent (multi-view depth within 1.5–3.2% of the final). Three facts point to optimization rather than a missing signal: (i) the absolute disparity is sub-pixel in all configurations (0.012–0.055 feature px), so no resolution threshold separates the groups, only the ratio does; (ii) the raw cost volume carries a matching signal everywhere, just spread over more bins when Δu is small (90%-mass width 55–74% of the range vs. 20–49%); (iii) at initialization all configurations are broad and alike, so the constant one-hot is learned.

Table 13: **The depth branch of healthy and collapsed models.** Δu: exact median per-candidate disparity in scale-0 feature pixels over the full official index (709/696/680 scenes). Remaining columns: scale-0 matching distribution on the first 50 scenes of the index, pooled over all pixels: normalized entropy $H/\log D$ (median), w_{90} = bins around the argmax holding 90% of the mass (median), share of pixels whose argmax is the most frequent bin, and median relative error of the multi-view depth against the final fused depth. Collapsed models are input-independent one-hots at a fixed candidate; at initialization every configuration is broad and alike. Two-grid model: Section 3.5.

Model	Views	D	Δu (px)	$H/\log D$	w_{90} (bins)	Modal-bin share	MV depth err.	Status
LoCoSplat	6	64	0.110	0.42	7	13%	1.8%	healthy
LoCoSplat	12	64	0.050	0.51	8	14%	2.0%	healthy
LoCoSplat, D=256	12	256	0.012	0.000	1	100%	48%	collapsed
LoCoSplat, D=32	24	32	0.047	0.51	5	19%	1.5%	healthy
LoCoSplat	24	64	0.023	0.47	8	14%	3.2%	healthy
LoCoSplat, D=128	24	128	0.012	0.000	1	100%	46%	collapsed
Two-grid model	6	128	0.055	0.39	8	7%	1.9%	healthy
Two-grid model	12	128	0.025	0.46	14	10%	2.5%	healthy
Two-grid model	24	128	0.012	0.000	1	100%	36%	collapsed
Any config. at init.	6–24	32–256	—	0.45–0.78	0.78–0.99 D	50–99%	—	untrained

E Additional Qualitative Results

E.1 Qualitative Results at 12 Input Views

All qualitative figures use one selection procedure: per 64×64 window (stride 32) we compute each baseline's RMS gradient error minus ours, take each baseline's worst window, and rank frames by the weakest of these gaps, so every baseline must show a localized failure. Frames where any baseline beats LoCoSplat in PSNR or where LoCoSplat is below 26 dB are discarded, and the top candidates are inspected zoomed. At 12 views (Fig. 7) the 6-view failure modes persist: pose-free models wash out text, masonry, and branches, and VolSplat produces fog and floaters over distant or textureless regions. MVSplat, DepthSplat, and GGN have only 6-view checkpoints and are omitted.

E.2 24-View Evaluation and Example Selection

All methods run on the same 680 evaluation batches: VolSplat and LoCoSplat with their 24-view checkpoints, AnySplat (448^2, rendering at its predicted, scale-aligned target cameras) and World-Mirror (518^2, ground-truth camera priors) along their official NVS paths. Selection follows Appendix E.1 (3,352 of 5,440 frames pass the gate; top 30 inspected). Rows 2–4 are ranks 1, 3, and 4; rows 1 and 5 come from an earlier pass against VolSplat alone and also pass the three-baseline gate.

E.3 Qualitative Results on ScanNet

The ScanNet examples of Fig. 5 were selected as in Appendix E.1.

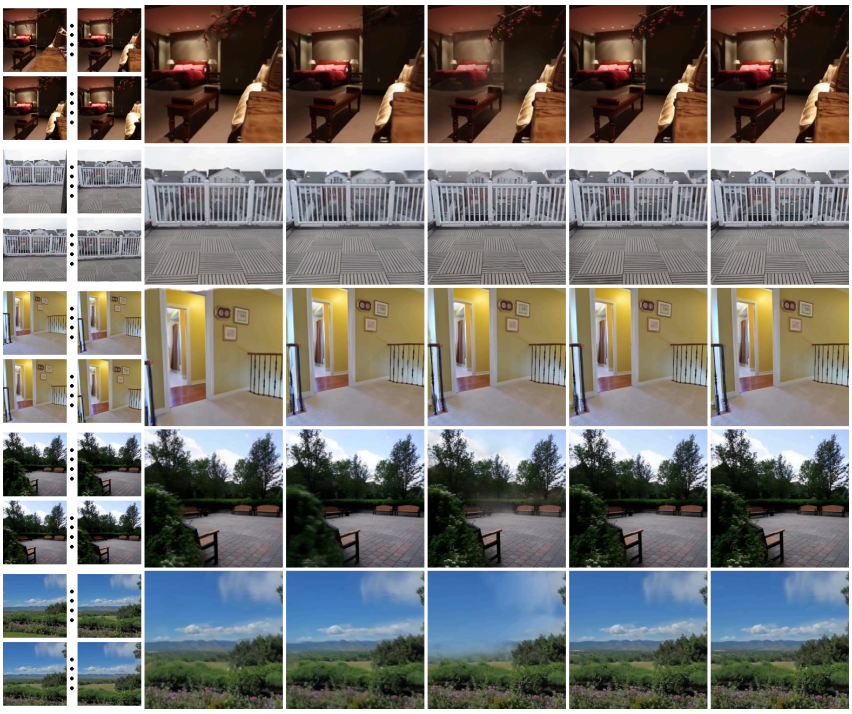

Figure 11: **More qualitative comparisons on RealEstate10K (Zhou et al., 2018), 12 input views** (four shown). Checkpoints and evaluation follow Fig. 7; selection is described in Appendix E.4. Red boxes highlight local differences.

E.4 More Qualitative Comparisons on RealEstate10K

Figs. 9 to 11 use the methods, checkpoints, and protocol of Figs. 3, 7 and 8 and the selection of Appendix E.1 (the gate is passed by 2,474/5,592, 2,810/5,528, and 3,241/5,328 frames at 6/12/24 views), excluding scenes already shown and favouring distinct failure modes among the top 40: blurred or duplicated thin structures for pixel-aligned and pose-free baselines, streaks and fog for VolSplat.