# CompactSplat: Spatially Adaptive Gaussian Distribution for Feedforward Scene Reconstruction

**Bing He [1] & Jingnan Gao [1] & Yunuo Chen [1]**
**& Zhengxue Cheng [1] & Rong Xie [1] & Li Song [1] & Wenjun Zhang [1]**
[1] Shanghai Jiao Tong University
{sandwich_theorem}@sjtu.edu.cn

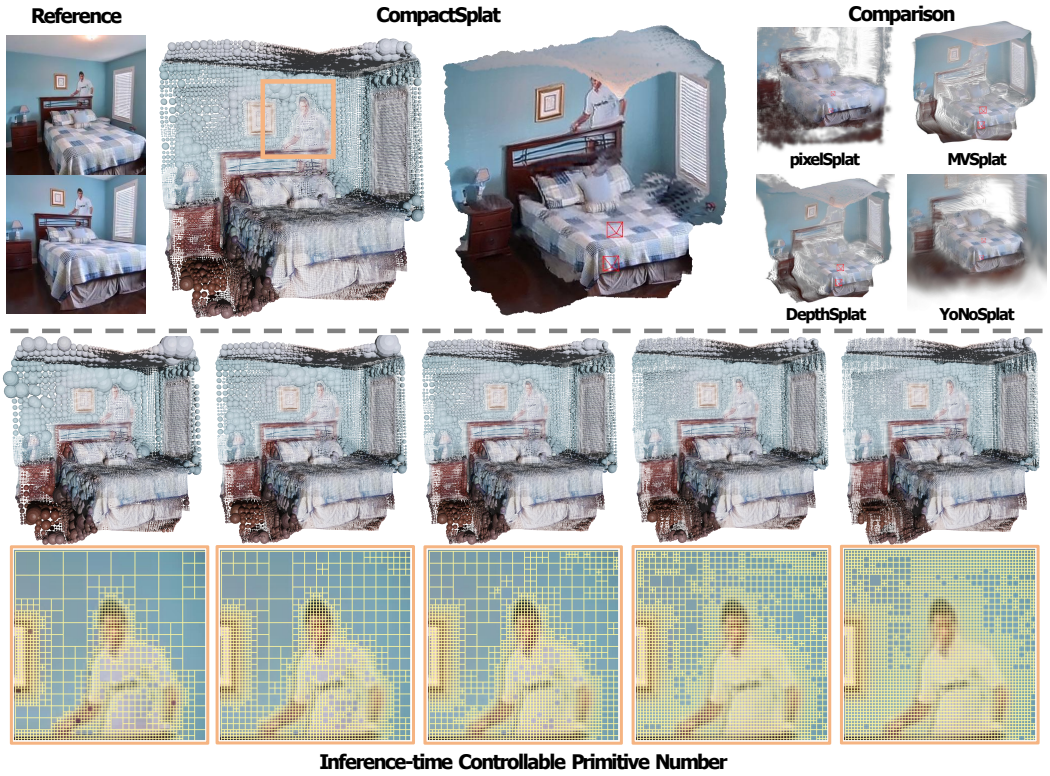


Figure 1: **CompactSplat** is a highly efficient feed-forward network that predicts 3D scene representations from sparse input images. While previous methods typically allocate one or more Gaussians per pixel—leading to excessive primitives and spatially redundant representations, CompactSplat adaptively allocates primitives based on regional complexity. It concentrates dense Gaussians in texturally rich areas while deploying expanded, scale-modulated primitives in flat regions. This paradigm enables high-fidelity reconstruction with substantially fewer primitives. Crucially, it inherently supports dynamic sparsity control at inference time, allowing seamless trade-offs between visual fidelity and memory footprint across diverse deployment constraints.

## Abstract

Reconstructing 3D scenes from sparse views without per-scene optimization remains highly challenging, especially for recovering accurate geometry and fine textures. While recent feedforward approaches leverage generalizable 3D Gaussian Splatting (3DGS) for scene generation, they typically assign one or multiple Gaussians to each pixel. Such uniform allocation not only produces highly redundant representations by wasting primitives in homogeneous areas, but also fails to ex-

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

ploit the inherent geometric priors of planar surfaces, often resulting in sub-optimal structural modeling. To address these limitations, we present **CompactSplat**, a novel feedforward framework that enables a content-aware allocation of 3D Gaussians. By integrating texture-guided spatial partitioning with hierarchical sampling, CompactSplat adaptively distributes primitives — concentrating dense Gaussians in complex regions while tiling flat areas with expanded, scale-modulated primitives. Moreover, our framework inherently supports dynamic sparsity control at inference time, enabling seamless trade-offs between rendering fidelity and memory footprint without requiring network retraining. Extensive experiments on RealEstate10K, DL3DV, and ScanNet demonstrate that CompactSplat consistently outperforms prior methods on both standard metrics and high-resolution rendering consistency, achieving high-fidelity reconstructions with significantly fewer primitives.

## 1 Introduction

Recovering geometrically accurate and visually consistent 3D scenes from sparse views is a fundamental problem for immersive VR, gaming, and digital content creation. The recent success of 3D Gaussian Splatting (3DGS) Kerbl et al. (2023) has significantly advanced this field by representing scenes as collections of 3D Gaussians, enabling real-time rendering with high visual fidelity.

Existing methods for constructing such representations generally follow two paradigms. Traditional approaches rely on dense multi-view captures and COLMAP-based Schönberger and Frahm (2016) point cloud initialization, followed by computationally expensive per-scene optimization. To avoid this bottleneck, feed-forward approaches have recently gained increasing attention. By leveraging pretrained models, they directly predict 3D Gaussians from sparse inputs in a single forward pass, eliminating per-scene optimization and enabling fast, scalable 3D reconstruction.

Despite their rapid inference, many existing feed-forward methods remain inefficient in their primitive allocation. Natural scenes exhibit substantial spatial variation in texture and geometric complexity, yet prior methods typically adopt a rigid pixel-wise prediction paradigm, assigning one Chen et al. (2024c); Xu et al. (2024b); He et al. (2026) or multiple Charatan et al. (2024); Tang et al. (2024) Gaussians to every pixel. This uniform allocation forces the network to populate large, textureless regions with thousands of primitives. Consequently, the generated 3D representations can suffer from substantial spatial redundancy. This redundancy increases representation storage costs and motivates more compact 3D priors for downstream tasks.

To address this issue, we introduce **CompactSplat**, a novel feed-forward framework for high-fidelity 3D reconstruction with content-aware Gaussian allocation. Instead of retaining a primitive at every pixel, CompactSplat allocates Gaussians according to image texture, using sparse primitives in homogeneous regions while concentrating denser predictions in areas with rich texture. To achieve this adaptive allocation, we decouple the primitive generation process into two complementary components. First, a **Texture-Aware Partitioner (TAP)** analyzes the input views to estimate regional texture complexity and partition each image with a quadtree. Second, a **Hierarchical Pyramid Sampler (HPS)** uses this guidance to sample explicit Gaussian attributes from a pooled multi-resolution attribute pyramid. Together, these components enable CompactSplat to adapt its primitive density to the input texture, reducing spatial redundancy while retaining strong reconstruction quality.

Crucially, CompactSplat provides a **unified paradigm for controllable inference**. Owing to the scale-aware synergy between TAP and HPS, our framework naturally supports dynamic sparsity control at inference time. By modulating a single partition threshold, the model can select different trade-offs between rendering fidelity and primitive count—**without requiring any network retraining or auxiliary modules.** This intrinsic adaptability provides an on-the-fly mechanism to select different representation sizes using a single set of pretrained weights.

Ultimately, CompactSplat establishes a more compact paradigm for 3D representation. By replacing rigid pixel-wise allocation with texture-driven primitive allocation, our approach adapts to spatial variations in image content. In our experiments, it delivers high-fidelity reconstructions with fewer primitives. This allocation transforms dense, redundant outputs into compact 3D representations with a tunable primitive count.

In summary, our contributions are as follows:

- We propose **CompactSplat**, a feed-forward framework for compact sparse-view 3D reconstruction. By combining a Texture-Aware Partitioner (TAP) with a Hierarchical Pyramid Sampler (HPS), our method adaptively allocates 3D Gaussians based on image texture, significantly reducing spatial redundancy.
- We introduce a **controllable inference paradigm** for feed-forward 3D Gaussian Splatting. With a single partition threshold, CompactSplat supports dynamic, on-the-fly sparsity control. This capability enables seamless trade-offs between visual fidelity and primitive count using a single set of network weights, completely eliminating the need for sparsity-specific retraining.
- Extensive experiments on RealEstate10K and ACID demonstrate that CompactSplat achieves **competitive quality with fewer primitives**. It retains strong performance on both standard metrics and high-resolution rendering consistency, while cross-dataset evaluation on DTU, DL3DV, and ScanNet reveals quality–compactness trade-offs.

## 2 Related Works

### 2.1 3D Gaussian Splatting

Since the advent of Neural Radiance Fields (NeRF) Mildenhall et al. (2021), novel view synthesis has advanced rapidly, evolving from implicit MLPs to grid-based Müller et al. (2022); Fridovich-Keil et al. (2022) and anti-aliased Barron et al. (2021, 2022) representations. To further accelerate rendering, recent paradigms shifted towards explicit point-based architectures. Notably, 3D Gaussian Splatting (3DGS) Kerbl et al. (2023) revolutionized the field by modeling scenes with semi-transparent, anisotropic ellipsoids, enabling photorealistic, real-time rendering via differentiable rasterization. Consequently, a proliferation of extensions has emerged to refine 3DGS. These include optimizations for training stability Cheng et al. (2024); Zhang et al. (2024); Radl et al. (2024); Diolatzis et al. (2024), anti-aliasing techniques Yan et al. (2024); Yu et al. (2024); Song et al. (2024); Liang et al. (2024), and adaptations for dynamic scenes Luiten et al. (2023); Wu et al. (2023); Wan et al. (2024); Huang et al. (2023); He et al. (2024). Additionally, several works focus on compressing these representations to reduce storage and accelerate inference Girish et al. (2024); Navaneet et al. (2024); Niedermayr et al. (2024); Lee et al. (2024); Fan et al. (2024); Chen et al. (2024b).

### 2.2 Generalizable 3D Reconstruction

To bypass the computationally expensive per-scene optimization of traditional 3DGS, recent literature has explored feedforward models that directly predict Gaussian attributes from sparse inputs. Splatter Image Szymanowicz et al. (2024) pioneered the translation of 2D images into Gaussian parameter maps. Subsequent frameworks integrated geometric priors to enhance spatial understanding; for instance, pixelSplat Charatan et al. (2024) utilized epipolar geometry, while MVSplat Chen et al. (2024c) and DepthSplat Xu et al. (2024b) leveraged cost volumes and cross-task depth interactions. Further advancements addressed specific challenges, such as expanding synthesis ranges Wang et al. (2024b), improving geometric surfaces via 2DGS He et al. (2026), and handling unposed inputs Kang et al. (2025); Jiang et al. (2025). VGGT Wang et al. (2025) and MV-DUSt3R Tang et al. (2025) likewise reconstruct uncalibrated views, while CompactSplat assumes known camera intrinsics and extrinsics. Other pipelines introduced triplane representations Zou et al. (2024); Xu et al. (2024a) and canonical space generation Ye et al. (2024) for feedforward 3D reconstruction and generation. SplatFormer Chen et al. (2024a) instead uses a point transformer to refine an already optimized Gaussian scene for novel-view extrapolation.

Pixel-wise prediction strategies, including multi-layered approaches like HiSplat Tang et al. (2024), can generate redundant primitives in homogeneous regions. Recent methods explore alternatives: TokenGS Ren et al. (2026) uses learnable queries to decouple Gaussian count from image resolution and view count, while C3G An et al. (2025) predicts a compact Gaussian set from unposed multi-view images using learned queries. VolSplat Wang et al. (2026) lifts features using predicted depth and decodes Gaussians from occupied voxels. Off The Grid Moreau et al. (2026) detects subpixel primitive positions and adapts patch density using image entropy. CompactSplat adopts a **content-aware allocation paradigm** based on an image-space quadtree. TAP concentrates finer partitions in textured regions, while HPS samples pooled Gaussian attributes and modulates primitive scales according to block footprints. This spatial hierarchy supports inference-time quality–compactness control with one checkpoint, reducing per-view redundancy while preserving detail at the selected operating point.

# 3 Method

## 3.1 Model Architecture

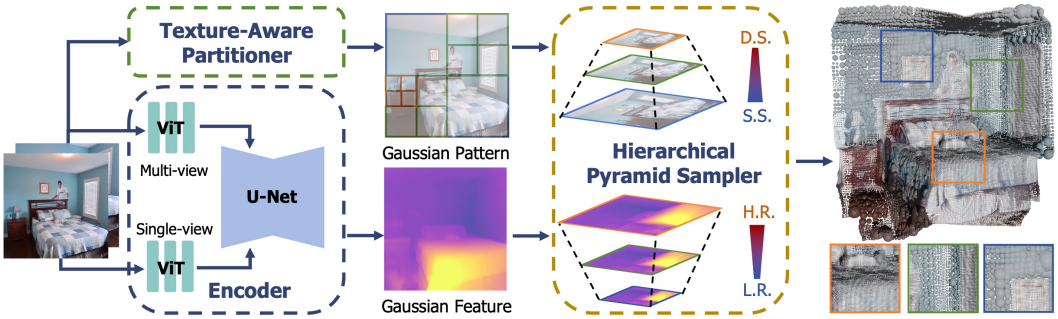


Figure 2: **Overview of CompactSplat.** A dual-path encoder first extracts and fuses multi-view and single-view features from sparse inputs, which are then processed by a multi-scale U-Net. Concurrently, the Texture-Aware Partitioner (TAP) evaluates scene complexity based on a user-defined threshold to generate multi-level spatial masks. Guided by these masks, the Hierarchical Pyramid Sampler (HPS) adaptively extracts features across decoder levels. Ultimately, this paradigm allocates sparse samples (S.S.) from low-resolution (L.R.) maps to flat regions, and dense samples (D.S.) from high-resolution (H.R.) maps to texturally complex areas.

Building upon the robust feature extraction paradigms established in recent feed-forward models He et al. (2026); Xu et al. (2024b), CompactSplat employs a dual-path encoder to capture both multi-view correspondences and single-view geometric priors. For the single-view branch, we utilize a pretrained monocular depth backbone (e.g., Depth Anything V2 Yang et al. (2024)) to extract structural and semantic cues, which are particularly beneficial for ill-posed regions lacking cross-view overlaps. Concurrently, the multi-view branch processes the inputs through self- and cross-attention mechanisms Vaswani et al. (2017); Liu et al. (2021b) to construct plane-sweep cost volumes Chen et al. (2024c); Collins (1996), enforcing strict multi-view consistency. The complementary features from both branches are then concatenated to form a unified volumetric representation.

This fused feature volume is subsequently fed into a multi-scale 2D U-Net Ronneberger et al. (2015) architecture. Unlike prior works that deterministically upsample all features into a single, full-resolution map for pixel-wise Gaussian prediction, our architecture decouples primitive generation to eliminate spatial redundancy, as illustrated in Fig. 2. Specifically, we introduce the Texture-Aware Partitioner (TAP), which analyzes spatial frequency and regional complexity from the inputs to produce a set of hierarchical partition masks. These masks dynamically dictate the appropriate Level of Detail (LoD) for representing different regions of the scene.

Guided by the TAP masks, our Hierarchical Pyramid Sampler (HPS) predicts explicit Gaussian attributes—including 3D center, scale, rotation, opacity, and spherical harmonics—directly from the multi-scale intermediate features of the U-Net decoder. Crucially, large, homogeneous regions are assigned to lower-resolution feature levels, allowing a single, scale-modulated Gaussian to seamlessly tile broad planar surfaces. In contrast, structurally complex areas are routed to the highest-resolution decoder features to preserve intricate geometric details. This multi-scale routing paradigm not only captures fine geometry but also naturally enables dynamic sparsity control, resulting in a highly compact yet expressive 3D representation.

The core motivation behind the Texture-Aware Partitioner (TAP) is to replace redundant pixel-wise generation with adaptive, density-aware primitive allocation. To this end, we formulate the image partitioning as a top-down quadtree decomposition driven by local textural complexity. This design is motivated by the inherent coupling of geometry and appearance within a single 3D Gaussian primitive. In natural environments, geometric complexity (e.g., depth discontinuities, occlusions, and structural boundaries) almost inevitably manifests as high-frequency textural variance. Although the converse is not strictly true—intricate textures frequently lie on structurally flat surfaces—relying on textural complexity as a subdivision proxy guarantees a conservative and sufficient upper bound. It ensures that all complex geometries receive dense primitive allocation, while aggressively pruning homogeneous regions where neither geometry nor texture demands high-resolution modeling.

## 3.2 Texture-Aware Partitioner (TAP)

Given an input image $I \in \mathbb{R}^{C\times H\times W}$ and a predefined maximum block size $S_{\text{max}}$, TAP first initializes the image plane as a coarse grid of $S_{\text{max}} \times S_{\text{max}}$ patches. For each block $B$, we evaluate its internal textural homogeneity. Specifically, we define the intra-block complexity based on the maximum channel-wise intensity spread. Let $I_c(x, y)$ denote the pixel value at spatial coordinate $(x, y)$ for channel $c$. The splitting criterion for block $B$ is determined by calculating the maximum difference:

$$\Delta(B) = \max_{c\in[1,C]} \left( \max_{(x,y)\in B} I_c(x, y) - \min_{(x,y)\in B} I_c(x, y) \right) \tag{1}$$

To realize dynamic sparsity control, we introduce a user-defined partition threshold $\tau$. If the complexity $\Delta(B)$ exceeds $\tau$ and the block dimensions are strictly greater than $1 \times 1$, the region is considered texturally complex. Consequently, block $B$ is recursively subdivided into four equal-sized child quadrants. This recursive subdivision continues until the block satisfies $\Delta(B) \leq \tau$ (indicating a smooth, homogeneous region) or reaches the single-pixel limit.

**Algorithm 1:** Texture-Aware Partitioning

**Input:** Input Image $I$, Partition Threshold $\tau$, Max Block Size $S_{max}$
**Output:** Partition Set $\mathcal{P} = \emptyset$

Initialize queue $Q$ with initial blocks $B = (\boldsymbol{\mu}, \boldsymbol{b}, l)$, where $\boldsymbol{b} = S_{max} \times S_{max}$ and $l = 0$
Initialize partition set $\mathcal{P} \leftarrow \emptyset$
**while** $Q$ is not empty **do**
  $B \leftarrow Q$.pop()
  Compute complexity $\Delta(B)$
  **if** $\Delta(B) > \tau$ **and** $\boldsymbol{b} > 1 \times 1$ **then**
    Split $B$ into 4 sub-quadrants $\{B_{child}\}$
    $Q$.push($\{B_{child}\}$)
  **else**
    $\mathcal{P} \leftarrow \mathcal{P} \cup B$
  **end if**
**end while**
**return** $\mathcal{P}$

The resulting leaf nodes of this quadtree represent the final spatially adaptive partition. For each leaf node $k$, TAP outputs its normalized 2D center coordinate $\boldsymbol{\mu}_k \in [-1, 1]^2$, spatial dimensions $\boldsymbol{b}_k \in [1, S_{\text{max}}]^2$, and a Level of Detail (LoD) index $l_k \in [0, \log_2(S_{\text{max}}) - 1]$ corresponding to the depth of the subdivision. To improve robustness, especially near image boundaries and invalid view regions, we apply a binary mask to filter out non-informative nodes.

**This quadtree-based formulation naturally resolves spatial redundancy**: flat regions terminate early at lower LoD levels, yielding fewer and larger blocks, whereas complex high-frequency regions are further subdivided into finer blocks at higher LoD levels. Importantly, the threshold $\tau$ serves as a dynamic control knob: adjusting $\tau$ at inference time directly modulates the overall primitive count without requiring retraining. The resulting tuples $\{(\boldsymbol{\mu}_k, \boldsymbol{b}_k, l_k)\}$ are then passed to the Hierarchical Pyramid Sampler to guide feature extraction. **This quadtree-based formulation naturally resolves spatial redundancy**: flat regions terminate early at lower LoD levels, yielding fewer and larger blocks, whereas complex high-frequency regions are further subdivided into finer blocks at higher LoD levels.

### 3.3 Hierarchical Pyramid Sampler (HPS)

While the Texture-Aware Partitioner (TAP) determines the optimal spatial layout and density of the primitives, the Hierarchical Pyramid Sampler (HPS) maps these abstract spatial partitions into explicit 3D Gaussian attributes. To this end, HPS performs scale-conditioned sampling from a foundational scene representation.

**Algorithm 2:** Hierarchical Pyramid Sampler

**Input:** Base attribute map $\mathbf{A}_{base}$, Partition Set $\mathcal{P}$, Max pyramid levels $L$
**Output:** 3D Gaussians Set $\mathcal{G} = \emptyset$

*% 1. Construct Attribute Scale Space (Pyramid)*
$\mathcal{A}[L-1] \leftarrow \mathbf{A}_{base}$
**for** $l = L - 1$ **to** 0 **do**
  $\mathcal{A}[l-1] \leftarrow \text{AveragePool}(\mathcal{A}[l])$
**end for**
*% 2. Hierarchical Pyramid Sampling*
**for each** $(\boldsymbol{\mu}_k, \mathbf{b}_k, l_k) \in \mathcal{P}$ **do**
  $\mathbf{a}_k \leftarrow \text{GridSample}(\mathcal{A}[l_k], \boldsymbol{\mu}_k)$
  $\boldsymbol{\mu}_k, \mathbf{r}_k, \Delta\mathbf{s}_k, \boldsymbol{\alpha}_k, \mathbf{c}_k \leftarrow \text{Decode}(\mathbf{a}_k)$
  $\mathbf{s}_k \leftarrow \Delta\mathbf{s}_k \odot \mathbf{b}_k$
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{\text{Gaussian}(\boldsymbol{\mu}_k, \mathbf{s}_k, \mathbf{r}_k, \boldsymbol{\alpha}_k, \mathbf{c}_k)\}$
**end for**
**return** $\mathcal{G}$

Instead of forcing the network to implicitly predict multi-scale attributes across different decoder layers, our architecture utilizes a DPT head to predict a unified, high-fidelity Gaussian attribute field $\mathbf{A}_{\text{base}}$ at full image resolution. However, directly sampling from this high-frequency field for large, homogeneous partitions would inherently lead to aliasing artifacts and a mismatch in the receptive field. To address this, we construct a discrete attribute scale space $\mathcal{A} = \{\mathbf{A}_{L-1}, \ldots, \mathbf{A}_1, \mathbf{A}_0\}$. Starting with the full-resolution field $\mathbf{A}_{L-1} = \mathbf{A}_{\text{base}}$, each subsequent coarser level $\mathbf{A}_l$ is derived by progressively applying a spatial low-pass filter (e.g., local aggregation). This multi-scale representation ensures that features at lower Level of Detail (LoD) indices naturally encapsulate broader regional contexts, perfectly mirroring the TAP quadtree hierarchy.

Given a quadtree leaf node $k$ from TAP, parameterized by its normalized center $\boldsymbol{\mu}_k \in [-1, 1]^2$, spatial footprint $\boldsymbol{b}_k$, and LoD index $l_k$, HPS performs discrete-to-continuous feature querying. By routing the sampling operation to the corresponding scale space level $\mathbf{A}_{l_k}$, the extracted attribute vector $\mathbf{a}_k$ is intrinsically aligned with the spatial frequency of the partition:

$$\mathbf{a}_k = \text{GridSample}(\mathbf{A}_{l_k}, \boldsymbol{\mu}_k) \tag{2}$$

A central design choice of HPS is its scale-aware geometric decoding. The queried vector $\mathbf{a}_k$ encapsulates base properties such as opacity, rotation, spherical harmonics, and a relative scale multiplier $\Delta\mathbf{s}_k$. To guarantee that the allocated primitive physically spans its designated spatial partition, the final scale of the Gaussian, denoted as $\mathbf{s}_k$, is explicitly modulated by the quadtree node dimensions:

$$\mathbf{s}_k = \Delta\mathbf{s}_k \odot \boldsymbol{b}_k \tag{3}$$

This explicit scale conditioning elegantly bridges the gap between 2D partition logic and 3D geometry. It enables a single Gaussian, sampled from a deeper level of the scale space, to morphologically expand and seamlessly tile expansive planar surfaces. This fundamentally breaks the representational redundancy of rigid pixel-wise predictions while rigorously preserving the rendering quality of high-frequency intricate details. Moren details are provided in the appendix.

### 3.4 Training Strategy

A core objective of CompactSplat is to establish a unified paradigm for controllable inference, enabling dynamic trade-offs between rendering fidelity and sparsity without network retraining. To achieve this, the model must be exposed to varying partition thresholds $\tau$ during the training process. However, jointly optimizing for multi-view geometric consistency and dynamic primitive pruning from scratch introduces severe training instability. To mitigate this, we propose a tailored two-stage curriculum learning strategy.

**Stage 1: Dense Geometric Initialization.** In the initial phase, we strictly fix the partition threshold to $\tau = 0$. Under this condition, the Texture-Aware Partitioner (TAP) behaves conservatively, decomposing the feature maps at the highest Level of Detail (LoD) across the entire scene. This dense supervision isolates the learning objective, forcing the dual-path encoder and the multi-scale U-Net to focus exclusively on establishing robust multi-view correspondences and extracting accurate geometric priors, completely free from the confounding effects of adaptive sparsity.

**Stage 2: Sparsity-Aware Fine-Tuning.** Once the network has converged to a reliable geometric understanding, we introduce stochastic thresholding. In this subsequent stage, $\tau$ is randomly sampled from a predefined uniform distribution $\tau \sim \mathcal{U}(0, \tau_{\text{max}})$ at each training iteration. This strategy exposes the model to a continuous spectrum of pruning ratios. Crucially, it forces the Hierarchical Pyramid Sampler (HPS) to accurately learn the geometric scale multipliers for lower-resolution Gaussians, ensuring they can morphologically expand to seamlessly tile broad, homogeneous regions under aggressive pruning scenarios. Through this two-stage curriculum, a single set of pretrained weights inherently acquires the versatility to reconstruct scenes at arbitrary sparsity levels on the fly.

### 3.5 Training Loss

To optimize our feed-forward architecture, we apply direct photometric supervision between the synthesized views and the corresponding target images. The overall objective function is formulated as a joint optimization of pixel-level fidelity and perceptual quality. Specifically, for a training batch containing $M$ samples, the loss is defined as:

$$L_{\text{total}} = \sum_{m=1}^{M} \left( \text{MSE}\left(I^m_{\text{render}}, I^m_{\text{gt}}\right) + \lambda \cdot \text{LPIPS}\left(I^m_{\text{render}}, I^m_{\text{gt}}\right) \right), \tag{4}$$

where $I^m_{\text{render}}$ and $I^m_{\text{gt}}$ denote the $m$-th rendered output and its corresponding ground truth, respectively. Consistent with established training protocols in recent generalizable 3DGS frameworks Charatan et al. (2024); Chen et al. (2024c); Xu et al. (2024b); He et al. (2026), the scaling factor $\lambda$ for the perceptual loss is empirically set to 0.05.

# 4 Experiment

## 4.1 Main Results

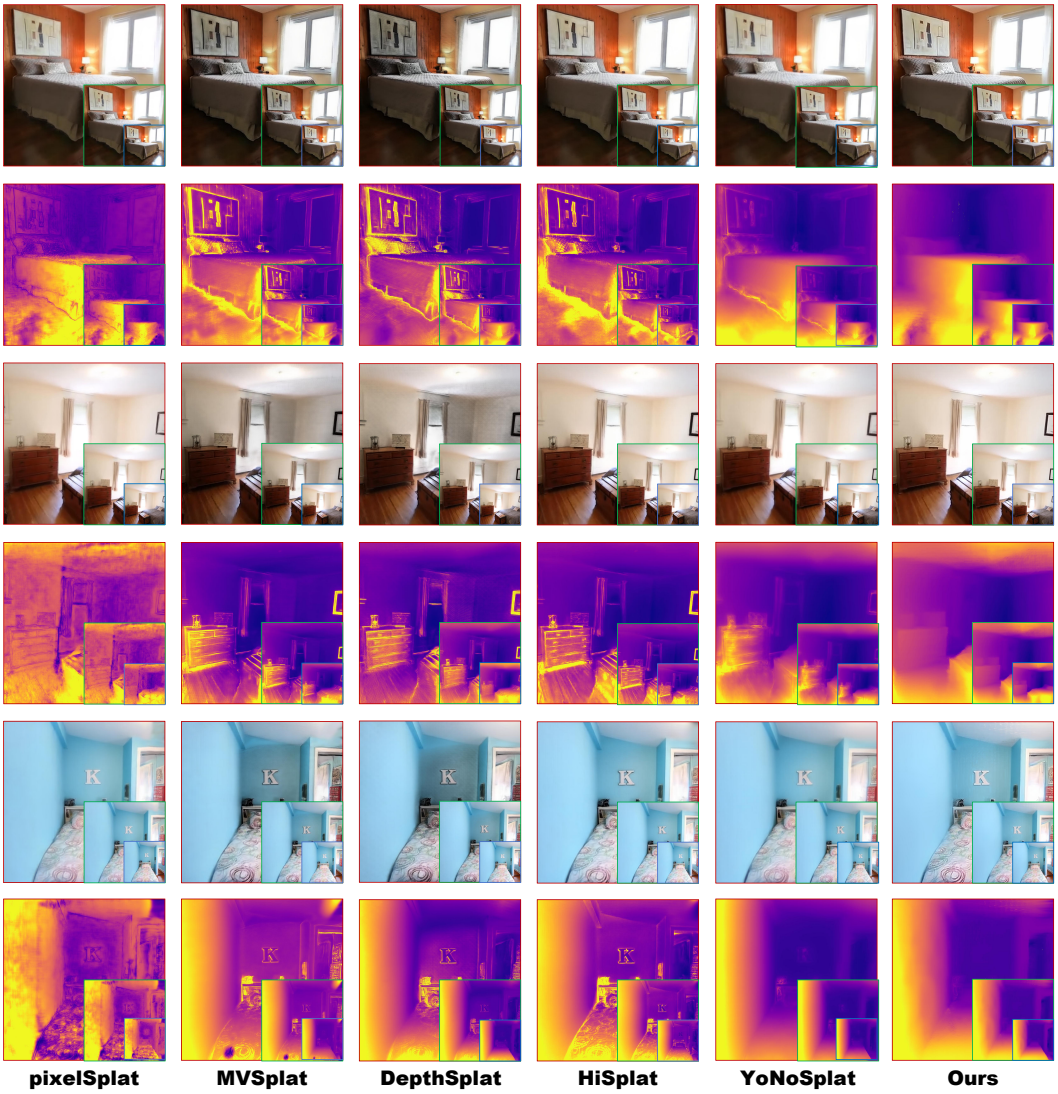


Figure 3: **Multi-resolution rendering comparison of 3D scenes.** We visualize rendered RGB images and depth maps at three scaling factors: $\times 1$ (blue box), $\times 2$ (green box), and $\times 4$ (red box). Under higher magnification, structural deficiencies in conventional 3D representations become increasingly apparent. In the RGB domain, primitive sparsity causes unfilled gaps, manifesting as dark, hollow regions. Correspondingly, in the depth space (where yellow indicates near and blue indicates distant surfaces), these geometric discontinuities erroneously register as spurious near-field artifacts. In contrast, CompactSplat's scale-modulated primitives seamlessly tile the surfaces, yielding watertight geometry and artifact-free renderings across all resolutions.

**Datasets.**To comprehensively evaluate our approach, we benchmark CompactSplat across a diverse set of environments commonly utilized in generalizable 3DGS literature. Specifically, we conduct experiments on RealEstate10K (RE10K) Zhou et al. (2018) for indoor scenes and ACID Liu et al. (2021a) for outdoor aerial trajectories, strictly adhering to the official train-test splits and precomputed camera poses established by prior works. To further validate the cross-dataset generalization and the robustness of our context-aware partitioning on complex geometries, we extend our evaluation to the DTU Jensen et al. (2014), DL3DV Ling et al. (2024), and the highly challenging ScanNet Dai et al. (2017) datasets. Comprehensive cross-dataset evaluations are detailed in the supplementary material.

**Evaluation Metrics.** We quantify the fidelity of novel view synthesis using standard image quality assessment metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and

Table 1: Novel view synthesis performance on the RealEstate10k dataset.

| Method | 256×256 (**Standard**) | | | 512×512 (**HRRC**) | | | 1024×1024 (**HRRC**) | | | Average | | | CPT | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | GPP↓ | GS Num↓ |
| pixelSplat 2024 | 26.049 | 0.862 | 0.137 | 25.782 | 0.868 | 0.207 | 24.920 | 0.877 | 0.269 | 25.584 | 0.869 | 0.204 | 3.000 | 393.2k |
| HiSplat 2024 | 27.193 | 0.882 | 0.117 | 25.269 | 0.870 | 0.198 | 24.262 | 0.878 | 0.248 | 25.575 | 0.877 | 0.188 | 1.750 | 229.4k |
| MVSplat 2024c | 26.359 | 0.868 | 0.129 | 20.408 | 0.809 | 0.290 | 17.966 | 0.755 | 0.425 | 21.578 | 0.811 | 0.281 | 1.000 | 131.1k |
| TranSplat 2025 | 26.687 | 0.875 | 0.125 | 20.610 | 0.815 | 0.286 | 18.154 | 0.761 | 0.427 | 21.817 | 0.817 | 0.279 | 1.000 | 131.1k |
| DepthSplat 2024b | 27.504 | 0.890 | **0.112** | 20.031 | 0.774 | 0.341 | 16.385 | 0.635 | 0.491 | 21.307 | 0.766 | 0.315 | 1.000 | 131.1k |
| SurfSplat 2026 | 27.537 | 0.892 | **0.112** | 26.331 | 0.866 | 0.217 | 24.897 | 0.842 | 0.320 | 26.255 | 0.867 | 0.216 | 1.000 | 131.1k |
| YonoSplat* 2025 | 25.735 | 0.851 | 0.130 | 25.415 | 0.857 | 0.205 | 24.785 | **0.872** | **0.261** | 25.312 | 0.860 | 0.199 | 1.000 | 131.1k |
| Ours-S | 27.086 | 0.884 | 0.123 | 26.066 | 0.866 | 0.205 | 25.118 | 0.857 | 0.276 | 26.090 | 0.869 | 0.201 | **0.554** | **72.6k** |
| Ours-B | 27.523 | 0.891 | 0.118 | 26.332 | 0.870 | **0.202** | 25.240 | 0.858 | 0.276 | 26.365 | 0.873 | 0.199 | **0.554** | **72.6k** |
| Ours-L | **27.567** | **0.892** | 0.117 | **26.357** | **0.871** | **0.202** | **25.252** | 0.858 | 0.276 | **26.392** | **0.874** | **0.198** | **0.554** | **72.6k** |

*Note: Due to architectural constraints, YonoSplat is evaluated at $224 \times 224$ resolution, which may yield marginally inflated metrics compared to the standard $256 \times 256$ resolution used by other baselines.

Table 2: Novel view synthesis performance on the ACID dataset.

| Method | 256×256 (**Standard**) | | | 512×512 (**HRRC**) | | | 1024×1024 (**HRRC**) | | | Average | | | CPT | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | GPP↓ | GS Num↓ |
| pixelSplat 2024 | 28.284 | 0.842 | 0.146 | 27.687 | 0.848 | 0.243 | 26.462 | 0.858 | 0.343 | 27.478 | 0.849 | 0.244 | 3.000 | 393.2k |
| HiSplat 2024 | 28.737 | 0.853 | 0.132 | 25.376 | 0.833 | 0.246 | 23.988 | 0.841 | 0.314 | 25.700 | 0.842 | 0.231 | 1.750 | 229.4k |
| MVSplat 2024c | 28.202 | 0.842 | 0.145 | 17.802 | 0.711 | 0.406 | 14.784 | 0.572 | 0.567 | 20.263 | 0.708 | 0.373 | 1.000 | 131.1k |
| TranSplat 2025 | **28.337** | **0.845** | **0.143** | 17.911 | 0.716 | 0.402 | 14.956 | 0.582 | 0.558 | 20.401 | 0.714 | 0.373 | 1.000 | 131.1k |
| SurfSplat 2026 | 28.336 | **0.845** | 0.144 | **26.868** | **0.814** | 0.281 | 21.253 | 0.690 | 0.457 | 25.486 | 0.783 | 0.294 | 1.000 | 131.1k |
| Ours | 27.992 | 0.837 | 0.157 | 26.384 | 0.810 | **0.271** | **24.063** | **0.782** | **0.376** | **26.146** | **0.810** | **0.268** | **0.625** | **91.4k** |

Learned Perceptual Image Patch Similarity (LPIPS). Furthermore, to validate the structural integrity and watertight properties of our representations, we employ the High-Resolution Rendering Consistency (HRRC) metric He et al. (2026) at both $2\times$ and $4\times$ spatial magnifications. Crucially, to demonstrate the efficiency of our feedforward architecture, we introduce two representation-specific metrics: the average **Number of Gaussians(GS Num.)**, and the **Gaussian Per Pixel(GPP)**, which explicitly highlights the **compactness(CPT)** of primitives compared to the unpruned baseline.

**Baselines.**Our framework is compared against leading sparse-view feedforward 3DGS models, including PixelSplat Charatan et al. (2024), MVSplat Chen et al. (2024c), TranSplat Zhang et al. (2025), HiSplat Tang et al. (2024), DepthSplat Xu et al. (2024b), SurfSplat He et al. (2026) and YonoSplat Ye et al. (2025). While architectures like PixelSplat and HiSplat indiscriminately predict multiple Gaussians per pixel, methods such as MVSplat, TranSplat, DepthSplat, SurfSplat and YonoSplat restrict their generation to a single primitive per pixel. Because our core contribution lies in decisively breaking the 1:1 pixel-to-primitive barrier to eliminate spatial redundancy, benchmarking against this latter group establishes a highly rigorous baseline for evaluating representational efficiency.

**Qualitative Comparisons.** Visual inspections of the rendered novel views in Figure 3 demonstrate that CompactSplat achieves photorealistic quality that is highly competitive with, and frequently surpasses, current state-of-the-art feed-forward methods. This superiority becomes particularly evident under high-resolution magnification. While baseline models often struggle with spatial voids, noisy boundaries, and spurious artifacts due to their rigid primitive allocation, CompactSplat maintains strict structural integrity. By leveraging complexity-aware allocation, our method seamlessly tiles flat planar surfaces with scale-modulated primitives while accurately preserving high-frequency edges. Consequently, it yields strictly watertight geometries and visually coherent structures that remain exceptionally robust even upon close inspection. Additional qualitative comparisons across various scenes are provided in our **supplementary video**.

**Quantitative Results.** The objective metrics presented in Table 1 and Table 2 further corroborate the transformative efficiency of our architecture. Across standard fidelity metrics (e.g., PSNR, SSIM, LPIPS) and the structure-aware HRRC metric, CompactSplat consistently matches or exceeds the performance of leading baselines. Crucially, it achieves this high-fidelity synthesis while drastically reducing the total Number of Gaussians. By pioneering a departure from the conventional pixel-wise prediction paradigm, CompactSplat establishes a highly compact representation. Furthermore, this compactness is inherently controllable; our architecture allows users to dynamically modulate the pruning ratio on the fly, offering an unprecedented capability to balance rendering quality and memory footprint without requiring any network retraining.

### 4.2 Analysis of Dynamic Sparsity Control

A core capability of CompactSplat is its inherent support for dynamic sparsity control without requiring network retraining. To investigate the efficacy of this mechanism, we systematically analyze the trade-off between reconstruction quality and representational compactness by modulating the partition threshold $\tau$ during inference, which effectively controls the average Gaussians Per Pixel.

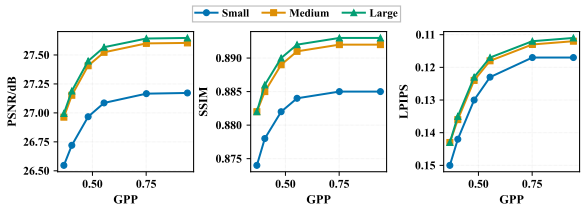


Figure 4: **Quantitative analysis.** By modulating the partition threshold $\tau$ at inference time, CompactSplat continuously traverses the Pareto frontier. The model achieves extreme reductions in primitive count (measured in Gaussian per pixel, GPP) with virtually negligible degradation in rendering fidelity.

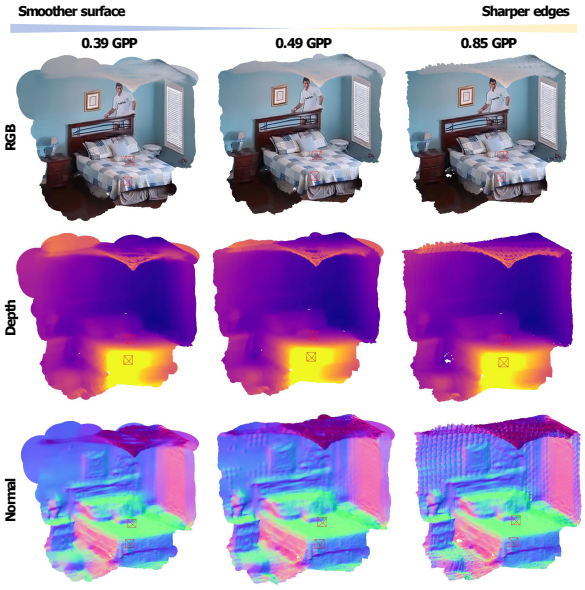


Figure 5: **Qualitative analysis.** Visualizations under varying GPP settings reveal an elegant architectural trade-off. Lower GPP limits enforce the use of expanded, scale-modulated primitives, yielding smoother, watertight planar surfaces. Conversely, higher GPP settings allocate denser primitives to more faithfully recover high-frequency edges and complex boundaries.

**Quantitative Trade-offs.** We evaluate our three model variants (Ours-S, Ours-B, and Ours-L) across a continuous spectrum of GPP constraints (see Fig. 4). As demonstrated by the objective metrics, CompactSplat exhibits a highly resilient Pareto frontier. Modulating the threshold to reduce the GPP within a reasonable operational range incurs only a marginal degradation in standard fidelity metrics. In exchange for this negligible drop in quantitative performance, our method yields substantial gains in representational compactness. This asymmetric trade-off confirms that the model successfully eliminates spatial redundancy, offering users a highly flexible knob to drastically reduce memory footprint and rendering overhead while rigorously preserving overall scene integrity.

**Qualitative Observations.** This robust quantitative performance is further corroborated by visual inspections of the rendered RGB, depth, and normal maps across varying GPP configurations (see Fig. 5). We observe distinct and highly complementary geometric behaviors at opposite ends of the sparsity spectrum. Operating at a high GPP allows the network to allocate dense, high-resolution primitives along intricate structures, yielding exceptionally sharp edges and crisp high-frequency boundaries. Conversely, operating at a lower GPP forces the architecture to leverage larger, scale-modulated primitives sampled from deeper pyramid levels. Interestingly, this highly pruned setting is intrinsically suited for modeling broad, homogeneous planar surfaces (e.g., walls and floors). By seamlessly tiling these expansive areas with single, expanded Gaussians, the low-GPP configuration fundamentally eliminates the spatial voids and "hollow" artifacts common in traditional sparse-view 3DGS, resulting in strictly watertight and visually coherent structural reconstructions.

## 5 Conclusion

In this paper, we presented **CompactSplat**, a novel feed-forward framework designed to overcome the severe spatial redundancy inherent in current sparse-view 3D Gaussian Splatting models. By synergizing a Texture-Aware Partitioner (TAP) with a Hierarchical Pyramid Sampler (HPS), our approach decisively breaks away from the rigid pixel-wise prediction paradigm, adaptively allocating primitives based on regional complexity. This scale-aware representation not only achieves state-of-the-art rendering fidelity across multiple datasets but also inherently supports dynamic sparsity control during inference. Consequently, CompactSplat delivers highly compact 3D priors, offering flexible, on-the-fly trade-offs between reconstruction quality and memory footprint.

## Acknowledgement

This work was partly supported by the NSFC (62431015, 62571317, 62501387), the Fundamental Research Funds for the Central Universities, Shanghai Key Laboratory of Digital Media Processing and Transmission under Grant 22DZ2229005, 111 project BP0719010.

## References

Honggyu An, Jaewoo Jung, Mungyeom Kim, Chaehyun Kim, Minkyeong Jeon, Jisang Han, Kazumi Fukuda, Takuya Narihira, Hyuna Ko, Junsu Kim, et al. C3g: Learning compact 3d representations with 2k gaussians. *arXiv preprint arXiv:2512.04021*, 2025.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024.

Yutong Chen, Marko Mihajlovic, Xiyi Chen, Yiming Wang, Sergey Prokudin, and Siyu Tang. Splatformer: Point transformer for robust 3d gaussian splatting. *arXiv preprint arXiv:2411.06390*, 2024a.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, pages 422–438. Springer, 2024b.

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 *European Conference on Computer Vision*, pages 370–386. Springer, 2024c.

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

Robert T Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE computer society conference on computer vision and pattern recognition*, pages 358–363. Ieee, 1996.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

Stavros Diolatzis, Tobias Zirr, Alexander Kuznetsov, Georgios Kopanas, and Anton Kaplanyan. N-dimensional gaussians for fitting of high dimensional functions. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

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

Xianze Fang, Jingnan Gao, Zhe Wang, Zhuo Chen, Xingyu Ren, Jiangjing Lyu, Qiaomu Ren, Zhonglei Yang, Xiaokang Yang, Yichao Yan, and Chengfei Lyu. Dens3r: A foundation model for 3d geometry prediction. *arXiv preprint arXiv:2507.16290*, 2025.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

Jingnan Gao, Zhe Wang, Xianze Fang, Xingyu Ren, Zhuo Chen, Shengqi Liu, Yuhao Cheng, Jiangjing Lyu, Xiaokang Yang, and Yichao Yan. More: 3d visual geometry reconstruction meets mixture-of-experts. *arXiv preprint arXiv:2510.27234*, 2025.

Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In *European Conference on Computer Vision*, pages 54–71. Springer, 2024.

Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2495–2504, 2020.

Bing He, Yunuo Chen, Guo Lu, Qi Wang, Qunshan Gu, Rong Xie, Li Song, and Wenjun Zhang. S4d: Streaming 4d real-world reconstruction with gaussians and 3d control points. *arXiv preprint arXiv:2408.13036*, 2024.

Bing He, Jingnan Gao, Yunuo Chen, Ning Cao, Gang Chen, Zhengxue Cheng, Li Song, and Wenjun Zhang. Surfsplat: Conquering feedforward 2d gaussian splatting with surface continuity priors. *arXiv preprint arXiv:2602.02000*, 2026.

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22012–22022, 2025.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024.

Zhihao Liang, Qi Zhang, Wenbo Hu, Lei Zhu, Ying Feng, and Kui Jia. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. In *European conference on computer vision*, pages 281–297. Springer, 2024.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14458–14467, 2021a.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

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. *Communications of the ACM*, 65(1): 99–106, 2021.

Arthur Moreau, Richard Shaw, Michal Nazarczuk, Jisu Shin, Thomas Tanay, Zhensong Zhang, Songcen Xu, and Eduardo Pérez-Pellitero. Off the grid: Detection of primitives for feed-forward 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11756–11766, 2026.

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

KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compgs: Smaller and faster gaussian splatting with vector quantization. In *European Conference on Computer Vision*, pages 330–349. Springer, 2024.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10349–10358, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering. *ACM Transactions on Graphics (TOG)*, 43(4):1–17, 2024.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021.

Jiawei Ren, Michal Jan Tyszkiewicz, Jiahui Huang, and Zan Gojcic. Tokengs: Decoupling 3d gaussian prediction from pixels with learnable tokens. *arXiv preprint arXiv:2604.15239*, 2026.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Xiaowei Song, Jv Zheng, Shiran Yuan, Huan-ang Gao, Jingwei Zhao, Xiang He, Weihao Gu, and Hao Zhao. Sa-gs: Scale-adaptive gaussian splatting for training-free anti-aliasing. *arXiv preprint arXiv:2403.19615*, 2024.

Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10208–10217, 2024.

Shengji Tang, Weicai Ye, Peng Ye, Weihao Lin, Yang Zhou, Tao Chen, and Wanli Ouyang. Hisplat: Hierarchical 3d gaussian splatting for generalizable sparse-view reconstruction. *arXiv preprint arXiv:2410.06245*, 2024.

Zhenggang Tang, Yuchen Fan, Dilin Wang, Hongyu Xu, Rakesh Ranjan, Alexander Schwing, and Zhicheng Yan. Mv-dust3r+: Single-stage scene reconstruction from sparse views in 2 seconds. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5283–5293. IEEE, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Diwen Wan, Ruijie Lu, and Gang Zeng. Superpoint gaussian splatting for real-time high-fidelity dynamic scene reconstruction. *arXiv preprint arXiv:2406.03697*, 2024.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5294–5306. IEEE, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024a.

Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Feng Chen, Jia-Wang Bian, Zheng Zhu, et al. Volsplat: Rethinking feed-forward 3d gaussian splatting with voxel-aligned prediction. In *European Conference on Computer Vision*, pages 1–19. Springer, 2026.

Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. Freesplat: Generalizable 3d gaussian splatting towards free view synthesis of indoor scenes. *Advances in Neural Information Processing Systems*, 37: 107326–107349, 2024b.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023.

Dejia Xu, Ye Yuan, Morteza Mardani, Sifei Liu, Jiaming Song, Zhangyang Wang, and Arash Vahdat. Agg: Amortized generative 3d gaussians for single image to 3d. *arXiv preprint arXiv:2401.04099*, 2024a.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 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. *arXiv preprint arXiv:2410.13862*, 2024b.

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

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 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. *arXiv preprint arXiv:2410.24207*, 2024.

Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. Yonosplat: You only need one model for feedforward 3d gaussian splatting, 2025.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024.

Chuanrui Zhang, Yingshuang Zou, Zhuoling Li, Minmin Yi, and Haoqian Wang. Transplat: Generalizable 3d gaussian splatting from sparse multi-view images with transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9869–9877, 2025.

Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21424–21433, 2024.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10324–10335, 2024.

# A Technical Appendices and Supplementary Material

## A.1 Preliminaries

Generalizable 3D Gaussian Splatting (3DGS) frameworks bypass the computationally heavy per-scene optimization by directly inferring a 3D scene representation from a sparse set of observations. Operating in a single forward pass, these networks typically adopt a dense, pixel-aligned prediction paradigm. Specifically, given $V$ source views $\{I^v\}_{v=1}^V$ along with their corresponding camera intrinsics $\{\mathbf{k}^v\}_{v=1}^V$ and extrinsics $\{\mathbf{T}^v\}_{v=1}^V$, a standard feedforward model $f_\theta$ outputs a set of Gaussian primitives for every pixel coordinate:

$$f_\theta : \{(I^v, \mathbf{k}^v, \mathbf{T}^v)\}_{v=1}^V \mapsto \left\{ \bigcup_{j=1}^{H\times W} \left(\boldsymbol{\mu}_j^v, \boldsymbol{\alpha}_j^v, \mathbf{r}_j^v, \mathbf{s}_j^v, \mathbf{c}_j^v\right) \right\}_{v=1}^V, \tag{5}$$

where $H$ and $W$ represent the spatial dimensions of the input images. The tuple defines the explicit attributes of the $j$-th Gaussian originating from view $v$: its 3D center coordinate $\boldsymbol{\mu}_j^v$, opacity $\boldsymbol{\alpha}_j^v$, rotation quaternion $\mathbf{r}_j^v$, scaling factor $\mathbf{s}_j^v$, and spherical harmonics $\mathbf{c}_j^v$.

The feasibility of this dense prediction relies heavily on the robust geometric priors extracted by modern vision backbones (e.g., ViTs Ranftl et al. (2021); Zhang et al. (2022); Wang et al. (2024a); Fang et al. (2025); Gao et al. (2025); Wang et al. (2025)). By unprojecting these rich 2D features into 3D space using camera parameters, the network establishes a rigid pixel-to-Gaussian mapping that can be supervised end-to-end via differentiable rasterization.

However, this rigid $H \times W$ formulation indiscriminately populates the 3D space with primitives regardless of the underlying scene complexity. By forcing a strict 1:1 mapping between pixels and Gaussians, existing architectures inevitably generate severe spatial redundancy in homogeneous regions and fail to exploit the geometric priors of piecewise planar surfaces. It is precisely this fundamental bottleneck in representational efficiency that CompactSplat seeks to overcome.

## A.2 Encoder Architecture

We adopt a dual-branch encoder design to extract both monocular and multi-view features for robust 3D reasoning, following the architecture proposed by DepthSplat Xu et al. (2024b) and subsequent SurfSPlat He et al. (2026).

**Multi-view Branch.** The multi-view encoder begins with a lightweight ResNet-style backbone composed of stride-2 convolutional layers, yielding spatially downsampled feature maps by a factor of $s$. To enable view aggregation, we employ a multi-view Swin Transformer Liu et al. (2021b) consisting of 6 stacked self- and cross-attention layers. This module outputs multi-view-aware features $\left\{\boldsymbol{F}^i\right\}_{i=1}^N$, where $\boldsymbol{F}^i \in \mathbb{R}^{\frac{H}{s}\times\frac{W}{s}\times C}$.

We further adopt the plane-sweep stereo technique Collins (1996); Xu et al. (2023) to construct geometric consistency. We uniformly sample $D$ candidate depths between near and far bounds. Given reference view $i$ and source view $j$, we warp features $\boldsymbol{F}_j$ to view $i$ at each depth $d_m$, resulting in $\left\{\boldsymbol{F}_{d_m}^{j\to i}\right\}_{m=1}^D$. These warped volumes are compared to $\boldsymbol{F}_i$ via dot-product similarity to construct a cost volume $\boldsymbol{C}^i \in \mathbb{R}^{\frac{H}{s}\times\frac{W}{s}\times D}$.

**Single-view Branch.** We utilize the ViT backbone from Depth Anything V2 model Yang et al. (2024) to extract monocular features. The output has a spatial resolution of $1/14$ relative to the original image and is bilinearly upsampled to match the cost volume resolution, yielding monocular features $\boldsymbol{F}_{\mathrm{m}}^i \in \mathbb{R}^{\frac{H}{s}\times\frac{W}{s}\times C_{\mathrm{m}}}$.

**U-Net and Depth Prediction.** The monocular and multi-view features $\boldsymbol{F}_{\mathrm{m}}^i$ and $\boldsymbol{C}^i$ are concatenated along the channel dimension and processed by a 2D U-Net to produce depth candidates $\boldsymbol{D}^i \in \mathbb{R}^{\frac{H}{s}\times\frac{W}{s}\times D}$. A softmax operation is applied over the depth axis, followed by a weighted summation to generate the predicted depth map.

To enhance depth quality, we employ a hierarchical cascade structure Gu et al. (2020), refining the predicted depth to $\boldsymbol{D}_{ds}^{i} \in \mathbb{R}^{\frac{2H}{s} \times \frac{2W}{s}}$, which is subsequently upsampled to full resolution using a DPT head Ranftl et al. (2021).

**Attribute Prediction.** The predicted depth is used to reconstruct Gaussian positions. For estimating the remaining Gaussian attributes—such as scale multipliers, high-frequency SH coefficients, and opacity—we apply an additional DPT head, conditioned on a concatenation of the input image, predicted depth, and encoder features.

**Hyperparameter Selection.** The downsample scale $s$ is set to 4. Channel number $C$ is set to 128, channel number $D$ is set to 128. The channel number $C_{\mathrm{m}}$ of the monocular feature is set to 64 for small model, 96 for base model, 128 for large model.

**Gaussian Processor.** The Surface Continuity Prior(SCP) and the Forced Alpha Blending methods from Surfsplat have been adopted to process the gaussian attributes.

**Note:** Our implementation is consistent with DepthSplat Xu et al. (2024b) and SurfSplat He et al. (2026) for reproducibility. No architectural modifications are made to the encoder unless otherwise stated.

### A.3 Implementation Details.

CompactSplat is implemented in PyTorch Paszke et al. (2019) and optimized using the AdamW optimizer Loshchilov and Hutter (2017) coupled with a cosine annealing learning rate schedule. For the core components of our proposed architecture, we set the maximum block size in the Texture-Aware Partitioner to $S_{\text{max}} = 16$. During the sparsity-aware fine-tuning stage of our curriculum, the dynamic partition threshold is sampled uniformly up to $\tau_{\text{max}} = 0.02$.

Following standard practices He et al. (2026); Xu et al. (2024b), we instantiate our model using varying capacities of the Depth Anything V2 Yang et al. (2024) monocular backbone (ViT-S, ViT-B, and ViT-L), denoting these variants as Ours-S, Ours-B, and Ours-L, respectively. All models are trained on NVIDIA A100 GPUs. Specifically, Ours-S is trained for 300K iterations with a batch size of 16, whereas Ours-B and Ours-L are trained for 600K iterations with a batch size of 8. The learning rate is configured at $2 \times 10^{-6}$ for the pretrained monocular backbone and $2 \times 10^{-4}$ for all newly initialized layers.

### A.4 Cross-Dataset Generalization

As detailed in Table 3, our extended cross-dataset evaluations underscore the robust generalizability of CompactSplat. Even when evaluated outside its primary training distribution, CompactSplat achieves rendering fidelity comparable to the state-of-the-art SurfSplat He et al. (2026) baseline, while consistently yielding a significantly more compact and memory-efficient 3D representation.

Table 3: Cross datasets performance.

| | Scannet | | | | DL3DV | | | | DTU | | | | Average | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | GPP↓ | PSNR↑ | SSIM↑ | LPIPS↓ | GPP↓ | PSNR↑ | SSIM↑ | LPIPS↓ | GPP↓ | PSNR↑ | SSIM↑ | LPIPS↓ | GPP↓ |
| pixelSplat 2024 | 19.606 | 0.714 | 0.324 | 3.000 | 27.201 | 0.882 | 0.104 | 3.000 | 12.752 | 0.329 | 0.639 | 3.000 | 19.853 | 0.642 | 0.356 | 3.000 |
| HiSplat 2024 | 19.095 | 0.691 | 0.342 | 1.750 | 26.242 | 0.869 | 0.112 | 1.750 | 16.019 | 0.671 | 0.277 | 1.750 | 20.452 | 0.744 | 0.244 | 1.750 |
| MVSplat 2024c | 18.725 | 0.692 | 0.333 | 1.000 | 23.841 | 0.768 | 0.156 | 1.000 | 13.914 | 0.470 | 0.386 | 1.000 | 18.827 | 0.643 | 0.292 | 1.000 |
| TranSplat 2025 | 18.944 | 0.705 | 0.332 | 1.000 | 23.913 | 0.771 | 0.161 | 1.000 | 14.956 | 0.527 | 0.327 | 1.000 | 19.271 | 0.668 | 0.273 | 1.000 |
| DepthSplat 2024b | 20.201 | 0.735 | 0.305 | 1.000 | 28.141 | 0.905 | 0.083 | 1.000 | 14.592 | 0.425 | 0.436 | 1.000 | 20.978 | 0.688 | 0.275 | 1.000 |
| SurfSplat 2026 | 20.305 | 0.731 | 0.313 | 1.000 | 27.384 | 0.890 | 0.106 | 1.000 | 15.544 | 0.488 | 0.329 | 1.000 | 21.078 | 0.703 | 0.249 | 1.000 |
| YonoSplat* 2025 | 20.323 | 0.727 | 0.311 | 1.000 | 25.636 | 0.843 | 0.124 | 1.000 | 11.721 | 0.343 | 0.596 | 1.000 | 19.227 | 0.638 | 0.261 | 1.000 |
| Ours | 19.905 | 0.706 | 0.353 | 0.643 | 25.573 | 0.868 | 0.152 | 0.715 | 15.368 | 0.476 | 0.348 | 0.698 | 20.208 | 0.683 | 0.284 | 0.685 |

Table 4: Comparison with non-pixel-aligned methods on RE10K.

| Method | PSNR↑ | SSIM↑ | LPIPS↓ | GS Num.↓ |
|---|---|---|---|---|
| TokenGS Ren et al. (2026) | 28.02 | 0.896 | 0.147 | 66k |
| C3G An et al. (2025) | 22.39 | 0.713 | 0.259 | 2k |
| VolSplat Wang et al. (2026) | 22.86 | 0.811 | 0.201 | 131k |
| CompactSplat-L | 27.57 | 0.892 | 0.117 | 72.6k |

### A.5 Comparison with other non-pixel-aligned methods

TokenGS (1024 tokens) and C3G use published results; VolSplat uses our two-view re-evaluation. TokenGS and CompactSplat use two $256 \times 256$ inputs. C3G reports $224 \times 224$ training inputs and uses test-time target-camera alignment. Training and camera handling differ across methods.

### A.6 Extended Qualitative Results

Figure 6 presents an additional set of reconstructed scenes across diverse environments. These qualitative visualizations further corroborate that CompactSplat reliably maintains photorealistic rendering quality and watertight geometric structures, all while empowering users with dynamic, on-the-fly control over representational compactness.

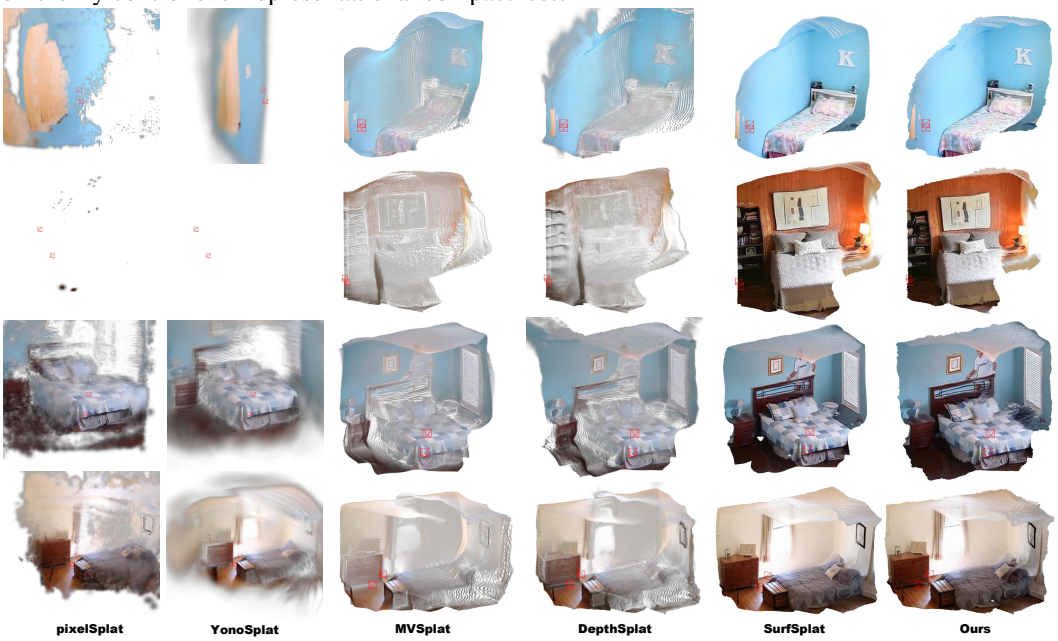


Figure 6: **More subjective comparison of generated 3D scenes.**

### A.7 Limitations and Future Directions.

While CompactSplat significantly improves representational efficiency, it leaves exciting room for future exploration. First, although our quadtree-based partitioning is highly efficient, it relies on rigid orthogonal grids. Exploring more sophisticated, shape-aware, or semantic-driven subdivision strategies could further push the boundaries of the Pareto frontier, extracting even higher visual fidelity at extreme pruning ratios. Second, our current formulation primarily eliminates spatial redundancy within individual views, leaving the inherent *inter-view* redundancy largely unexplored. Since multiple source cameras inevitably observe overlapping geometries, future research could investigate cross-view primitive pruning or multi-view fusion mechanisms to further condense the global 3D scene representation.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Sec. 1

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 5

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Sec. 2

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 4

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Once the paper is accepted, we will release the code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 4

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Sec. 3

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. 4

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: No societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing assets used in the paper have been properly credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.