

Compact Feed-Forward 3D Gaussians via Saliency-Guided Primitive Merging

Tim-Felix Faasch¹

tim-felix.faasch@de.bosch.com

Jochen Kall¹

jochen.kall@de.bosch.com

Cyrrill Stachniss^{2, 3}

cyrill.stachniss@igg.uni-bonn.de

¹ Bosch Research

Hildesheim, Germany

² University of Bonn

Bonn, Germany

³ Lamarr Institute for Machine Learning

and Artificial Intelligence

Bonn, Germany

Abstract

3D scene reconstruction, modeling, and rendering are highly relevant for numerous tasks, and 3D Gaussian splatting has become a standard choice in this context. Its feed-forward variants provide fast reconstruction from sparse input views but often produce per-pixel primitives, leading to highly redundant and thus inefficient representations. We present a structure-aware merging pipeline that takes per-pixel primitives from any feed-forward method and consolidates them into a compact, content-adaptive Gaussian set while largely retaining visual quality at just $1/20^{\text{th}}$ of the Gaussians of a per-pixel method. We group spatially coherent Gaussians of similar appearance into variable-size clusters via adaptive superpixel segmentation guided by a saliency map, which allocates fine segments to textured regions and coarse segments to homogeneous areas. We compress each cluster into a compact latent representation through a learned encoder, then match and consolidate representations across views based on geometric overlap and feature similarity via a learned merger. A level-of-detail decoder then produces the final Gaussians at a controllable resolution, enabling a flexible quality-efficiency trade-off at inference. As a post-processing module, the pipeline is backbone-agnostic, leveraging the strengths of existing feed-forward methods. This leads to better and more robust quality than achieved by previous approaches that target a reduction in primitive count, while providing a highly compact representation, that can be rendered efficiently.

1 Introduction

3D Gaussian splatting (3DGS) [17] has emerged as a powerful representation for reconstructing 3D scenes with photorealistic quality, leading to its widespread adoption in simulation [16, 46], 3D generation [57, 69], and robotics [68, 60]. Traditional per-scene optimization requires dense scene observations and consumes substantial compute, limiting its applicability to large-scale scenarios. Feed-forward (FF) reconstruction methods address this by predicting Gaussian primitives directly from input views [2, 11, 15, 67, 48, 49, 61, 62],

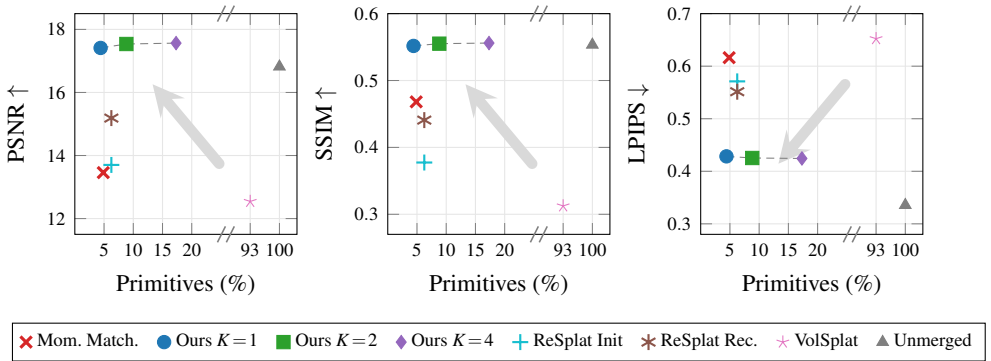


Figure 1: NVS quality vs. relative primitive count (DA3 backbone, averaged over DL3DV-Bench, MipNeRF360, and Tanks & Temples). Arrows indicate favorable direction.

but they typically produce per-pixel or per-voxel Gaussians, resulting in highly redundant representations [49, 43].

Since rendering cost in 3DGS scales with primitive count — each Gaussian must be sorted, projected, and alpha-composited — redundant primitives inflate inference latency and memory footprint without contributing additional visual detail. This inefficiency is particularly problematic for applications that scale to many scenes and complex downstream tasks, such as simulation-driven training of models for robotics or autonomous driving, where fast rendering is crucial for large-scale data generation. Redundancy can also hinder generalization to novel views, as overfitting to input views may lead to high-frequency artifacts. To address these challenges, we focus on reducing the number of primitives in FF-reconstructed scenes while maintaining visual quality. A good merging strategy should be content-adaptive, allocating representational capacity where it matters most, and it should be able to consolidate redundant primitives across overlapping views.

Existing approaches have made significant progress in addressing the problem of redundant Gaussians through spatial discretization [45, 42, 43], iterative refinement [47], or through detection-based primitive placement [49]. While these methods achieve notable reductions in primitive count with good rendering quality, opportunities for further improvement remain: uniform discretization strategies do not fully exploit the varying visual importance across scene regions, iterative refinement methods can lead to overfitting on input views, and detection-based methods operate during initial reconstruction without support for merging primitives across multiple views or adapting to new reconstruction backbones. We aim to address these open challenges with our work.

The main contribution of this paper is a novel technique to substantially reduce the complexity of 3D Gaussian scenes that feed-forward methods produce. We propose a structure-aware, superpixel-based primitive merging strategy that takes per-pixel Gaussians from any feed-forward method and consolidates them into a highly compact representation. Content-adaptive superpixel segmentation, guided by local image structure, groups spatially coherent, perceptually similar per-pixel Gaussians. A learned set-attention encoder compresses each group into a compact latent representation, which we then match across views and merge using a learned fusion module. A local self-attention module refines the result, and a decoder produces output Gaussians at a controllable level of detail. This encode-merge-decode pipeline retains the bulk of the novel view synthesis quality at just $1/20^{\text{th}}$ of the Gaussians

of a per-pixel method, vastly increasing the rendering speed for downstream applications while providing more robust and reliable reconstructions than competing reduced-primitive methods, particularly in sparse-view settings. Crucially, the merging module attaches to any per-pixel Gaussian predictor without retraining the backbone, making it a flexible solution for improving the efficiency of existing FF 3DGS methods.

2 Related Work

3D Scene Representations: Traditional computer graphics has long relied on explicit representations such as meshes and point clouds for rasterization-based rendering pipelines [2]. Neural Radiance Fields (NeRF) [28] introduced a neural implicit approach that learns to represent scenes from multi-view imagery, enabling applications in reconstruction [6, 28], simulation [88], and content generation [61, 83]. Despite their quality, NeRFs suffer from high computational demands due to volumetric rendering. 3D Gaussian splatting (3DGS) [10] is an explicit representation that addresses this limitation by representing scenes as collections of 3D Gaussians rendered through differentiable rasterization, achieving both faster optimization and interactive rendering frame rates. However, 3DGS still requires per-scene optimization, which incurs significant computational cost when scaling to a large number of scenes and fails to converge to a meaningful geometry for sparse-view settings.

Feed-Forward Reconstruction: To overcome the limitations of per-scene optimization, feed-forward (FF) methods predict 3D Gaussians directly from input views [2, 11, 15, 37, 48, 49, 51, 52]. The backbone is typically a large vision transformer (ViT) [12] that encodes the input views and regresses the Gaussian parameters. FF reconstruction runs significantly faster than per-scene optimization and can handle sparse-view settings, but often places Gaussians less efficiently. Most existing FF methods predict per-pixel Gaussians [2, 11, 23, 37, 48, 49, 51, 52], which results in a highly redundant representation with an excessive number of sub-optimally placed primitives.

Efficient 3D Gaussian Splatting: The number of primitives in a 3DGS reconstruction directly affects rendering speed, memory consumption, and storage cost. Several strategies aim to reduce this overhead. Compression techniques such as quantization and learned entropy coding lower the per-primitive memory footprint without reducing the number of Gaussians [8, 9, 10, 20, 30, 40]. In contrast, compaction methods aim to reduce the primitive count itself: hierarchical representations merge primitives that contribute primarily to the background [18], and modified densification strategies prune redundant Gaussians during optimization [6, 13, 20, 21, 26]. Compression and compaction are complementary: attribute-compression schemes such as HAC/HAC++ [8, 10] can be applied on compact Gaussian scenes for further storage savings. However, both compression and compaction methods of this kind are tightly coupled to the per-scene optimization process, and are therefore not directly compatible with feed-forward reconstruction methods.

Some methods specifically tackle the efficiency of FF-produced representations. The most common approach is voxelization, where some methods directly predict voxel-aligned Gaussians [27, 32, 42] and others apply post-processing voxelization [15, 43]; yet uniform spatial discretization does not account for varying visual complexity across scene regions. Detection-based approaches [49] move from per-pixel regression to importance-driven Gaussian placement, allocating primitives according to visual saliency. Iterative refinement methods [47] predict Gaussians in a subsampled space and refine them using gradient-free feedback from the rendering error, yielding $16\times$ fewer primitives than per-pixel methods. Both

directions integrate Gaussian reduction directly into the reconstruction model itself, which is fundamentally different and complementary to our post-hoc compression approach. While these methods potentially allow for more efficient FF reconstruction, they are less flexible than our post-processing pipeline, and cannot directly leverage the strong performance of existing FF backbones. Graph-based fusion mechanisms [44, 54] address the redundancy of Gaussians in overlapping views by progressively merging per-pixel Gaussians across views via depth-proximity matching, GRU-based feature updates, or graph convolutions with overlap-weighted edges followed by pooling layers that prune geometrically close primitives. These methods rely on globally uniform proximity thresholds and do not account for visual complexity, allocating primitives uniformly regardless of scene content. In contrast, our method focuses on redistributing primitive capacity based on visual complexity, in addition to addressing the redundancy of overlapping views, which allows for much greater compression of the Gaussian representation while retaining high reconstruction quality. A complementary direction compresses redundant multi-view inputs into a compact latent state before Gaussian prediction using an Information Bottleneck formulation, enabling FF models to scale to over a hundred input views [40]. This acts as an input-side compression module and does not address the redundancy of the predicted per-pixel Gaussians; it could therefore be combined with our output-side primitive merging to obtain an end-to-end efficient pipeline.

Supersixel Segmentation: Supersixel segmentation is a widely used technique in computer vision that groups pixels into small, perceptually meaningful regions based on color and spatial proximity [0, 45, 55]. Practitioners often apply it as a pre-processing step for tasks such as object recognition, image segmentation, and scene understanding to reduce computational complexity and improve efficiency of subsequent processing steps [9]. Recent supersixel methods with adaptive sizing capabilities [59, 56] are particularly effective, as they can allocate more segments to visually complex regions. However, as far as we are aware, no prior work has applied these methods to 3D scene reconstruction.

3 Preliminaries

3D Gaussian Splatting. In 3DGS [17], a scene is represented as a set of N anisotropic Gaussians $\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_N\}$. Each Gaussian $\mathcal{G}_i$ is parameterized by a mean $\mu_i \in \mathbb{R}^3$ and a covariance $\Sigma_i \in \mathbb{R}^{3 \times 3}$, with its evaluation at position $\mathbf{x} \in \mathbb{R}^3$ given by

$$\mathcal{G}_i(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^\top \Sigma_i^{-1}(\mathbf{x} - \mu_i)\right). \quad (1)$$

For rendering, Gaussians are projected into image space via EWA splatting [60], yielding a 2D covariance $\Sigma'_i = \mathbf{J}\mathbf{W}\Sigma_i\mathbf{W}^\top\mathbf{J}^\top$ with the viewing transformation $\mathbf{W}$ and the projection Jacobian $\mathbf{J}$. The 3D covariance is decomposed as $\Sigma_i = \mathbf{R}_i\mathbf{S}_i\mathbf{S}_i^\top\mathbf{R}_i^\top$ into rotation $\mathbf{R}_i$ and diagonal scale $\mathbf{S}_i$, and is parameterized by a quaternion $\mathbf{q}_i \in \mathbb{R}^4$ and scale vector $\mathbf{s}_i \in \mathbb{R}^3$. The view-dependent color $\mathbf{c}_i$ of $\mathcal{G}_i$ is encoded via spherical harmonic (SH) coefficients $\hat{\mathbf{c}}_{0,i}, \dots, \hat{\mathbf{c}}_{M,i}$, where $\mathbf{c}_i = \sum_{k=0}^M \hat{\mathbf{c}}_{k,i} H_k(\mathbf{d})$ for view direction $\mathbf{d} \in \mathbb{R}^3$, with the SH basis functions H_k . The final pixel color is obtained by front-to-back alpha blending:

$$\mathbf{c}_{\text{pixel}} = \sum_{i=1}^N T_i \alpha_i \mathbf{c}_i, \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (2)$$

where T_i is the transmittance, and α_i the opacity of $\mathcal{G}_i$. Each Gaussian is fully described by its set of parameters $\Theta_i = (\mu_i, \mathbf{q}_i, \mathbf{s}_i, \alpha_i, \hat{\mathbf{c}}_{0,\dots,M,i})$. In the standard per-scene optimization

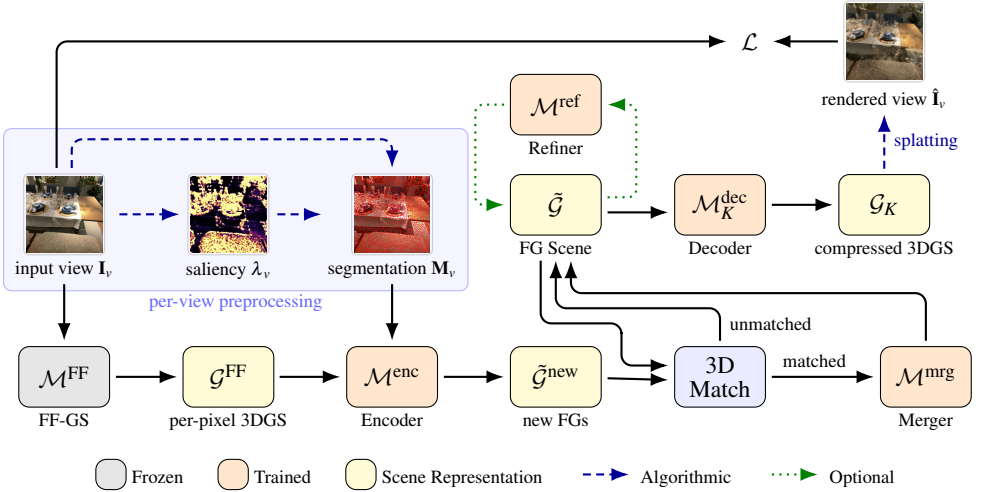


Figure 2: Pipeline overview. Per-pixel Gaussians $\mathcal{G}^{\text{FF}}$ are grouped into saliency-guided superpixels. The encoder $\mathcal{M}^{\text{enc}}$ compresses each group into a single Feature Gaussian (FG), the merger $\mathcal{M}^{\text{mrg}}$ fuses FGs that overlap across views, the refiner $\mathcal{M}^{\text{ref}}$ updates each FG based on its neighbors. The decoder $\mathcal{M}_K^{\text{dec}}$ expands each FG into K output Gaussians, resulting in the output scene $\mathcal{G}_K$.

setting, these parameters are optimized via gradient descent, minimizing a combination of different photometric losses between images rendered through differentiable rasterization and ground-truth images.

Feed-Forward Gaussian Splatting. Feed-forward (FF) methods [10, 15, 23, 28] bypass per-scene optimization by predicting Gaussian parameters directly from N input views $\{\mathbf{I}_v\}_{v=1}^N$. A typical architecture feeds each view through a vision transformer (ViT) [10] backbone to extract per-patch features, and applies multi-view cross-attention to obtain multi-view-aware feature maps. Prediction heads then regress point maps, which can be projected into 3D space to retrieve the Gaussian means μ_i using known camera parameters. The remaining parameters are similarly predicted from the patch features. FF methods have varying input requirements: *posed* methods require known camera intrinsics and extrinsics [10, 15], whereas *unposed* methods jointly predict camera parameters and scene geometry [23, 28].

4 Saliency-Guided Superpixel Merging

Given per-pixel Gaussian primitives predicted by an arbitrary frozen feed-forward backbone $\mathcal{M}^{\text{FF}}$, our method consolidates them into a compact representation that largely retains visual fidelity. A key design goal is content-adaptivity: homogeneous regions (e.g., sky, walls) can be aggressively compressed, whereas fine detail (e.g., at edges or textures) requires a higher representational capacity. We achieve this through an encode-merge-decode pipeline depicted in Fig. 2. First, saliency-guided superpixel segmentation groups spatially coherent, perceptually similar Gaussians at a content-adaptive granularity (Sec. 4.1). A learned encoder $\mathcal{M}^{\text{enc}}$ compresses each group into a latent-augmented representation we call a *Feature Gaussian* (Sec. 4.2). A level-of-detail decoder $\mathcal{M}_K^{\text{dec}}$ reconstructs 3D Gaussians, compati-

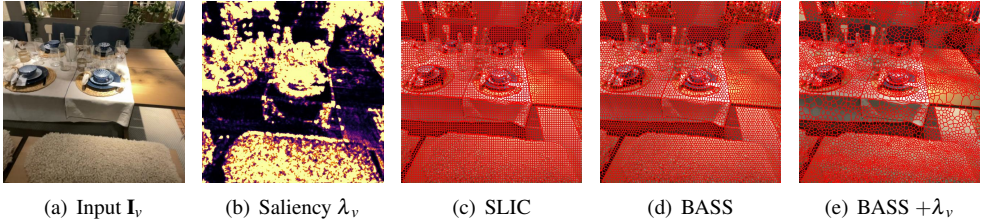


Figure 3: Superpixel comparison at similar segment count: SLIC (uniform), BASS with uniform seeds, and BASS with saliency-guided seeds λ_v .

ble with existing rendering pipelines, at a controllable resolution K (Sec. 4.3). Cross-view matching identifies overlapping Feature Gaussians, and a learned merger $\mathcal{M}^{\text{mrg}}$ consolidates them, enabling further compression (Sec. 4.4). A local self-attention refiner $\mathcal{M}^{\text{ref}}$ provides neighborhood awareness so that thin structures spanning multiple groups can be faithfully reconstructed (Sec. 4.5). The pipeline is trained end-to-end with photometric losses on the rendered output (Sec. 4.6).

4.1 Content-Adaptive Superpixel Grouping

The first step partitions each view’s per-pixel Gaussians into groups that the encoder can compress. A good group for merging should be color-homogeneous (so appearance can be preserved), spatially contiguous (necessary for 3D coherence), and have regular shape (easier to represent with few Gaussians). At the same time, group size should adapt to local scene complexity: edges and corners carry geometric detail that merging would destroy, so they require smaller groups and thus more output primitives; flat regions can be aggressively merged.

Bayesian Adaptive Superpixel Segmentation (BASS) [69] produces segments with the first three properties by iteratively maximizing a posterior that balances color homogeneity with spatial compactness, while maintaining regular segment boundaries. Standard BASS initializes seeds on a uniform hexagonal grid, and achieves content adaptivity by iteratively shifting superpixel centers and borders. For very small segment sizes, this can still lead to a large number of segments in flat regions, which is undesirable for our application. We replace the uniform initialization with a saliency-guided seed placement: we compute the Shi-Tomasi corner response λ_v [56] at each pixel (the minimum eigenvalue of the image structure tensor [64]) and sample seeds densely where this response is high and sparsely elsewhere. BASS then refines these seeds into final segments of similar color and a regular shape. The output is a per-view partition map $\mathbf{M}_v$ that assigns each pixel (and its associated Gaussian) to a superpixel group $\mathcal{S}_j$. Through the combination of saliency-guided seeding and BASS segmentation, we obtain small segments in regions of high detail and larger segments in flatter regions, effectively adapting to local scene complexity (Fig. 3).

4.2 Feature Gaussian Encoder

Given a superpixel group $\mathcal{S}_j$, the encoder $\mathcal{M}^{\text{enc}}$ compresses its Gaussians into an intermediate representation that downstream stages (matching, merging, refinement) can process efficiently. To achieve this, we introduce the *Feature Gaussian* $\tilde{\mathcal{G}}_j$, a latent-augmented representation that encodes the entire group with a single set of parameters. A Feature Gaussian

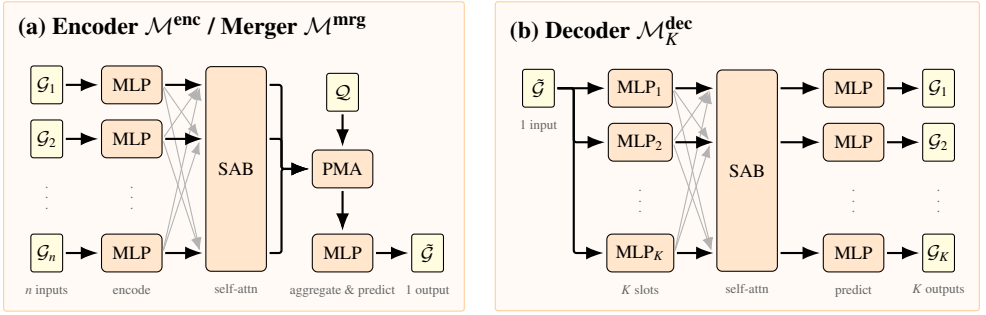


Figure 4: Internal architecture of our modules built on the Set Transformer [22] building blocks SAB and PMA. (a) The encoder and merger share the same many-to-one pattern: n input tokens are refined by SABs and aggregated into a single output via PMA and a learnable query $\mathcal{Q}$. (b) The decoder inverts this pattern: a single Feature Gaussian is expanded into K output Gaussians via learned slot tokens.

has the same spatial format as a regular Gaussian (position μ_j , scale $\mathbf{s}_j$, and rotation $\mathbf{q}_j$), but replaces view-dependent color with a base color $\mathbf{c}_j$ and a latent vector $\mathbf{z}_j \in \mathbb{R}^d$. The latent stores the visual appearance and geometric structure of the entire group so that the decoder can later expand it into one or more output Gaussians that capture the group’s complexity.

Compressing a variable-size set of Gaussians into a single Feature Gaussian requires an architecture that is permutation-invariant (the output should not depend on input order) and handles arbitrary group sizes. We use the Set Transformer [22]: an MLP projects per-Gaussian attributes into tokens, a stack of Set Attention Blocks (SABs) refines them via pairwise interactions, and a Pooling-by-Multihead-Attention (PMA) layer aggregates the set into a single output using a learnable query $\mathcal{Q}$ (Fig. 4). A final MLP produces the latent feature $\mathbf{z}_j$, and a position offset from the opacity-weighted centroid, allowing the encoder to compensate for outliers that would otherwise skew the placement of the Feature Gaussian. The base color $\mathbf{c}_j$ is the opacity-weighted average of the group members’ diffuse color. Notably, the encoder does not predict the scale $\mathbf{s}_j$ or rotation $\mathbf{q}_j$ that are needed for cross-view matching; these are produced by the decoder $\mathcal{M}_1^{\text{dec}}$. This design prevents a collapse mode where the encoder shrinks geometry to avoid cross-view matching entirely, which would simplify reconstruction but hurts compression.

4.3 Level-of-Detail Decoder

The decoder $\mathcal{M}_K^{\text{dec}}$ produces the K output Gaussians from each Feature Gaussian, which are compatible with any differentiable rasterizer and can be directly rendered. We train multiple decoder heads for different K : the $K=1$ head provides the geometry of the Feature Gaussian, which is used for cross-view matching and serves as maximum-compression output, while heads with $K>1$ trade a higher number of Gaussians for higher reconstruction fidelity. This exposes K as an inference-time knob, allowing the user to balance quality against rendering compute cost without retraining.

Expanding a single latent into K outputs requires learned specialization. We use a slot-based design (Fig. 4): a slot-seed MLP maps the latent feature $\mathbf{z}_j$ and base color $\mathbf{c}_j$ into K slot tokens, per-slot self-attention lets slots coordinate, and output heads predict each Gaussian’s parameters. Mean μ_i and base color $\mathbf{c}_i$ are predicted as offsets from the Feature Gaussian’s

parameters. At inference, we prune Gaussians with $\alpha_i < \tau_a$ for further compression.

4.4 Cross-View Matching and Merging

Feature Gaussians $\tilde{\mathcal{G}}$ from different views that observe the same 3D region should be merged to reduce redundancy. We identify candidate groups using spatial overlap and feature similarity, then fuse them with a learned merger $\mathcal{M}^{\text{mrg}}$.

For each Feature Gaussian $\tilde{\mathcal{G}}_j$, we retrieve the k nearest neighbors via 3D k NN on positions, excluding candidates from the same input view. A candidate pair passes the match gate if both (i) the cosine similarity between latent features exceeds τ_f , and (ii) the AABB intersection-over-union of the FGs’ geometry exceeds τ_g . This produces a set of matched edges over all Feature Gaussians.

Since one FG may match several others, and matching is not transitive, we take the connected components of this edge graph as the final merge groups. Feature Gaussians with no matches are appended to the final set $\tilde{\mathcal{G}}$ unchanged. $\mathcal{M}^{\text{mrg}}$ merges each connected group using the same SAB+PMA [22] backbone as the encoder (Fig. 4). Dedicated zero-initialized residual heads predict parameter updates to the 0th Feature Gaussian in the group: $\Delta\mu, \Delta\mathbf{z}, \Delta\mathbf{m}, \Delta\mathbf{c}_0, \Delta\mathbf{s}, \Delta\mathbf{q}$. This way the merger starts as an identity and training remains stable.

4.5 Refinement and Online Reconstruction

Refinement. The final rendering quality depends in part on how well neighboring Feature Gaussians $\tilde{\mathcal{G}}$ interact — particularly along thin structures or at depth boundaries. In these areas, individually compressed groups might not perfectly line up in the final reconstruction. The refiner $\mathcal{M}^{\text{ref}}$ addresses this by updating each Feature Gaussian $\tilde{\mathcal{G}}_j$ based on its k NN neighborhood through self-attention: neighbors’ relative positions and attributes serve as context tokens, and residual heads predict parameter updates $\Delta\mu, \Delta\mathbf{z}, \Delta\mathbf{m}, \Delta\mathbf{c}_0, \Delta\mathbf{s}, \Delta\mathbf{q}$. As with the merger, all residual projections are zero-initialized for stable training.

Online reconstruction. Our pipeline also supports online reconstruction, where views are processed as a sequence as they arrive rather than in a single batch. This allows for updates to the reconstructed scene as new observations become available and enables reconstruction of large scenes that do not fit into memory at once. In practice, we initialize the Feature Gaussian set $\tilde{\mathcal{G}}$ from a first batch of observations. When a new view arrives, the pipeline groups and encodes its per-pixel Gaussians into new Feature Gaussians, then matches and merges them into the existing set $\tilde{\mathcal{G}}$. The decoder produces output Gaussians on demand from the current state of $\tilde{\mathcal{G}}$ via $\mathcal{M}_K^{\text{dec}}$.

4.6 Training Objectives

During training, the feed-forward backbone $\mathcal{M}^{\text{FF}}$ remains frozen; we optimize only the encoder $\mathcal{M}^{\text{enc}}$, merger $\mathcal{M}^{\text{mrg}}$, refiner $\mathcal{M}^{\text{ref}}$, and decoders $\mathcal{M}_K^{\text{dec}}$. We train the pipeline end-to-end on multi-view image data with the following losses.

Photometric reconstruction. The primary supervision consists of Mean Squared Error (MSE), SSIM [45], and LPIPS [53] losses between rendered and ground-truth images, optionally computed on both input views and held-out novel views.

Teacher loss. Due to randomly initialized weights of our modules, the geometry of the compressed Gaussians is in turn essentially random during early training. This can cause single Gaussians to cover the renders from the training views entirely, inhibiting gradient flow.

To address this, we supervise the encoder with a closed-form moment-matching target [L8]: for each superpixel group $\mathcal{S}_j$, we fit a single Gaussian to the group’s opacity-weighted mean

$$\mu_j^{\text{teach}} = \frac{\sum_{i \in \mathcal{S}_j} \alpha_i \mu_i}{\sum_{i \in \mathcal{S}_j} \alpha_i} \quad (3)$$

and opacity-weighted covariance

$$\Sigma_j^{\text{teach}} = \frac{\sum_{i \in \mathcal{S}_j} \alpha_i (\mu_i - \mu_j^{\text{teach}})(\mu_i - \mu_j^{\text{teach}})^\top}{\sum_{i \in \mathcal{S}_j} \alpha_i}. \quad (4)$$

We compare the resulting Gaussian parameters to the output of our merging pipeline using a weighted sum of a geodesic quaternion loss for $\mathbf{q}_i$ and L1 losses for the remaining parameters. This teacher signal decays on a schedule, allowing the learned encoder to eventually surpass the moment-matched initialization.

Decoder diversification. For $K > 1$ decoders, a regularizer encourages the K output Gaussians to be spatially diverse without collapsing into identical copies. We define a target AABB-IoU and apply an attraction/repulsion term around it, and additionally penalize large opacity differences within a group to prevent single-primitive dominance. Decaying this loss during training lets the photometric objective override it once diversity is established, allowing the model to collapse to fewer Gaussians where a single primitive suffices.

The complete training objective combines all terms:

$$\mathcal{L} = \lambda_{\text{MSE}} \mathcal{L}_{\text{MSE}} + \lambda_{\text{SSIM}} \mathcal{L}_{\text{SSIM}} + \lambda_{\text{LPIPS}} \mathcal{L}_{\text{LPIPS}} + \lambda_{\text{teach}} \mathcal{L}_{\text{teach}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}}, \quad (5)$$

where λ_{teach} and λ_{div} follow a decay schedule and concrete weight values are provided in the supplementary material.

5 Experimental Evaluation

We evaluate our saliency-guided superpixel-based primitive merging strategy on novel view synthesis quality and number of primitives.

5.1 Experimental Setup

Implementation Details. We evaluate our approach with three complementary FF backbones: DepthSplat [L8], a posed method that leverages multi-view cost volumes augmented with monocular depth features; AnySplat [L9], an unposed method that jointly estimates geometry from unconstrained views; and Depth Anything 3 (DA3) [L3], a foundation model that unifies depth estimation, pose estimation, and per-pixel 3D Gaussian prediction in a single ViT backbone, supporting both posed and unposed operation. We train all added modules with the AdamW optimizer [L5] for 300 000 steps on the DL3DV-10K dataset [L4], using a cosine schedule with a maximum learning rate of 5×10^{-5} . For the level-of-detail decoder, we maintain three heads with $K=1$, $K=2$, and $K=4$ output Gaussians per Feature Gaussian, which are jointly trained. We keep the backbones frozen during training.

Evaluation Details. We test the reconstruction quality on DL3DV-Bench [L2], which contains 140 scenes not used during training. We additionally evaluate on MipNeRF360 [L6] and Tanks and Temples [L9], two widely used benchmarks that were not included in the

Table 1: Novel view synthesis quality for all backbone variants and baselines. Best in **bold**, second best underlined.

Backbone	Method	r_c (%)↓	DL3DV-Bench			MipNeRF360			T&T		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
AnySplat	Unmerged	100.0%	13.73	0.361	0.495	12.17	0.345	0.497	11.89	0.416	0.466
	Voxelized	83.2%	13.76	0.360	0.497	12.17	0.344	0.498	11.89	0.413	0.468
	Mom. Match.	5.3%	14.45	0.410	0.566	13.46	0.398	0.559	12.49	0.463	0.517
	Ours ($K=1$)	<u>4.4%</u>	14.70	0.408	0.550	14.48	0.405	0.544	13.24	0.458	0.504
	Ours ($K=2$)	8.9%	14.81	0.411	0.549	14.74	0.406	0.544	13.81	0.473	0.505
	Ours ($K=4$)	17.8%	14.78	0.409	0.547	14.81	0.408	0.542	13.93	0.474	0.503
DepthSplat	Unmerged	100.0%	15.78	0.577	<u>0.347</u>	13.96	0.480	<u>0.389</u>	13.72	0.549	<u>0.377</u>
	Mom. Match.	6.1%	13.16	0.466	0.573	12.35	0.424	0.569	11.43	0.498	0.550
	Ours ($K=1$)	5.3%	17.17	0.529	0.440	15.37	0.457	0.452	14.33	0.508	0.461
	Ours ($K=2$)	10.5%	17.41	0.536	0.436	16.05	0.469	0.448	14.60	0.517	0.458
	Ours ($K=4$)	21.1%	17.50	0.539	0.433	16.31	0.474	0.446	14.78	0.522	0.457
DA3	Unmerged	100.0%	18.68	0.620	0.297	16.71	0.495	0.356	15.07	0.546	0.353
	Mom. Match.	4.9%	13.79	0.459	0.637	14.04	0.440	0.601	12.55	0.505	0.611
	Ours ($K=1$)	4.4%	18.37	0.578	0.422	17.87	0.514	0.442	16.00	0.562	0.421
	Ours ($K=2$)	8.8%	18.50	0.582	0.417	<u>18.02</u>	<u>0.517</u>	0.440	<u>16.09</u>	<u>0.566</u>	0.418
	Ours ($K=4$)	17.3%	<u>18.54</u>	<u>0.583</u>	0.416	18.06	0.518	0.439	16.09	0.566	0.418
ReSplat	Init	6.2%	13.17	0.380	0.556	15.57	0.369	0.566	12.37	0.383	0.591
	Recurrent 4	6.2%	15.39	0.468	0.519	16.79	0.418	0.558	13.38	0.437	0.578
VolSplat	—	93.4%	14.12	0.366	0.607	11.81	0.251	0.700	11.71	0.320	0.650

training data. We sample input views at regular intervals along the camera trajectory, with novel views placed in-between. Images are resized so that the longer side is 476 pixels. We mask out areas of the scene not visible in any input view and exclude them from metric computation to focus evaluation on the quality of the reconstruction rather than the extrapolation. Each experiment uses 3, 6, 9, and 12 input views; results are averaged across all scenes and view splits. Detailed per-view-count results and input view reconstruction metrics appear in the supplementary material, which also reports results computed without visibility masking. We report PSNR, SSIM [45], and LPIPS [53] for novel view synthesis. For efficiency we report the relative primitive count r_c (fraction of the unmerged count that is retained), which directly correlates with rendering time and memory usage. Further evaluation details are provided in the supplementary material.

In addition to reporting the quality of the reconstruction directly produced by the three backbones, we compare against ReSplat [47], which predicts Gaussians in a $16\times$ subsampled space and refines them recurrently, and VolSplat [42], a voxel-based method producing Gaussian in a regular grid. We additionally report the performance of AnySplat’s built-in post-hoc voxelization and of heuristic moment matching of Gaussians grouped into superpixels. Due to unavailable public code, we could not include Off The Grid [49] and Fuse-and-Refine [43] in our evaluation.

5.2 Main Results

Novel View Synthesis Quality. Tab. 1 reports novel view synthesis results for all methods. Using our pipeline with the strongest backbone (DA3), the $K=1$ decoder achieves 17.41 dB PSNR averaged over all benchmarks at $r_c=4.4\%$, even slightly surpassing the DA3 unmerged quality at 16.82 dB. The gain in performance over DA3 is only achieved on MipNeRF360 and Tanks & Temples, which have much larger viewpoint changes than

Table 2: Online reconstruction quality at end-of-sequence, averaged over all 9 MipNeRF360 scenes (target views). Best per column in **bold**.

Method	#Prim.↓	r_c (%)↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
Naïve backbone	16.1 M	100.0	15.27	0.372	0.560	60
Ours $K=1$	1.0 M	6.2	15.33	0.367	0.612	<u>365</u>
Ours $K=2$	<u>2.0 M</u>	<u>12.4</u>	<u>15.41</u>	0.368	0.611	378
Ours $K=4$	4.0 M	24.8	15.45	<u>0.370</u>	<u>0.610</u>	245

DL3DV-Bench. This reflects the regularising effect of superpixel-based merging: fusing per-pixel Gaussians into compact, spatially coherent groups can remove floaters and overly large Gaussians, and reduces high-frequency noise. These effects are consistent across all backbones, with the largest gain for DepthSplat, which sometimes produces overly large Gaussians that strongly impact the reconstruction quality. LPIPS responds very sensitively to texture degradation from merging, and therefore our pipeline does not fully preserve it. ReSplat achieves a comparable r_c of 6.2% but with lower quality than our method, and without the flexibility of adapting to more powerful foundation models, as they become available. VolSplat performs significantly worse than the other methods, and fails to reproduce the reconstruction quality reported in its original paper. This is likely due to the fact that VolSplat was exclusively trained on RealEstate-10K [58], which mostly contains scenes with smaller viewpoint changes, different to our benchmarks.

Primitive Count vs. Quality Trade-off. Fig. 1 shows the quality–efficiency trade-off for the DA3 backbone, averaged over all three benchmarks. Our method ($K=1, 2, 4$) traces a favorable curve that stays close to the unmerged quality ceiling with r_c between 4.4% and 17.3%. ReSplat and VolSplat produce suboptimal results, failing to achieve the same quality as our method at comparable or higher primitive counts.

Qualitative Comparisons. Fig. 5 shows qualitative novel view synthesis results at 12 input views across all three benchmarks. Our method ($K=1$, DA3 backbone) largely reproduces the unmerged quality at a fraction of the primitive count. ReSplat occasionally yields sharper detail but introduces more severe structured artifacts elsewhere. VolSplat fully fails to reconstruct the scene structure for many scenes.

Online Reconstruction. We evaluate our pipeline in an online setting where views are integrated incrementally. Starting from an empty scene, batches of 12 views are processed in sequence, in the order of the camera trajectory, until all context views are integrated. Half of the total views are used as input and the rest as target views for evaluation. We report primitive growth averaged over the 7 scenes from MipNeRF360 that provide at least 84 context views (bicycle, bonsai, counter, flowers, garden, kitchen, room), and novel view synthesis quality at the end of the sequence averaged over all 9 MipNeRF360 scenes. Fig. 6 compares the cumulative Gaussian count of our three decoders against the naïve baseline, which accumulates all raw per-pixel Gaussians without merging. The naïve baseline grows at ~ 1.84 M Gaussians per 12 views; our $K=1$ decoder grows at ~ 0.12 M per step — a $16\times$ reduction in slope owing to within-step compression from superpixel grouping, with additional savings from cross-step merging of overlapping regions. At 84 integrated views, $K=1$ retains 0.82 M Gaussians against 12.9 M for the naïve baseline. Tab. 2 summarises per-scene-average quality at the end of each sequence. Despite holding $16\times$ fewer primitives, our $K=1$ decoder closely approaches the naïve backbone on PSNR and SSIM while trading off some perceptual fidelity (LPIPS), and renders $6.1\times$ faster.

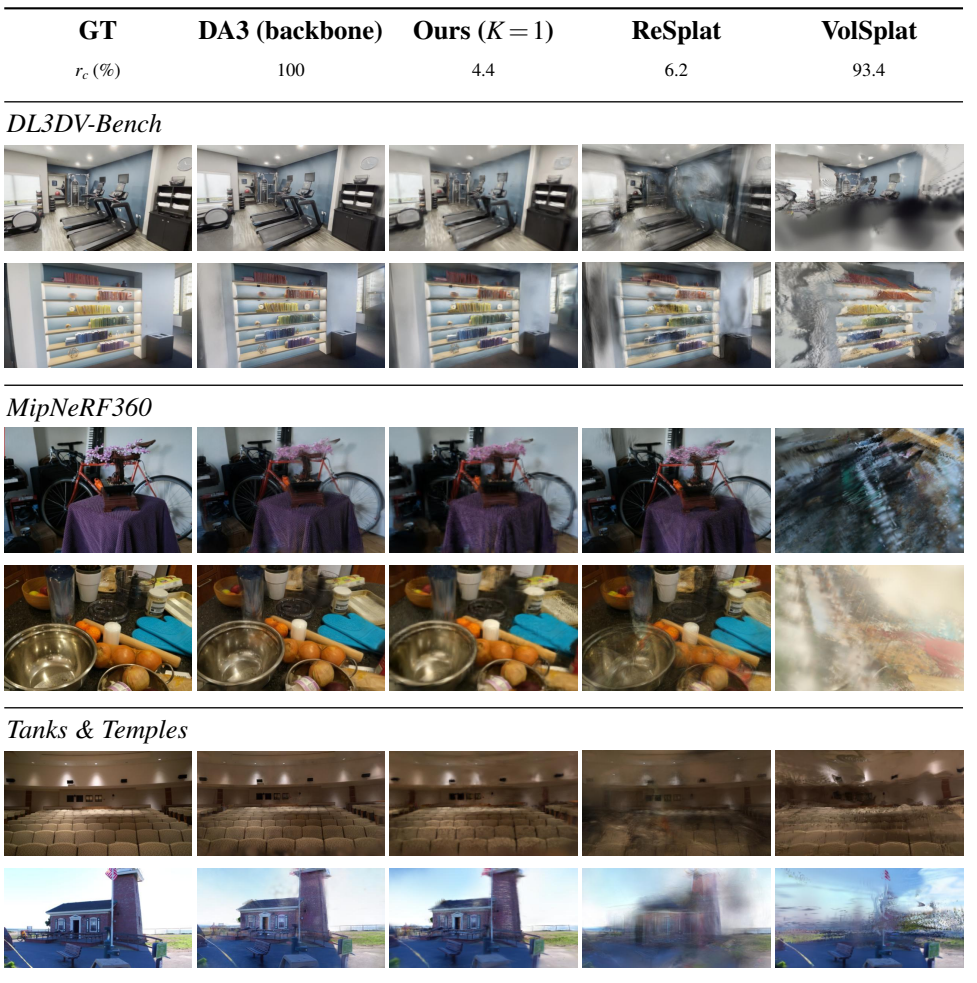


Figure 5: Qualitative novel view synthesis comparison at 12 input views (DA3 backbone). Unseen areas are highlighted in red and excluded from metric computation.

5.3 Ablation Studies

Replacing our learned merger with heuristic moment matching to combine the Gaussians within each superpixel performs worse across all metrics, confirming that a learned representation is essential for quality-preserving fusion. The magnitude of the benefit of the learned merging pipeline depends on the uniformity of the Gaussians predicted by the backbone: with the more uniform AnySplat Gaussians, moment matching performs better and the gap to our method is smaller (Tab. 1).

All further ablations use the AnySplat backbone and report the $K = 1$ decoder averaged over DL3DV-Bench, MipNeRF360, and Tanks & Temples. Tab. 3 summarizes all variants against the unmerged reference. Removing the refiner $\mathcal{M}^{\text{ref}}$ degrades all metrics (Tab. 3), indicating that the refiner contributes significantly to overall performance. Disabling cross-view merging slightly improves PSNR at the cost of a higher primitive count, suggesting

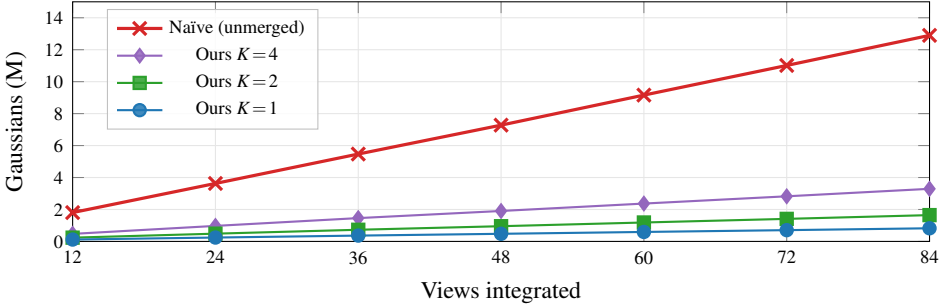


Figure 6: Gaussian count vs. integrated views in the online setting, averaged over 7 MipNeRF360 scenes.

Table 3: Ablation study (AnySplat backbone, $K=1$, averaged over all benchmarks). Best in **bold**, second best underlined.

Variant	r_c (%)↓	PSNR↑	SSIM↑	LPIPS↓
<i>Backbone (unm.)</i>	100.0%	12.59	0.374	0.486
Ours (full)	<u>4.4%</u>	14.14	0.423	0.533
w/o refiner	4.7%	13.49	0.389	0.534
w/o cross-view	5.3%	<u>14.13</u>	<u>0.424</u>	<u>0.531</u>
zero shot larger SPs	2.0%	13.97	0.424	0.548
zero shot smaller SPs	8.7%	14.11	0.420	0.523

that cross-view fusion reduces redundancy at the cost of a slight quality drop. Within-view merging is the dominant source of compression in our pipeline, while cross-view merging is an optional addition that provides a smaller, but not negligible, further reduction: relative to within-view merging alone, cross-view merging retains 84.6% of the Gaussians, a compression roughly comparable to AnySplat’s built-in post-hoc voxelization (83.2%, Tab. 1).

Superpixel Algorithm Comparison. To evaluate the segmentation quality of different superpixel algorithms across a wide range of segment counts, without the cost of the full pipeline evaluation, we use a proxy metric: each pixel in an image is replaced by the mean color of its superpixel, and the resulting image is compared to the ground truth. This measures how faithfully a given superpixel layout can represent the visual image content, which determines the upper bound on merging quality. We apply this proxy to 1 000 images (100 scenes $\times$ 10 views) from the training set and report PSNR against the original pixels. Fig. 7 shows the quality vs. superpixel count trade-off for BASS [49] with uniform seeding, BASS with our Shi-Tomasi corner detector seeding, and SLIC [11]. BASS with Shi-Tomasi corner seeding consistently dominates the other algorithms at all budgets, achieving higher proxy PSNR at equal superpixel count. Based on these results, we evaluate the full pipeline (without retraining) with BASS and Shi-Tomasi seeding at three superpixel size settings, which yield $\sim 23k$, $\sim 12k$, and $\sim 5k$ superpixels per image on average.

Larger superpixels reduce the relative primitive count r_c at the cost of quality, e.g., from 14.14 dB PSNR to 13.97 dB PSNR at roughly half the r_c (Tab. 3). This confirms that finer superpixels better preserve reconstruction quality, at the cost of a higher r_c , with diminishing returns at very small superpixel sizes. Using larger superpixels therefore provides a similar trade-off to using the higher K decoder heads, with the decoder heads being slightly more effective at the same r_c . For example, the $K=2$ decoder achieves 14.45 dB PSNR at $r_c = 8.9\%$, while the $K=4$ decoder with larger superpixels achieves 14.45 dB PSNR at a similar

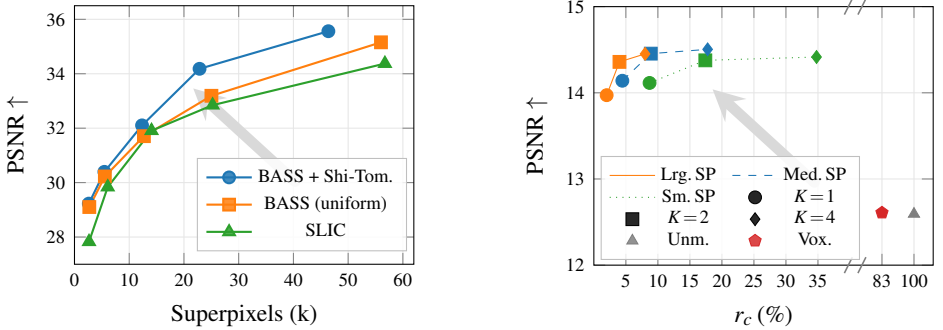


Figure 7: *Left*: Proxy PSNR vs. superpixel count for three segmentation algorithms (100 scenes $\times$ 10 views). Each pixel is replaced by its superpixel mean color and compared to the original image. *Right*: NVS PSNR vs. relative primitive count for three SP sizes (AnySplat backbone, zero-shot, averaged over all benchmarks). Arrows indicate favorable direction.

$r_c = 8.0\%$. This provides the user with two complementary mechanisms to flexibly adjust the quality–efficiency trade-off according to their needs, both without retraining.

6 Conclusion

We present a structure-aware approach for strongly compressing Gaussian splatting representations generated by feed-forward networks. Our superpixel-based primitive merging strategy intelligently consolidates per-pixel Gaussians into a compact, content-adaptive representation. We group Gaussians via saliency-guided superpixel segmentation, encode each group into a Feature Gaussian with a learned encoder, and match and merge Feature Gaussians across views. A level-of-detail decoder provides explicit control over the output primitive count at inference, enabling flexible quality–efficiency trade-offs. The pipeline operates as a backbone-agnostic post-processing module that retains the bulk of the reconstruction quality across three benchmarks and three feed-forward backbones at just $1/20^{\text{th}}$ of the Gaussians, accepting a slight reduction in fine-detail fidelity in exchange for dramatically improved efficiency. Compared to other reduced-primitive methods, our approach provides more robust and higher quality reconstructions, particularly in sparse-view settings. Additionally, for large viewpoint changes, the merged primitives can even improve some metrics by suppressing common FF artifacts such as semi-transparent fog, or high-frequency noise.

Limitations. Our method depends on the quality of the underlying feed-forward reconstruction; if the initial primitives are of low quality, merging may not recover a good representation. The matching and merging pipeline introduces a computational overhead of 540 ms compared to direct feed-forward inference (details in the supplementary material). While SSIM and PSNR metrics are preserved or improved, the LPIPS metric is slightly degraded due to some loss of fine detail in the merged representation.

Future work. Promising directions include extensions to dynamic scenes where temporal consistency must be maintained, and replacing the algorithmic superpixel grouping with a learned grouping mechanism that can be optimized jointly with the merging pipeline. LPIPS degradation could potentially be mitigated by using a richer output representation, such as Textured Gaussians [8] or 3D Convex Splatting [14].

Acknowledgements. This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy, EXC-2070 – 390732324 – PhenoRob and by the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG). As part of the IPCEI ME/CT the project is supported by the Federal Ministry for Economic Affairs and Energy, by the Ministry for Economic Affairs, Labor and Tourism of Baden-Württemberg and financed by the European Union – NextGenerationEU.

References

- [1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(11):2274–2282, 2012. doi: 10.1109/TPAMI.2012.120.
- [2] Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michaël Iwanicki, and Sébastien Hillaire. *Real-Time Rendering*. CRC Press, 4 edition, 2018. doi: 10.1201/b22086.
- [3] Yinlong Bai, Hongxin Zhang, Sheng Zhong, Junkai Niu, Hai Li, Yijia He, and Yi Zhou. MemGS: Memory-Efficient Gaussian Splatting for Real-Time SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2025.
- [4] Isabela B. Barcelos, Felipe D.C. Belem, Leonardo D.M. Joao, Zenilton K.G.D. Patrocínio Jr., Alexandre X. Falcao, and Silvio J.F. Guimarães. A Comprehensive Review and New Taxonomy on Superpixel Segmentation. *ACM Computing Surveys*, 56(8): 1–39, 2024. doi: 10.1145/3643826.
- [5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [6] Brian Chao, Hung-Yu Tseng, Lorenzo Porzi, Chen Gao, Tuotuo Li, Qinbo Li, Ayush Saraf, Jia-Bin Huang, Johannes Kopf, Ming-Hsuan Yang, and Yung-Yu Chuang. Textured Gaussians for Enhanced 3D Scene Appearance Modeling. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [7] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian Splats from Image Pairs for Scalable Generalizable 3D Reconstruction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [8] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.
- [9] Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast Feedforward 3D Gaussian Splatting Compression. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2025.

- [10] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC++: Towards 100X Compression of 3D Gaussian Splatting. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 47(11):10210–10226, 2025. doi: 10.1109/TPAMI.2025.3594066.
- [11] 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 *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.
- [12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2021.
- [13] Guangchi Fang and Bing Wang. Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.
- [14] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [15] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, and Feng Zhao. AnySplat: Feed-Forward 3D Gaussian Splatting from Unconstrained Views. *ACM Trans. on Graphics (TOG)*, 44(6):1–16, 2025. doi: 10.1145/3763326.
- [16] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Y.K. Lau, Feng Gao, Yin Yang, and Chenfanfu Jiang. VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2024.
- [17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. on Graphics (TOG)*, 42(4):1–14, 2023. doi: 10.1145/3592433.
- [18] Bernhard Kerbl, Andréas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM Trans. on Graphics (TOG)*, 43(4):1–15, 2024. doi: 10.1145/3658160.
- [19] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Trans. on Graphics (TOG)*, 36(4):1–13, 2017. doi: 10.1145/3072959.3073599.
- [20] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian Representation for Radiance Field. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

- [21] Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Optimized Minimal 3D Gaussian Splatting. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2025.
- [22] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2019.
- [23] Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the Visual Space from Any Views. *arXiv preprint*, arXiv:2511.10647, 2025.
- [24] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Anirudha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A Large-Scale Scene Dataset for Deep Learning-Based 3D Vision. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [25] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2019.
- [26] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3DGS: High-Quality Radiance Fields with Limited Resources. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH) Asia*, 2024.
- [27] 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 *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [28] 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 *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.
- [29] 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 *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2026.
- [30] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [31] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One Image to High-Quality 3D Object Generation Using Both 2D and 3D Diffusion Priors. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2024.

- [32] Xuanchi Ren, Yifan Lu, Hanxue Liang, Zhangjie Wu, Huan Ling, Mike Chen, Sanja Fidler, Francis Williams, and Jiahui Huang. SCube: Instant Large-Scale Scene Reconstruction Using VoxSplats. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.
- [33] Kyle Sargent, Zizhang Li, Tanmay Shah, Charles Herrmann, Hong-Xing Yu, Yunzhi Zhang, Eric Ryan Chan, Dmitry Lagun, Li Fei-Fei, Deqing Sun, and Jiajun Wu. ZeroNVS: Zero-Shot 360-Degree View Synthesis from a Single Image. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [34] Hanno Scharf. *Optimale Operatoren in der digitalen Bildverarbeitung*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2000.
- [35] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905, 2000. doi: 10.1109/34.868688.
- [36] Jianbo Shi and Carlo Tomasi. Good Features to Track. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1994.
- [37] Stanislaw Szymonowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter Image: Ultra-Fast Single-View 3D Reconstruction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [38] Adam Tonderski, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. NeuRAD: Neural Rendering for Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [39] Roy Uziel, Meitar Ronen, and Oren Freifeld. Bayesian Adaptive Superpixel Segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.
- [40] Haishan Wang, Mohammad Hassan Vali, and Arno Solin. Smol-GS: Compact Representations for Abstract 3D Gaussian Splatting. *arXiv preprint*, arXiv:2512.00850, 2025.
- [41] Weijie Wang, Donny Y. Chen, Zeyu Zhang, Duochao Shi, Akide Liu, and Bohan Zhuang. ZPressor: Bottleneck-Aware Compression for Scalable Feed-Forward 3DGS. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2025.
- [42] Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Zheng Zhu, Donny Y. Chen, and Bohan Zhuang. VolSplat: Rethinking Feed-Forward 3D Gaussian Splatting with Voxel-Aligned Prediction. *arXiv preprint*, arXiv:2509.19297, 2025.
- [43] Yiming Wang, Lucy Chai, Xuan Luo, Michael Niemeyer, Manuel Lagunas, Stephen Lombardi, Siyu Tang, and Tiancheng Sun. Learning Efficient Fuse-and-Refine for Feed-Forward 3D Gaussian Splatting. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2025.
- [44] Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. FreeSplat: Generalizable 3D Gaussian Splatting Towards Free-View Synthesis of Indoor Scenes. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

- [45] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.
- [46] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [47] Haofei Xu, Daniel Barath, Andreas Geiger, and Marc Pollefeys. ReSplat: Learning Recurrent Gaussian Splatting. *arXiv preprint*, arXiv:2510.08575, 2025.
- [48] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian Splatting and Depth. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [49] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. GRM: Large Gaussian Reconstruction Model for Efficient 3D Reconstruction and Generation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.
- [50] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. GS-SLAM: Dense Visual SLAM with 3D Gaussian Splatting. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [51] 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.
- [52] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: Large Reconstruction Model for 3D Gaussian Splatting. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.
- [53] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [54] 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 *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.
- [55] Yuhang Zhang, Richard Hartley, John Mashford, and Stewart Burn. Superpixels via Pseudo-Boolean Optimization. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2011.
- [56] Tingyu Zhao, Bo Peng, Zhenguang Zhang, Daipeng Yang, and Xi Wu. Content-Aware Dynamic Superpixel Segmentation. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

- [57] Yuhang Zheng, Xiangyu Chen, Yupeng Zheng, Songen Gu, Runyi Yang, Bu Jin, Pengfei Li, Chengliang Zhong, Zengmao Wang, Lina Liu, Chao Yang, Dawei Wang, Zhen Chen, Xiaoxiao Long, and Meiqing Wang. GaussianGrasper: 3D Language Gaussian Splatting for Open-Vocabulary Robotic Grasping. *IEEE Robotics and Automation Letters (RA-L)*, 9(9):7827–7834, 2024. doi: 10.1109/LRA.2024.3432348.
- [58] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning View Synthesis Using Multiplane Images. *ACM Trans. on Graphics (TOG)*, 37(4):1–12, 2018. doi: 10.1145/3197517.3201323.
- [59] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. DrivingGaussian: Composite Gaussian Splatting for Surrounding Dynamic Autonomous Driving Scenes. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [60] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. EWA Volume Splatting. In *Proc. of the IEEE Visualization Conf. (VIS)*, 2001.