CAG-GS: Consistent Anchor Guided Gaussian Splatting for Large-Scale Scene Rendering

Shijie Xu[1,2], **Qiulei Dong**[1,2*]

[1]State Key Laboratory of Multimodal Artificial Intelligence Systems,
Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
xushijie2023@ia.ac.cn, qldong@nlpr.ia.ac.cn

Abstract

Recently, 3D Gaussian Splatting for scene rendering has attracted much attention in computer vision and graphics, but generally suffers from large burdens of both computation and storage when handling large-scale scenes. Some existing works in literature employ a divide-and-conquer strategy for alleviating this issue, where an input large scene is divided into lots of local blocks, and each block is handled separately. However, such a strategy generally leads to limited performance due to the inevitable inconsistency among the 3D Gaussians from different blocks. To address this problem, we propose a Consistent Anchor Guided Gaussian Splatting for large-scale scene rendering under the divide-and-conquer strategy, called CAG-GS. In CAG-GS, a set of learnable anchors for each local block is injected with the corresponding semantic features from a pre-trained semantic segmentation model SAM2 through an explored semantic mapping module, and then these anchors are used to predict the attributes of 3D Gaussians. Moreover, we explore a coarse-to-fine training strategy for CAG-GS, where each local block is optimized independently while being guided by globally consistent semantics. Extensive experimental results on five large-scale scenes demonstrate the superiority of the proposed method over five state-of-the-art methods in most cases.

1 Introduction

In recent years, 3D Gaussian Splatting (3D-GS) (Kerbl et al. 2023) has emerged as a promising rendering technique. By employing an explicit 3D Gaussian scene representation and an efficient differentiable rendering pipeline, it achieves high-fidelity and real-time rendering performance. However, the basic 3D-GS technique requires a vast number of Gaussians when handling large-scale scenes, leading to severe computational and storage burdens.

To address this issue, some methods in literature (Lin et al. 2024; Kerbl et al. 2024; Chen et al. 2024b) generally adopt a divide-and-conquer strategy, where the scene is divided into multiple blocks, each assigned a subset of training views for independent optimization. However, given the significant variations in visual appearance, including lighting and color, across the captured image collections, such a division strategy inevitably leads to an imbalanced distribution of training data across blocks. Due to the lack of effective global consistency constraints, existing methods are subject to inconsistent optimization objectives between blocks resulting from this imbalance (Fan et al. 2024a; Wu et al. 2025). As a result, the overall quality of the merged reconstruction is degraded. For instance, Figure 1 illustrates that the existing methods tend to introduce floaters in the airspace to fit the imbalanced lighting distribution, resulting in unrealistic brightness variations in the synthesized novel views.

To alleviate the above problem, we propose a Consistent Anchor Guided Gaussian Splatting for large-scale scene rendering, called CAG-GS, which incorporates semantic guidance to improve global consistency under the divide-and-conquer strategy. In CAG-GS, the scene is implicitly represented by a set of anchors, each associated with a learnable feature to neurally predict the attributes of nearby Gaussians (Lu et al. 2024; Ren et al. 2024). Building upon this foundation, a semantic mapping module is introduced to inject semantic information into anchor features for constructing semantically consistent implicit Gaussians. Concretely, the semantic mapping module is trained using the visual features extracted from a pre-trained segmentation model SAM2 (Ravi et al. 2025) as supervision. In addition, a coarse-to-fine training strategy is explored for the proposed CAG-GS. We first train a coarse global model using all training views and then divide the scene into multiple blocks. The training views are assigned to each block according to the blending weights during rendering, ensuring sufficient supervision for each block. Subsequently, each block is further fine-tuned using the assigned training views. With the semantic guidance from SAM2 and the priors provided by the coarse model, the subsequent block-wise fine-tuning enables finer-grained modeling of scene details while preserving global consistency. Finally, the blocks are cropped along their division boundaries and then seamlessly merged to obtain the final reconstruction.

In summary, the main contributions of this paper are as follows:

- We propose CAG-GS for large-scale scene rendering, which uses an anchor-based Gaussian construction that injects semantic features from a pre-trained segmentation model into the anchors. Thanks to the injected semantic features, the anchors from different blocks are endowed with certain consistent scene information so that the in-

*Corresponding author

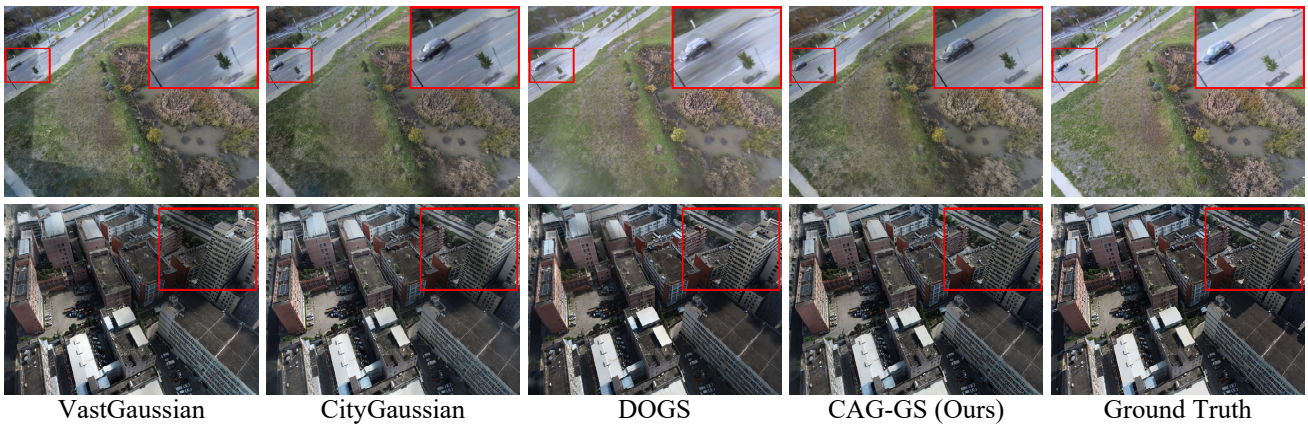


Figure 1: Visual comparison with the existing methods. The compared methods lack effective global consistency constraints and thus struggle to handle complex lighting variations within the large-scale scenes. In contrast, our method incorporates globally consistent semantic guidance, which implicitly regulate the color distribution of Gaussians, leading to more realistic brightness variations in novel views.

consistency problem among the 3D Gaussians caused by the divide-and-conquer strategy is alleviated to some extent.

- We explore a coarse-to-fine strategy for training the proposed CAG-GS, which enables fine-grained modeling of scene details while preserving global consistency across the entire scene.
- We conduct extensive experiments for evaluating the proposed CAG-GS in comparison to five SOTA methods. The results in Section 4 demonstrate the effectiveness of the proposed CAG-GS.

2 Related Work

3D Gaussian Splatting

By representing scenes with explicit 3D Gaussians, 3D Gaussian Splatting (3D-GS) (Kerbl et al. 2023) achieves high-fidelity and real-time rendering. Subsequent developments refined vanilla 3D-GS in various dimensions, including rendering quality (Yu et al. 2024; Liang et al. 2024; Yan et al. 2024), speed (Navaneet et al. 2023; Fan et al. 2024c; Hanson et al. 2025), and storage efficiency (Chen et al. 2024c; Wang et al. 2024; Liu et al. 2024a). Moreover, several works extended it to related tasks such as surface reconstruction (Huang et al. 2024; Chen et al. 2024a; Huang et al. 2025) and semantic understanding (Qin et al. 2024; Zhou et al. 2024; Ji et al. 2025).

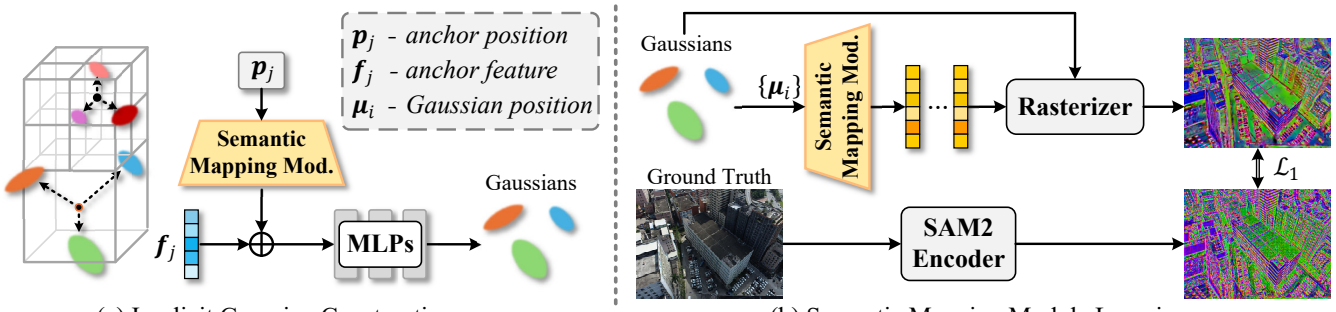
However, the gradient-based heuristic densification strategy adopted in 3D-GS can lead to redundant Gaussians, which in turn increase rendering latency and compromise robustness to view changes. Although several works have alleviated this issue through efficient pruning algorithms (Navaneet et al. 2023; Fan et al. 2024b,c) and controllable densification strategies (Mallick et al. 2024; Li et al. 2024; Zeng et al. 2025), the resulting point-cloud-like 3D Gaussians remain unorganized. In light of this, Scaffold-GS (Lu et al. 2024) proposed a structured scene representation, where the scene was discretized into a voxel grid, and each voxel instantiated a set of implicit Gaussians for rendering. Octree-GS (Ren et al. 2024) extended the above work by incorporating multi-resolution grids and tailored optimization strategies, thus improving the capability of modeling scene geometry across varying levels of detail.

Large-Scale Scene Reconstruction

Large-scale scene reconstruction has long been a central research topic in computer vision. Traditional methods aim to recover sparse scene geometry and camera parameters from the input images (Snavely, Seitz, and Szeliski 2008; Agarwal et al. 2011; Liu and Dong 2025). These are then used to generate a mesh-based representation of the scene (Bleyer, Rhemann, and Rother 2011; Lafarge et al. 2013).

With the advancement of neural rendering techniques, various methods have introduced novel scene representations and differentiable rendering pipelines for large-scale view synthesis. For instance, concurrent methods Block-NeRF (Tancik et al. 2022) and Mega-NeRF (Turki, Ramanan, and Satyanarayanan 2022) adopted the divide-and-conquer strategy, training separate NeRF modules to represent sub-regions of the scene. Switch-NeRF (Mi and Xu 2023) introduced a gating network to fuse the outputs of the NeRF modules, thereby improving consistency of the entire scene.

Recent methods leverage 3D-GS for large-scale scenes owing to its superior rendering efficiency. VastGaussian (Lin et al. 2024) was the first to incorporate the divide-and-conquer strategy into the 3D-GS framework, where each block is represented by an independent Gaussian model. In the follow-up work, CityGaussian (Liu et al. 2024b) proposed to first train a coarse global Gaussian model, which serves as the basis for scene division and provides global priors for subsequent block-wise fine-tuning. To further improve consistency across blocks, DOGS (Chen and Lee 2024) and Momentum-GS (Fan et al. 2024a) jointly optimized different blocks through inter-GPU communication.



(a) Implicit Gaussian Construction (b) Semantic Mapping Module Learning

Figure 2: Anchor representation in CAG-GS. (a) Implicit Gaussian Construction: the semantic mapping module encodes anchor coordinates into semantic vectors, which are subsequently fused with anchor-specific features and passed through MLP decoders to predict the attributes of the implicit Gaussians. (b) Semantic Mapping Module Learning: the semantic mapping module is trained by aligning its outputs with semantic features extracted by SAM2.

Our method shares similarities with Momentum-GS in the use of implicit scene representation and the divide-and-conquer optimization strategy. Unlike Momentum-GS, which implicitly regularizes anchor features by sharing the same MLP decoders across different blocks, our method directly incorporates consistent semantic information to guide feature learning.

3 Method

In this section, we begin by reviewing the fundamentals of vanilla 3D-GS. Then, we describe the proposed CAG-GS for large-scale scene rendering and its training strategy in detail.

Preliminary

3D-GS represents scenes using anisotropic 3D Gaussian primitives, where each Gaussian G_i is defined by its position $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$. The covariance matrix $\boldsymbol{\Sigma}_i$ is parameterized by a scaling vector $\boldsymbol{s}_i$ and a rotation quaternion $\boldsymbol{q}_i$. For color representation, each Gaussian is further associated with opacity σ_i and spherical harmonic coefficients.

During rendering, G_i is projected onto the image plane and approximated as the 2D counterpart G'_i (Zwicker et al. 2001), and its spherical harmonic coefficients are decoded into view-dependent color $\boldsymbol{c}_i$. Subsequently, the rasterizer sorts the 3D Gaussians in depth order and performs α-blending to composite their colors at each pixel:

$$\boldsymbol{C}(\boldsymbol{x}') = \sum_{i \in \mathcal{I}(\boldsymbol{x}')} \boldsymbol{c}_i \alpha_i \prod_{k=1}^{i-1} (1 - \alpha_k), \ \alpha_i = \sigma_i G'_i(\boldsymbol{x}'), \quad (1)$$

where $\boldsymbol{x}'$ is the queried pixel and $\mathcal{I}(\boldsymbol{x}')$ represents the sorted indices of the Gaussians that contribute to this pixel.

For each training view, the 3D Gaussians are optimized by minimizing the photometric loss function between the rendered image $\boldsymbol{I}$ and the ground truth image $\hat{\boldsymbol{I}}$:

$$\mathcal{L}_{\text{3DGS}} = (1 - \lambda)\mathcal{L}_1(\boldsymbol{I}, \hat{\boldsymbol{I}}) + \lambda \mathcal{L}_{\text{SSIM}}(\boldsymbol{I}, \hat{\boldsymbol{I}}), \quad (2)$$

where $\mathcal{L}_1$ and $\mathcal{L}_{\text{SSIM}}$ denote the L1 loss and the SSIM loss (Wang et al. 2004), respectively, and λ is a weighting coefficient.

Consistent Anchor Guided Gaussian Splatting

In this subsection, we present the proposed CAG-GS. We first provide the definition of anchors, and then describe the process of constructing implicit Gaussians based on these anchors.

Anchor Definition. Given the large variations in geometry and texture scales in large-scale scenes, a hierarchical scene representation based on Level-of-Detail (LoD) is adopted to model details at different granularities, following Octree-GS (Ren et al. 2024). Prior to training, the sparse point cloud $\mathcal{P}$ obtained from SfM is voxelized at L different resolutions to generate a set of anchor points:

$$\mathcal{A} = \left\{ \left\lfloor \frac{\boldsymbol{p}}{\delta/2^l} \right\rceil \cdot \delta/2^l \,\middle|\, \boldsymbol{p} \in \mathcal{P}, l = 0, \ldots, L-1 \right\}, \quad (3)$$

where $\lfloor \cdot \rceil$ is the element-wise rounding operation, and δ represents the base voxel size. Each anchor $\mathcal{A}_j$ is uniquely determined by its LoD level l_j and position $\boldsymbol{p}_j$, and is further assigned a learnable feature $\boldsymbol{f}_j \in \mathbb{R}^d$ and K learnable spatial offsets $\{\boldsymbol{o}_j^{(1)}, \ldots, \boldsymbol{o}_j^{(K)}\} \in \mathbb{R}^{3K}$. Here, D denotes the dimensionality of the anchor features and K denotes the expected number of Gaussians to be generated per anchor. Anchors with higher LoD levels are located on denser grids, allowing them to capture finer details.

Implicit Gaussian Construction. We introduce a semantic mapping module to incorporate semantic information, guiding the construction of implicit Gaussians. This module learns to map 3D spatial coordinates into a semantic feature space.

During rendering, visible anchors are first selected based on their LoD levels and their distances from the target viewpoint. For instance, when observed from a distant viewpoint, anchors with higher LoD levels are excluded from the rendering pipeline. As illustrated in Figure 2(a), for each selected anchor, the semantic mapping module encodes its spatial position into a high-dimensional semantic vector, which is then fused with its original feature, yielding an enhanced anchor feature as follows:

$$\tilde{\boldsymbol{f}}_j = W \cdot \gamma(\boldsymbol{p}_j) + \boldsymbol{f}_j, \quad (4)$$

where $\gamma : \mathbb{R}^3 \rightarrow \mathbb{R}^D$ represents the semantic mapping module, D denotes the dimensionality of its output, and $W \in \mathbb{R}^{d \times D}$ is a learnable weight matrix. Finally, the semantic feature is fed into MLP decoders to predict the opacities, colors, and covariances of the implicit 3D Gaussians, while their centers are determined by applying K learnable spatial offsets to the anchor position.

In our method, the semantic mapping module is implemented by a compact multi-resolution hash grid (Müller et al. 2022). During training, it enables efficient queries of semantic features at arbitrary spatial locations through hash-based feature indexing and trilinear feature interpolation. Notably, since the feature hash tables are fixed in size, the memory consumption of the semantic mapping module remains constant, regardless of the number of anchors.

Coarse-to-Fine Training Strategy

CAG-GS is designed to incorporate semantic guidance to improve global consistency under the divide-and-conquer strategy. To this end, we explore a coarse-to-fine training strategy for the proposed CAG-GS, consisting of three stages: coarse global model training, scene division, and block-wise local fine-tuning. In this subsection, we first present the optimization objectives employed in both the coarse and fine stages of training, then elaborate on the scene division strategy, and finally describe the process of merging the blocks into the final reconstruction.

Optimization Objective. Beyond the photometric loss used in vanilla 3D-GS, we additionally introduce a semantic feature alignment loss and geometric consistency constraints to jointly optimize the proposed CAG-GS.

The semantic feature alignment loss is introduced to train the semantic mapping module by encouraging its embedding space to align with the feature space defined by SAM2 (Ravi et al. 2025). We use SAM2 as the source of semantic guidance due to its fine-grained perception of semantically homogeneous regions in images, which facilitates the construction of semantically consistent Gaussians in 3D space. As illustrated in Figure 2(b), during training, the positions of the implicit Gaussians are fed into the semantic mapping module to obtain the corresponding embedding vectors. Then, the 3D-GS rasterizer is reused to composite the embedding vectors into the feature map as follows:

$$\boldsymbol{F}(\boldsymbol{x}') = \sum_{i \in \mathcal{I}(\boldsymbol{x}')} \gamma(\boldsymbol{\mu}_i) \alpha_i \prod_{k=1}^{i-1} (1 - \alpha_i), \ \alpha_i = \sigma_i G_i'(\boldsymbol{x}'), \tag{5}$$

where $\gamma(\boldsymbol{\mu}_i)$ represents the embedding vector associated with Gaussian G_i. To reduce computational overhead, feature maps are synthesized at a downscaled resolution while maintaining the original aspect ratio.

The semantic feature alignment loss is defined as the L1 loss between the feature map $\boldsymbol{F}$ and the extracted semantic features:

$$\mathcal{L}_{\text{feat}} = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x}' \in \mathcal{X}} \left\| \boldsymbol{F}(\boldsymbol{x}') - \hat{\boldsymbol{F}}(\boldsymbol{x}') \right\|_1, \tag{6}$$

where $\mathcal{X}$ denotes the set of pixels, and $\hat{\boldsymbol{F}}$ represents the per-pixel features extracted using SAM2. For training efficiency, these features are extracted and stored during pre-processing and loaded at each training iteration.

To improve the robustness of the synthesized feature maps to view changes, additional geometric consistency constraints are introduced. These constraints include a flattening loss and a normal consistency loss, encouraging the implicit Gaussians to align with the local surface of the scene. First, the flattening loss is applied to compress the Gaussians along their shortest axis:

$$\mathcal{L}_{\text{flat}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \min(\boldsymbol{s}_i), \tag{7}$$

where $\mathcal{I}$ denotes the indices of all implicit Gaussians and $\boldsymbol{s}_i \in \mathbb{R}^3$ is the scaling vector of Gaussian G_i. Thus, each Gaussian defines a local plane, with its normal $\boldsymbol{n}_i$ given by the direction of its shortest axis, and its distance to the camera center computed as $\boldsymbol{d}_i = \boldsymbol{n}_i^{\mathrm{T}}(\boldsymbol{\mu}_i + \boldsymbol{R}^{\mathrm{T}}\boldsymbol{t})$, where $(\boldsymbol{R}, \boldsymbol{t})$ denotes the transformation from the world coordinates to the camera frame. The normal map $\boldsymbol{N}_r(\boldsymbol{x}')$ and the unbiased depth map $\boldsymbol{D}(\boldsymbol{x}')$ are rendered following PGSR (Chen et al. 2024a). Subsequently, the unbiased depth map is back-projected into 3D space to recover per-pixel 3D coordinates. For each pixel, the surface normal $\boldsymbol{N}(\boldsymbol{x}')$ is estimated from the 3D coordinates of its adjacent pixels. The normal consistency loss is defined as follows:

$$\mathcal{L}_{\text{norm}} = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x}' \in \mathcal{X}} w(\boldsymbol{x}') \left\| \boldsymbol{N}(\boldsymbol{x}') - \boldsymbol{N}_r(\boldsymbol{x}') \right\|_1, \tag{8}$$

where the weighting term $w(\boldsymbol{x}')$ is given by

$$w(\boldsymbol{x}') = \left(1 - |\nabla \hat{\boldsymbol{I}}|(\boldsymbol{x}')\right)^2, \tag{9}$$

and $|\nabla \hat{\boldsymbol{I}}|$ denotes the normalized image gradient magnitude, scaled to the range $[0, 1]$.

Accordingly, the overall loss function is formulated as:

$$\mathcal{L} = \mathcal{L}_{\text{3DGS}} + \lambda_1 \mathcal{L}_{\text{feat}} + \lambda_2 \mathcal{L}_{\text{flat}} + \lambda_3 \mathcal{L}_{\text{norm}}, \tag{10}$$

where λ_1, λ_2, and λ_3 denote the weighting coefficients for the semantic alignment, flattening, and normal consistency losses, respectively.

Scene Division Strategy. The divide-and-conquer strategy is required to balance workload across different blocks and provide sufficient supervision for each block. We design a scene division strategy based on the trained coarse model, which primarily involves determining block boundaries, assigning training views, and instantiating local models.

To obtain the block boundaries, a scene bounding box is first estimated based on the anchor positions in the coarse model. We sequentially divide the scene along the two axes of the bounding box, ensuring that each block contains an approximately equal number of training views to balance the optimization workload across blocks (Lin et al. 2024).

During subsequent training view assignment, views within each block boundary are initially assigned to the corresponding block. To ensure sufficient supervision for each block, additional views outside the boundary are selected according to blending weights during rendering. Specifically,

Scene	*Building*			*Rubble*			*Residence*			*Sci-Art*			*SmallCity*		
Metric	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
VastGaussian†	21.80	0.728	0.225	25.20	0.742	0.264	21.01	0.699	0.261	22.64	0.761	0.261	<u>28.33</u>	0.835	0.220
CityGaussian	21.55	0.778	0.246	25.77	0.813	0.228	22.00	0.813	0.211	21.39	0.837	0.230	27.46	0.865	0.204
DOGS	22.73	0.759	0.204	25.78	0.765	0.257	21.94	0.740	0.244	**24.42**	0.804	0.219	**28.58**	0.847	0.219
Octree-GS	**23.66**	0.776	0.267	25.34	0.763	0.299	<u>22.29</u>	0.762	0.288	23.38	0.828	0.240	27.31	0.849	0.229
Momentum-GS	23.23	**0.815**	**0.194**	<u>25.93</u>	**0.827**	**0.201**	22.21	<u>0.818</u>	**0.197**	23.02	<u>0.856</u>	<u>0.205</u>	28.01	**0.880**	<u>0.179</u>
CAG-GS (Ours)	<u>23.46</u>	<u>0.814</u>	<u>0.198</u>	**26.18**	<u>0.816</u>	<u>0.221</u>	**22.62**	**0.820**	<u>0.202</u>	<u>23.80</u>	**0.863**	**0.201**	27.79	<u>0.879</u>	**0.179**

Table 1: Quantitative comparison of rendering quality on the five scene. The methods listed in the first block adopt explicit scene representations, while others adopt implicit ones. The best and second best results are highlighted in **bold** and <u>underline</u>, respectively. VastGaussian† denotes the DOGS implementation, with the results from the DOGS paper. The results of Momentum-GS are cited from the v1 version of its paper, and those of other comparative methods are from their original papers.

the blending weight of Gaussian G_i when querying pixel $\boldsymbol{x}'$ is defined as follows:

$$\hat{\alpha}_i(\boldsymbol{x}') = \sigma_i G'_i(\boldsymbol{x}') \prod_{k=1}^{i-1} \big(1 - \sigma_t G'_t(\boldsymbol{x}')\big). \tag{11}$$

By accumulating the blending weights of the Gaussians constructed by anchor $\mathcal{A}_j$ over all pixels, the importance score of $\mathcal{A}_j$ for the current view is computed:

$$s(\mathcal{A}_j) = \sum_{i \in \mathcal{I}_j} \sum_{\boldsymbol{x}' \in \mathcal{X}} \hat{\alpha}_i(\boldsymbol{x}'), \tag{12}$$

where $\mathcal{I}_j$ denotes the set of indices corresponding to the Gaussians constructed by anchor $\mathcal{A}_j$. The ratio between the sum of importance scores of anchors within the block boundary and the total importance score is computed. If this ratio exceeds a threshold ε, the corresponding training view is assigned to the target block.

Finally, a local model is instantiated for each block, with modules and anchors initialized from the coarse model. The parameters of the MLP decoders and the semantic mapping module are inherited from the coarse model. To initialize anchors, we iterate over the assigned training views, and select those with positive importance scores as initial anchors. This process introduces a considerable number of out-of-boundary anchors. Although these anchors will be discarded during final merging, their involvement in the fine-tuning helps provide reliable estimates for neighboring regions.

Block Merging. Once all blocks are fine-tuned, anchors falling outside the block boundaries are discarded, and the remaining anchors are merged along the shared borders. The features of these anchors are then fused with semantic features queried from their corresponding semantic mapping modules. Finally, the semantic mapping modules of all local models are discarded, while the MLP decoders are retained for predicting Gaussian attributes.

4 Experiments

Experimental setup

Datasets. We conduct experiments on five large-scale scenes drawn from three benchmark datasets: *Building* and *Rubble* from Mill19 (Turki, Ramanan, and Satyanarayanan 2022), *Residence* and *Sci-Art* from UrbanScene3D (Lin et al. 2022), and *SmallCity* from MatrixCity (Li et al. 2023). For the Mill19 and UrbanScene3D datasets, the images are downsampled to 25% of their original width and height, and the camera parameters are provided by MegaNeRF (Turki, Ramanan, and Satyanarayanan 2022). For the *SmallCity* scene, we use aerial images and resize them to a width of 1600 pixels while maintaining the original aspect ratio. The associated camera parameters are provided by the MatrixCity dataset. For all scenes, we use COLMAP (Schönberger and Frahm 2016) to reconstruct sparse point clouds from the images using the provided camera parameters.

Metrics. We evaluate rendering quality using the standard PSNR↑, SSIM↑ (Wang et al. 2004), and LPIPS↓ (Zhang et al. 2018) metrics, where ↑ indicates that higher is better and ↓ indicates the opposite.

Implementation. Our method is implemented by Pytorch and all experiments are conducted on 8 RTX 3090Ti GPUs. For each scene, we optimize the coarse global model for 60,000 iterations and fine-tune each block for another 60,000 iterations. Specifically, we use SAM2-L to extract semantic features. For each anchor, the feature dimensionality d is set to 32, and is used to construct $K = 10$ Gaussians. The output dimensionality D of the semantic mapping module matches that of the extracted features. The weighting coefficient λ in the photometric loss is fixed to 0.2, while the other loss terms are weighted by $\lambda_1 = 1.0$, $\lambda_2 = 100.0$, and $\lambda_3 = 0.015$. During scene division, the threshold of the importance score ratio ε is set to 0.1. For SmallCity, the entire scene is divided into 4×4 blocks, whereas the other scenes are divided into 2×4 blocks.

Comparative Evaluation

We compare the proposed CAG-GS with five state-of-the-art methods, including VastGaussian (Lin et al. 2024), CityGaussian (Liu et al. 2024b), and DOGS (Chen and Lee 2024) based on explicit scene representations, as well as Octree-GS (Ren et al. 2024) and Momentum-GS (Fan et al. 2024a) based on implicit ones. Among these methods, Octree-GS represents the entire scene with a single model, without employing the divide-and-conquer strategy.

Rendering Quality. We evaluate the rendering quality of CAG-GS on the five scenes in comparison to the five afore-

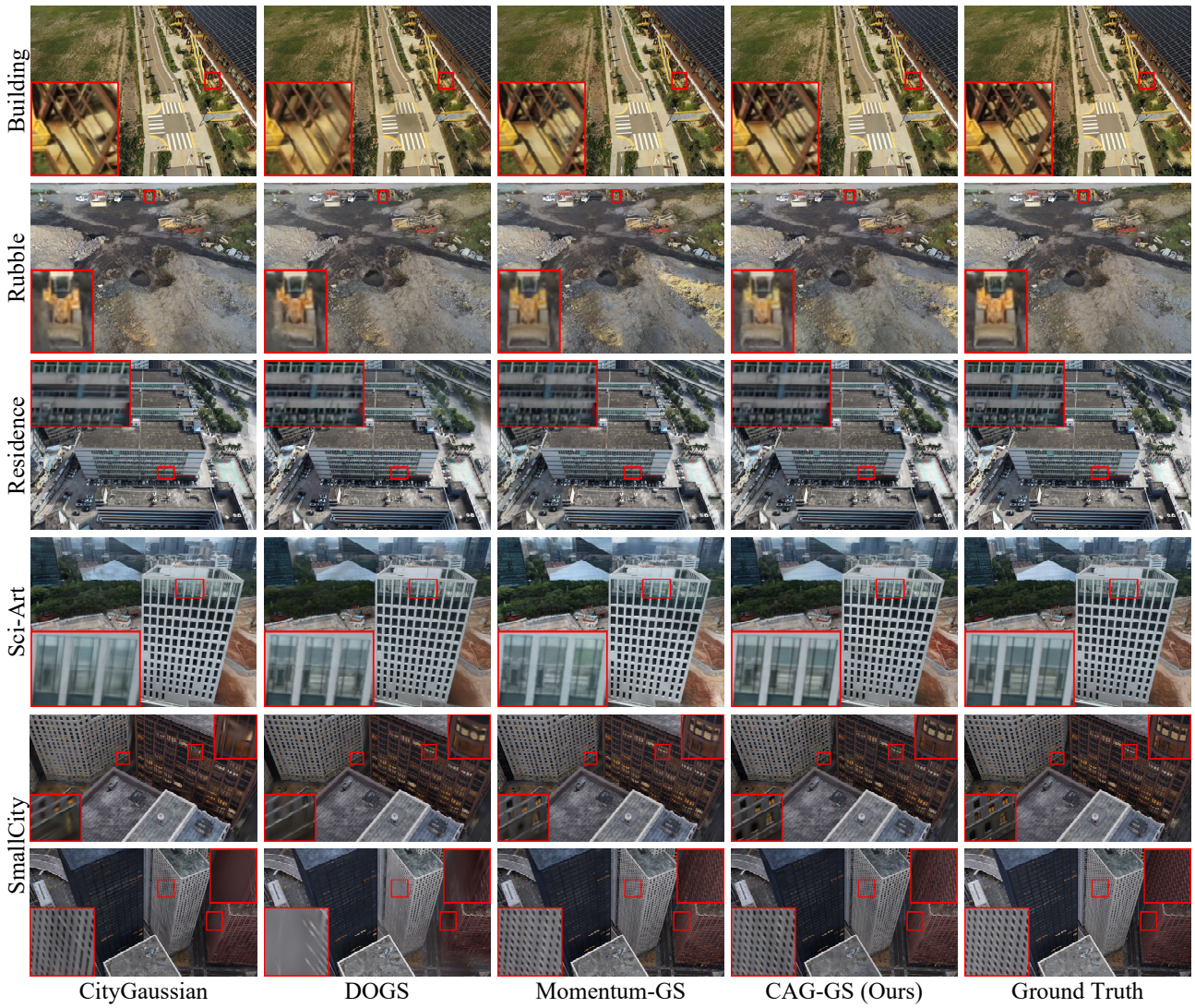


Figure 3: Qualitative comparison on the five large-scale scenes. The results of the comparative methods are rendered using their publicly released checkpoints.

mentioned methods. The corresponding results are reported in Table 1. As seen from the table, our method outperforms the comparative methods in most cases, demonstrating its effectiveness. In particular, it achieves significantly better SSIM and LPIPS scores than the methods based on explicit representations. It is worth noting that Momentum-GS shows competitive or even better performance on certain metrics, mainly owing to its use of a much larger number of anchors. Several rendering results of CAG-GS as well as three comparative methods (CityGaussian, DOGS, and Momentum-GS) are given in Figure 3. As shown in the figure, the comparative methods often struggle with thin structures, resulting in noticeable blurring and artifacts. In contrast, our method better preserves fine-grained and intricate details, thus producing clearer and more faithful renderings.

Rendering Efficiency. We report the rendering speed and peak GPU memory usage of CAG-GS on the *Building*, *Rubble*, *Residence*, and *Sci-Art* scenes, alongside those of CityGaussian, Octree-GS, and Momentum-GS for comparison. The corresponding results are presented in Table 2. As seen from the table, our method and Octree-GS generally require less GPU memory than CityGaussian, yet their rendering performance is slower due to the additional overhead of Gaussian construction. Our method uses a similar number of anchors as Octree-GS, but its sequential decoding of Gaussian attributes from anchor features across blocks leads to slightly lower peak memory usage and slower rendering speed. Meanwhile, Momentum-GS employs a large number of anchors, resulting in reduced memory efficiency and slower rendering compared to our method.

Ablation Study

This subsection verifies the effectiveness of each key element in CAG-GS by conducting ablation studies on the

Ground Truth w/o Semantic Guidance Full

Figure 4: Visualization of the effect of semantic guidance in CAG-GS. Anchor feature maps and rendered RGB images of CAG-GS with and without semantic guidance are shown, along with the ground truth RGB image. The current view covers two blocks, and the color shift in the first feature map marks their boundary.

Scene	*Building*		*Rubble*		*Residence*		*Sci-Art*	
Metric	FPS	Mem.	FPS	Mem.	FPS	Mem.	FPS	Mem.
CityGaussian	36	9.04	57	5.47	45	6.21	85	2.60
Octree-GS	41	3.91	38	4.61	28	6.08	42	4.86
Momentum-GS	21	8.02	24	4.83	17	6.60	16	8.08
CAG-GS (Ours)	28	3.38	27	3.16	26	3.38	31	3.13

Table 2: Comparison of rendering speed and peak GPU memory usage (GB). Octree-GS is reproduced and tested following its official implementation. The results of CityGaussian and Momentum-GS are obtained using their publicly released checkpoints.

Scene	*Building*			*Residence*		
Metric	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
w/o semantic	22.68	0.788	0.229	21.86	0.792	0.206
w/o geometric	23.18	0.812	0.208	22.32	0.808	**0.201**
from scratch	22.89	0.802	0.217	21.95	0.796	0.213
Ours	**23.46**	**0.814**	**0.198**	**22.62**	**0.820**	0.202

Table 3: Ablation study on the coarse-to-fine training strategy. The best results are highlighted in **bold**.

Building and *Residence* scenes.

Training Strategy. We first analyze the impact of the semantic guidance and geometric constraints by respectively taking out the corresponding loss terms, and then verify the effectiveness of coarse model training by training the local models from scratch without initialization. The corresponding results are reported in Table 3. As seen from the table, the performance of CAG-GS drops when the semantic guidance or the geometric constraints are removed. In addition, training the local models from scratch also results in inferior performance, demonstrating the effectiveness of using the coarse model for initialization. To intuitively demonstrate the effectiveness of the semantic guidance, we provide an example of the rendering results in Figure 4. It can be seen that, without the semantic guidance, the learned anchor features show abrupt transitions at the block boundary, along with visually irregular color variations on the building facade crossing the boundary. In contrast, the semantic guidance improves feature consistency and leads to better visual quality in the corresponding region. The above results

Scene	*Building*			*Residence*		
Metric	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DINOv2-L	23.03	0.805	0.211	22.35	0.812	0.209
DAnyV2-L	22.82	0.793	0.223	22.16	0.796	0.227
SAM2-S	23.26	0.810	0.212	22.44	0.818	0.203
SAM2-B	23.29	0.814	0.206	22.47	**0.822**	0.202
SAM2-L	**23.46**	**0.814**	**0.198**	**22.62**	0.820	**0.202**

Table 4: Ablation study on foundation models and SAM2 sizes for semantic guidance. The best results are highlighted in **bold**.

demonstrate that our training strategy is helpful for CAG-GS to produce higher-quality renderings for large-scale scenes.

Foundation Model. We first evaluate the effectiveness of SAM2 by replacing it with DINOv2 (Oquab et al. 2024) and DepthAnythingV2 (Yang et al. 2024), and then analyze the impact of SAM2 with different model sizes. The corresponding results are reported in Table 4. As seen from the table, among the foundation models evaluated, SAM2 achieves the best performance for semantic guidance in most cases. On the other hand, despite being specifically designed for geometry-related tasks, DepthAnythingV2 performs relatively worse in our setting. This is possibly because the geometric priors it provides mainly focus on depth information, which may lead to suboptimal guidance for constructing consistent Gaussians. Furthermore, it can be seen from the table that SAM2 with larger capacity generally yields better rendering quality. Please see the supplemental material for more experimental results and discussions.

5 Conclusion

In this paper, we propose CAG-GS, a semantic-guided anchor representation for large-scale scene rendering, while a coarse-to-fine strategy is explored for training. The proposed method integrates semantic features from the pre-trained segmentation model SAM2 into the implicit Gaussian construction via a semantic mapping module, enhancing global consistency across different blocks under the divide-and-conquer reconstruction strategy. The experimental results on the five scenes demonstrate that the introduced semantic guidance and the explored training strategy enable the proposed CAG-GS to produce high-fidelity and visually consistent renderings for large-scale scenes.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62376269, 62472262, 61991423, U1805264), Taishan Scholarship Construction Engineering.

References

Agarwal, S.; Furukawa, Y.; Snavely, N.; Simon, I.; Curless, B.; Seitz, S. M.; and Szeliski, R. 2011. Building Rome in a Day. *Commun. ACM*, 54(10): 105–112.

Bleyer, M.; Rhemann, C.; and Rother, C. 2011. PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In *British Machine Vision Conference*, 1–11.

Chen, D.; Li, H.; Ye, W.; Wang, Y.; Xie, W.; Zhai, S.; Wang, N.; Liu, H.; Bao, H.; and Zhang, G. 2024a. PGSR: Planar-based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction. *IEEE Trans. Vis. Comput. Graph.*

Chen, J.; Ye, W.; Wang, Y.; Chen, D.; Huang, D.; Ouyang, W.; Zhang, G.; Qiao, Y.; and He, T. 2024b. GigaGS: Scaling up Planar-Based 3D Gaussians for Large Scene Surface Reconstruction. arXiv:2409.06685.

Chen, Y.; and Lee, G. H. 2024. DOGS: Distributed-Oriented Gaussian Splatting for Large-Scale 3D Reconstruction Via Gaussian Consensus. In *Advances in Neural Information Processing Systems*, volume 37, 34487–34512.

Chen, Y.; Wu, Q.; Lin, W.; Harandi, M.; and Cai, J. 2024c. HAC: Hash-Grid Assisted Context for 3D Gaussian Splatting Compression. In *European Conference on Computer Vision*, volume 15065, 422–438.

Fan, J.; Li, W.; Han, Y.; and Tang, Y. 2024a. MomentumGS: Momentum Gaussian Self-Distillation for High-Quality Large Scene Reconstruction. arXiv:2412.04887.

Fan, L.; Yang, Y.; Li, M.; Li, H.; and Zhang, Z. 2024b. Trim 3D Gaussian Splatting for Accurate Geometry Representation. arXiv:2406.07499.

Fan, Z.; Wang, K.; Wen, K.; Zhu, Z.; Xu, D.; and Wang, Z. 2024c. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. In *Advances in Neural Information Processing Systems*, volume 37, 140138–140158.

Hanson, A.; Tu, A.; Lin, G.; Singla, V.; Zwicker, M.; and Goldstein, T. 2025. Speedy-Splat: Fast 3D Gaussian Splatting with Sparse Pixels and Sparse Primitives. In *IEEE Conference on Computer Vision and Pattern Recognition*, 21537–21546.

Huang, B.; Yu, Z.; Chen, A.; Geiger, A.; and Gao, S. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*, 1–11.

Huang, H.; Wu, Y.; Deng, C.; Gao, G.; Gu, M.; and Liu, Y. 2025. FatesGS: Fast and Accurate Sparse-View Surface Reconstruction Using Gaussian Splatting with Depth-Feature Consistency. In *AAAI Conference on Artificial Intelligence*, volume 39, 3644–3652.

Ji, Y.; Zhu, H.; Tang, J.; Liu, W.; Zhang, Z.; Tan, X.; and Xie, Y. 2025. FastLGS: Speeding Up Language Embedded Gaussians with Feature Grid Mapping. In *AAAI Conference on Artificial Intelligence*, volume 39, 3922–3930.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4): 1–14.

Kerbl, B.; Meuleman, A.; Kopanas, G.; Wimmer, M.; Lanvin, A.; and Drettakis, G. 2024. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM Trans. Graph.*, 43(4): 1–15.

Lafarge, F.; Keriven, R.; Brédif, M.; and Vu, H. 2013. A Hybrid Multiview Stereo Algorithm for Modeling Urban Scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1): 5–17.

Li, Y.; Jiang, L.; Xu, L.; Xiangli, Y.; Wang, Z.; Lin, D.; and Dai, B. 2023. MatrixCity: A Large-scale City Dataset for City-scale Neural Rendering and Beyond. In *IEEE International Conference on Computer Vision*, 3205–3215.

Li, Y.; Lyu, C.; Di, Y.; Zhai, G.; Lee, G. H.; and Tombari, F. 2024. GeoGaussian: Geometry-Aware Gaussian Splatting for Scene Rendering. In *European Conference on Computer Vision*, volume 15093, 441–457.

Liang, Z.; Zhang, Q.; Hu, W.; Zhu, L.; Feng, Y.; and Jia, K. 2024. Analytic-Splatting: Anti-Aliased 3D Gaussian Splatting via Analytic Integration. In *European Conference on Computer Vision*, volume 15075, 281–297.

Lin, J.; Li, Z.; Tang, X.; Liu, J.; Liu, S.; Liu, J.; Lu, Y.; Wu, X.; Xu, S.; Yan, Y.; and Yang, W. 2024. VastGaussian: Vast 3D Gaussians for Large Scene Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 5166–5175.

Lin, L.; Liu, Y.; Hu, Y.; Yan, X.; Xie, K.; and Huang, H. 2022. Capturing, Reconstructing, and Simulating: The UrbanScene3D Dataset. In *European Conference on Computer Vision*, volume 13668, 93–109.

Liu, X.; Wu, X.; Zhang, P.; Wang, S.; Li, Z.; and Kwong, S. 2024a. CompGS: Efficient 3D Scene Representation via Compressed Gaussian Splatting. In *ACM International Conference on Multimedia*, 2936–2944.

Liu, Y.; and Dong, Q. 2025. EquiPose: Exploiting Permutation Equivariance for Relative Camera Pose Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1127–1137.

Liu, Y.; Luo, C.; Fan, L.; Wang, N.; Peng, J.; and Zhang, Z. 2024b. CityGaussian: Real-Time High-Quality Large-Scale Scene Rendering with Gaussians. In *European Conference on Computer Vision*, volume 15074, 265–282.

Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; and Dai, B. 2024. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. In *IEEE Conference on Computer Vision and Pattern Recognition*, 20654–20664.

Mallick, S. S.; Goel, R.; Kerbl, B.; Steinberger, M.; Carrasco, F. V.; and la Torre, F. D. 2024. Taming 3DGS: High-Quality Radiance Fields with Limited Resources. In *SIGGRAPH Asia 2024 Conference Papers*, 1–11.

Mi, Z.; and Xu, D. 2023. Switch-NeRF: Learning Scene Decomposition with Mixture of Experts for Large-scale Neural

Radiance Fields. In *International Conference on Learning Representations*.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.*, 41(4): 1–15.

Navaneet, K. L.; Meibodi, K. P.; Koohpayegani, S. A.; and Pirsiavash, H. 2023. Compact3D: Compressing Gaussian Splat Radiance Field Models with Vector Quantization. arXiv:2311.18159.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H. V.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; Assran, M.; Ballas, N.; Galuba, W.; Howes, R.; Huang, P.; Li, S.; Misra, I.; Rabbat, M.; Sharma, V.; Synnaeve, G.; Xu, H.; Jégou, H.; Mairal, J.; Labatut, P.; Joulin, A.; and Bojanowski, P. 2024. DINOv2: Learning Robust Visual Features without Supervision. *Trans. Mach. Learn. Res.*

Qin, M.; Li, W.; Zhou, J.; Wang, H.; and Pfister, H. 2024. LangSplat: 3D Language Gaussian Splatting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 20051–20060.

Ravi, N.; Gabeur, V.; Hu, Y.; Hu, R.; Ryali, C.; Ma, T.; Khedr, H.; Rädle, R.; Rolland, C.; Gustafson, L.; Mintun, E.; Pan, J.; Alwala, K. V.; Carion, N.; Wu, C.; Girshick, R. B.; Dollár, P.; and Feichtenhofer, C. 2025. SAM 2: Segment Anything in Images and Videos. In *International Conference on Learning Representations*.

Ren, K.; Jiang, L.; Lu, T.; Yu, M.; Xu, L.; Ni, Z.; and Dai, B. 2024. Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians. arXiv:2403.17898.

Schönberger, J. L.; and Frahm, J. 2016. Structure-from-Motion Revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, 4104–4113.

Snavely, N.; Seitz, S. M.; and Szeliski, R. 2008. Modeling the World from Internet Photo Collections. *Int. J. Comput. Vis.*, 80(2): 189–210.

Tancik, M.; Casser, V.; Yan, X.; Pradhan, S.; Mildenhall, B. P.; Srinivasan, P. P.; Barron, J. T.; and Kretzschmar, H. 2022. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 8238–8248.

Turki, H.; Ramanan, D.; and Satyanarayanan, M. 2022. Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly- Throughs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 12912–12921.

Wang, Y.; Li, Z.; Guo, L.; Yang, W.; Kot, A. C.; and Wen, B. 2024. ContextGS : Compact 3D Gaussian Splatting with Anchor Level Context Model. In *Advances in Neural Information Processing Systems*, volume 37, 51532–51551.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.*, 13(4): 600–612.

Wu, Y.; Qi, Z.; Shi, Z.; and Zou, Z. 2025. Block-Gaussian: Efficient Large-Scale Scene Novel View Synthesis via Adaptive Block-Based Gaussian Splatting. arXiv:2504.09048.

Yan, Z.; Low, W. F.; Chen, Y.; and Lee, G. H. 2024. Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering. In *IEEE Conference on Computer Vision and Pattern Recognition*, 20923–20931.

Yang, L.; Kang, B.; Huang, Z.; Zhao, Z.; Xu, X.; Feng, J.; and Zhao, H. 2024. Depth Anything V2. In *Advances in Neural Information Processing Systems*, volume 37, 21875–21911.

Yu, Z.; Chen, A.; Huang, B.; Sattler, T.; and Geiger, A. 2024. Mip-Splatting: Alias-Free 3D Gaussian Splatting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 19447–19456.

Zeng, Z.; Wang, Y.; Ju, L.; and Guan, T. 2025. Frequency-Aware Density Control via Reparameterization for High-Quality Rendering of 3D Gaussian Splatting. In *AAAI Conference on Artificial Intelligence*, volume 39, 9833–9841.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *IEEE Conference on Computer Vision and Pattern Recognition*, 586–595.

Zhou, S.; Chang, H.; Jiang, S.; Fan, Z.; Zhu, Z.; Xu, D.; Chari, P.; You, S.; Wang, Z.; and Kadambi, A. 2024. Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, 21676–21685.

Zwicker, M.; Pfister, H.; van Baar, J.; and Gross, M. H. 2001. EWA Volume Splatting. In *Proceedings Visualization*, 29–538.