2025 IEEE International Conference on Robotics and Automation (ICRA)
May 19-23, 2025. Atlanta, USA

Hash-GS: Anchor-Based 3D Gaussian Splatting with Multi-Resolution Hash Encoding for Efficient Scene Reconstruction

Yijia Xie[1], Yuhang Lin[1], Laijian Li[1], Lina Liu[2], Xiaobin Wei[3], Yong Liu[1,*], Jiajun Lv[1,*]

***Abstract*— Realistic 3D object and scene reconstruction is pivotal in advancing fields such as world model simulation and embodied intelligence. In this paper, we introduce Hash-GS, a storage-efficient method for large-scale scene reconstruction using anchor-based 3D Gaussian Splatting (3DGS). The vanilla 3DGS struggles with high memory demands due to the large number of primitives, especially in complex or extensive scenes. Hash-GS addresses these challenges with a compact representation by leveraging high-dimensional features to parameterize primitive properties, stored in compact hash tables, which reduces memory usage while preserving rendering quality. It also incorporates adaptive anchor management to efficiently control the number of anchors and neural Gaussians. Additionally, we introduce an analytic 3D smoothing filter to mitigate aliasing and support Level-of-Detail for optimized rendering across varying intrinsic parameters. Experimental results on several datasets demonstrate that Hash-GS improves storage efficiency while maintaining competitive rendering performance, especially in large-scale scenes.**

I. Introduction

Scene reconstruction is an ill posed and long-standing problem in computer vision and computer graphics field. The emergence of implicit Neural Radiance Fields (NeRF) [1] has brought 3D reconstruction into a new stage, achieving photo-realistic scene reconstruction. The recent 3D Gaussian Splatting (3DGS) representation [2] which is developed from explicit point-based rendering [3, 4] have made remarkable progress in rendering quality and speed. Nowadays, real-time high-fidelity novel view rendering has been widely demanded by applications such as virtual reality (VR) interactions [5, 6], SLAM [7–11], autonomous driving [12, 13] and large-scale mapping [14–18].

Although 3DGS has significantly advanced real-time rendering, its explicit representation and heuristic densification strategy impose substantial memory requirements. Scaffold-GS [19] utilizes anchors with features to predict local neural Gaussians to describe local properties, reducing redundant Gaussians. However, features are optimized independently without considering mutual information between spatially adjacent anchors. Other well-established works introduce additional compression techniques to encode and quantize Gaussian attributes to reduce storage overhead [20–22].

In large-scale scenes or distant viewing perspectives, the number of Gaussians within the view frustum increases dramatically, seriously affecting computing efficiency. In such scenarios, many details are inherently unobservable or unnecessary to capture. Some works [15, 16, 23, 24] introduce Level-of-Detail (LOD) [25] to strike a balance between visual quality and computation burden. However, these methods exhibit inefficient storage efficiency and some of them lack adequate anti-aliasing.

In this paper, following [15, 19] we present *Hash-GS*, an aliasing-free method that exploits compact feature-parameterized Gaussian attributes with LOD support. *Hash-GS* integrates the analytic 3D filter, and utilizes size-constrained hash tables to manage features with an improved anchor management mechanism. Our contributions can be briefly summarized as follows:

- We introduce an anchor-based 3D Gaussian representation that leverages hash tables [26] which aggregate contextual information to facilitate efficient memory storage. We further support LOD and introduce a LOD bias to refine LOD anchor-level selection, enhancing both storage efficiency and rendering performance.
- We propose an adaptive anchor management strategy that utilizes block-wise gradient analysis to efficiently control the addition and pruning of anchor points and neural Gaussians, fully utilizing spatial features while preventing excessive anchor growth.
- To address the aliasing issue in large-scale scene rendering with varying focal lengths, we reanalyze the filter structure [27] and introduce an analytic 3D smoothing filter, effectively mitigating aliasing artifacts.

II. Related Work

A. Neural Scene Representation and Reconstruction

Traditional scene reconstruction techniques rely on explicit primitives like meshes, surface [28] or point clouds [29], NeRF [1, 30, 31] introduced an implicit representation using multi-layer perceptrons (MLPs), leading to various parameterization methods aimed at improving efficiency and accuracy. Grid-assisted methods have been widely explored [26, 32–36] and integrated into localization systems [7, 8]. For instance, PlenOctree [35] encodes spherical harmonics (SH) and density into an octree structure, while iNGP [26] leverages shallow MLPs with multi-resolution grids to enhance parameter efficiency. Beyond hybrid grid-based fashion, triplane [9, 37] represents scenes with three orthogonal planes, exploiting 3D content sparsity, and low-rank decomposition [38, 39] reduces the dimensionality of voxel grids using tensor decomposition, producing more compact models.

* Corresponding authors. Email: yongliu@iipc.zju.edu.cn.
[1] Institute of Cyber-Systems and Control, Zhejiang University, China.
[2] China Mobile Research Institute, Beijing, China.
[3] WASU Media & Network Co.Ltd.
This work was supported by AI-based All-media Service Platform, Major Scientific and Technological Innovation Project of Hangzhou under Grant 2022AIZD0019.

2025 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3315-4139-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICRA55743.2025.11128324

Recent advances in differentiable point-based rendering [2–4, 16] have further expanded the field. 3D Gaussian Splatting [2] uses anisotropic 3D Gaussians as primitives that are projected onto a 2D plane and rasterized by α-blending. To reduce redundant Gaussian primitives and improve robustness to novel viewpoints, recent methods[15, 19] incorporate local feature encoding, enabling a hybrid representation. HAC[40] transfers local features into the hash grid[26] and further compresses the hash grid through CNC[41].

B. Level-of-Detail

Level-of-Detail [25] is a well-established technique to balance rendering speed and visual quality, commonly based on mip-maps [42]. This concept has been naturally extended to NeRF-based reconstruction [30, 31, 36], applied in multi-scale large scene reconstruction [14, 43] and multi-granularity Signed Distance Function (SDF) learning [44]. VR-NeRF [5] leverages mip-mapping properties of multi-resolution grids for continuous LOD transitions, while others [45, 46] combine prefiltering with LOD achieving physically-based rendering under neural representation.

LOD techniques have quickly been adopted in 3DGS. Octree-GS [15] selects Gaussian particles based on view distance, while Hierarchical-GS [16] adopts a hierarchical tree structure, where each node represents a Gaussian, progressing from the bottom up. The optimal tree cut is then determined to satisfy pixel-level granularity from any perspectives. [23] applies varying degrees of compression to the trained model to capture varying level details. [24] constructs a hierarchy model using LiDAR, allowing adaptation to lightweight devices.

C. Anti-Aliasing

When the focal length changes or sampling distant objects, Nyquist sampling [27] may not be met, leading to aliasing. There are two primary strategies to address this issue: multi-sampling and pre-filtering (area sampling). Early NeRF-based approaches by [30, 31] introduced cone casting with positional encoding for area sampling. Tri-miprf [37] develops sphere sampling based on multi level tri-plane representation. Zip-nerf [36] performs super sampling within the truncated frustum of the viewing cone.

In 3DGS, a large number of small Gaussians representing rich details are likely to cause aliasing. [47] categorizes the Gaussians into multiple groups and selects the appropriate size of Gaussians for rendering in accordance with resolution. Mip-splatting [27] applies the 3D Gaussian low-pass filter and 2D box filter to achieve aliasing-free rendering. [48] treats the pixels covered by Gaussians as integration domains, calculating opacities with an approximation function rather than using pixel centers.

III. Preliminaries

A. 3D Gaussian Splatting

Starting from SfM [49] point cloud and a set of posed images, 3DGS utilizes explicit 3D Gaussian distributions to represent scenes. Each anisotropic primitive is parameterized by spherical harmonics (SHs), an opacity $o \in \mathbb{R}$, a center point $\mu \in \mathbb{R}^3$ and a covariance matrix $\mathbf{\Sigma}$. For any 3D point in space, the Gaussian distribution is defined as:

$$G(x) = e^{-\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu)} \tag{1}$$

The covariance matrix $\mathbf{\Sigma}$ is further factorized into the product of a scaling matrix $\mathbf{S} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$, yielding the form $\mathbf{RSS}^T\mathbf{R}^T$ which can be regarded as an anisotropic elliptical kernel. During rendering, the 3D primitives are projected onto the 2D pixel plane. To address the non-linear and non-affine nature of perspective transformation, a local affine approximation $\mathbf{J}$, derived from the first-order Taylor expansion, is applied to project $\mathbf{\Sigma}$ [50] using the world-to-camera transformation matrix $\mathbf{W}$:

$$\mathbf{\Sigma}' = \mathbf{JW\Sigma W}^T\mathbf{J}^T \tag{2}$$

$\mathbf{\Sigma}'$ is the approximation of projecting an ellipse $\mathbf{\Sigma}$ from 3D space onto the 2D plane. For any pixel p, the color $C(p)$ can be efficiently computed using α-blending:

$$C(p) = \sum_{i=1}^{n} c_i o_i G_i^{2D}(p) \Pi_{j=1}^{i-1} \left(1 - o_j G_j^{2D}(p)\right) \tag{3}$$

$$G^{2D}(p) = e^{-\frac{1}{2}(p-\mu')^T \mathbf{\Sigma}'(p-\mu')} \tag{4}$$

where μ' is projection of the 3D center μ onto the 2D image plane, and c_i is color derived from SHs.

B. Anchor-based Gaussian Splatting

Vanilla 3DGS often leads to redundancy 3D Gaussians, some methods [15, 19] address this by using anchor points with high-dimensional features to guide Gaussian arrangement, with attributes dynamically decoded through MLPs, optimizing memory efficiency. Specifically, given the coarsest voxel size ϵ, the SfM points $\mathbf{P}$ are voxelized into γ levels:

$$\mathbf{V} = \{\lfloor \frac{\mathbf{P}}{\epsilon} \rceil \cdot \epsilon, ..., \lfloor \frac{\mathbf{P}}{\epsilon/s^{\gamma-1}} \rceil \cdot \epsilon/s^{\gamma-1}\}. \tag{5}$$

where $\lfloor \rceil$ means rounding, s is the scale ratio between two adjacent levels. Each non empty voxel is assigned an anchor point, and aside from the properties described in Section. III-A (excluding SHs), each anchor i is associated with a fix number k_i of position offsets O_i^j, a scaling factor $\beta_i \in \mathbb{R}^3$ and a feature f_i. Neural Gaussians which actually participate in rendering are derived from anchor attributes. The positions of neural Gaussians generated by anchor i are calculated as:

$$\mu_i^j = \mu_i + \beta_i O_i^j, \quad j \in [0, k_i]. \tag{6}$$

The anchor imposes an upper limit on the scale of the neural Gaussians:

$$S_i^j = S_i \cdot sigmoid(\Psi_S(f_i, \mathbf{v}_i, d_i)) \tag{7}$$

where $\mathbf{v}_i$ and d_i is the direction vector and the distance between the viewpoint and the anchor respecitively. The remaining attributes of the k_i neural Gaussians are decoded from tiny MLPs. Only neural Gaussians with opacity greater than 0 contribute to the final rendering. During optimization, the positions, rotations and opacities of all anchors remain fixed. This compact representation reduces redundant Gaussians and enhances robustness for novel view rendering.

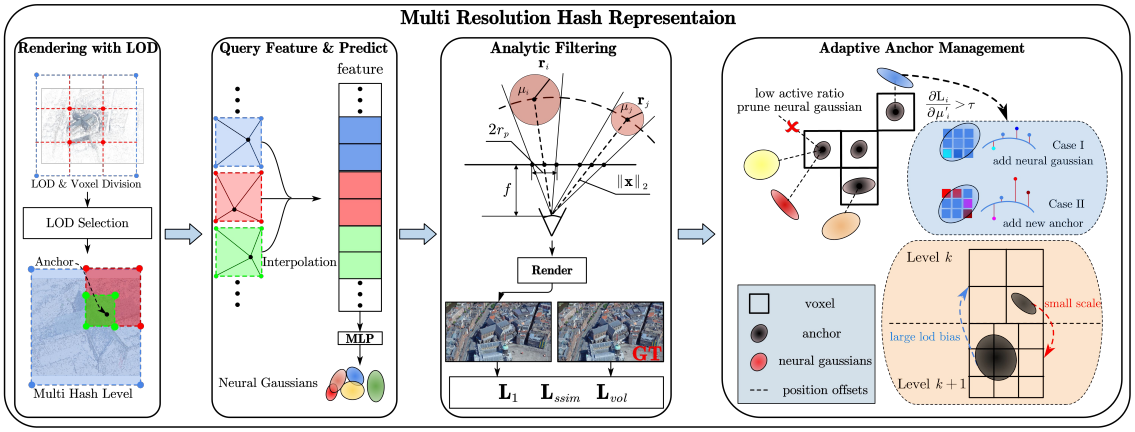

Fig. 1: The pipeline of the proposed method with four main modules. Our system design is closely focused on addressing the high storage costs, low rendering speed and zooming rendering requirements in large-scale scene reconstruction.

IV. Methodology

The overview of the proposed method is shown in Fig.1. Our goal is to maximize parameter utilization to minimize storage overhead, while fulfilling zooming requirements and balancing rendering quality with computational efficiency for large-scale reconstruction.

A. Multi Resolution Hash Representation

Inspired by iNGP [26] that the natural scenes are sparsity and exhibit smoothness on certain attributes, we transfer the high-dimension features associated with anchors into multi-resolution 3D scene hash encoding to further enhance memory efficiency. In contrast to HAC[40], which utilizes the hash grid for compression purposes, our approach employs it to reduce the redundancy of anchor features introduced by the multi-level voxel resolution initialization in [15]. Given a base resolution Φ and a scale ratio s between two adjacent levels, space is divided into L levels. Features at resolution level l are stored in a hash table T^l with a fixed-size N. The query index $h^l(\mu_i)$ of anchor (x_i, y_i, z_i) at level l is computed using three large primes through bitwise XOR ($\oplus$):

$$h^l(\mu_i) = (x_i\pi_1 \oplus y_i\pi_2 \oplus z_i\pi_3) \mod N\,. \tag{8}$$

The result $f_i^l = T^l(h^l(\mu_i))$ from each level is concatenated to form the complete feature $f_i = concat(f_i^l)$.

Since MLPs tend to learn low-frequency signals [1] and hash collisions are common, using the direct output of MLP as the color for neural Gaussians [19] can result in color discrepancies. To address this, we additionally assign a base color property c_i to each anchor, while allowing the MLP output to adjust the base color:

$$c_i^j = c_i + \tanh(\Psi_c(f_i, \mathbf{v_i}, d_i))\,. \tag{9}$$

B. Rendering with LOD Structure

Anchor LOD Level Initialization: Following the approach outlined in [15], we first determine the maximum level γ across the scene, using the distance between all training perspectives and anchors:

$$\gamma = \lfloor \log_s(d_{max}/d_{min}) \rceil + 1\,, \tag{10}$$

where d_{max} and d_{min} represent 0.99 and 0.01 quantiles of the distance distribution.

We begin voxelization at the finest resolution and progressively increase the voxel size. To ensure distinct separation between LOD levels, a level γ_l is validated only when $N_{\gamma_l}/N_{\gamma_{l+1}}$ less than the threshold τ_N, where N_{γ_l} denotes the number of anchor at level γ_l. Otherwise we will multiply the scale ratio between l and $l+1$. By assigning each anchor an integer attribute $\mathbf{L}_i$ corresponding to its LOD level, anchor selection is streamlined, enabling efficient and high-quality rendering from new viewpoints.

Anchor Selection with LOD Level: In large-scale scene rendering, anchor selection based solely on LOD levels can introduce discontinuities during viewpoint changes. Rounding-based anchor selection may lead to abrupt transitions between LOD levels, which can result in visual artifacts or inefficient rendering.

To address this, we implement an adaptive anchor selection strategy. First, we select anchors within the frustum based on the distance between the viewpoint and the anchor following [15]. This ensures that primitives with appropriate granularity are chosen:

$$\mathbf{U}_d = \mathbf{L}_i \leq \lfloor \min(\max(\log_s(d_{max}/d), 0), \gamma - 1) \rceil\,. \tag{11}$$

Besides, inspired by [5, 15], we introduce an LOD bias θ_i. This helps smooth transitions between adjacent LOD levels by selecting additional anchors from neighboring levels:

$$\mathbf{U}_b = \mathbf{L}_i \leq \log_s(d_{max}/d) + \theta_i\,. \tag{12}$$

By combining anchors from both $\mathbf{U}_d$ and $\mathbf{U}_b$, our approach ensures continuous and smooth LOD transitions during viewpoint changes. For anchors in $\mathbf{U}_b$, we adjust neural Gaussian opacities using:

$$(o_i^j)' = o_i^j \cdot \theta_i \cdot c_\theta, \quad i \in \mathbf{U}_b, j \in [0, k_i]\,. \tag{13}$$

Depending on the rounding method, we assign different initial values to θ_i and c_θ. Unlike previous method[15] that periodically adjust θ_i via gradients, our approach allows direct optimization, resulting in more efficient rendering and enhanced visual consistency.

C. Adaptive Anchor Management

The position offsets O_i^j of neural Gaussians are explicitly optimized and stored. However, as illustrated in [19], neural Gaussians are often underutilized, resulting in additional overhead in storage. To address this inefficiency, we propose a block-wise gradient analysis method that enables the gradual adjustment of the number of neural Gaussian k_i for each anchor. By analyzing the block-wise gradient, we can determine whether to increase k_i or introduce new anchors.

Gradient Analysis: In vanilla 3DGS [2], the decision to densify a primitive is based on the average value of $\frac{\partial L^m}{\partial (\mu_i')^m}$ over I iterations:

$$\frac{\partial L}{\partial \mu_i'} = \frac{\sum_{m=1}^{I} ||\frac{\partial L^m}{\partial (\mu_i')^m}||}{I} = \frac{\sum_{m=1}^{I} \sqrt{(\frac{\partial L^m}{\partial ((\mu_i')_u^m})^2 + (\frac{\partial L^m}{\partial ((\mu_i')_v^m})^2}}{I} \tag{14}$$

where $(\mu_i')^m = ((\mu_i')_u^m, (\mu_i')_v^m)$ is the primitive center on image plane from viewpoint m, denote $g_{i,u}^m = \frac{\partial L^m}{\partial ((\mu_i')_u^m} = \sum_{p=1}^{P}\sum_{n=1}^{3} \frac{\partial L_p^m}{\partial c_n^p} \cdot \frac{\partial c_n^p}{\partial \alpha_i} \cdot \frac{\partial \alpha_i}{\partial (\mu_i')_u^m}$. Once $\frac{\partial L}{\partial \mu_i'} > \tau$, the Gaussian i is selected for densification.

The gradient terms may cancel each other out due to the uncertainty of their signs in terms of large Gaussian kernel. This leads to insufficient capture of details. To this end, AbsGS [51] calculates the absolute gradient values. For all P pixels covered by Gaussians i, the absolute gradient of $(g_{i,u}^m)_{abs} = (\frac{\partial L^m}{\partial (\mu_i')_u^m})_{abs}$ is computed as :

$$(g_{i,u}^m)_{abs} = \sum_{p=1}^{P}\sum_{n=1}^{3} \| \frac{\partial L_p^m}{\partial c_n^p} \cdot \frac{\partial c_n^p}{\partial \alpha_i} \cdot \frac{\partial \alpha_i}{\partial (\mu_i')_u^m} \| . \tag{15}$$

However, we find that Eq. (15) obscures the true meaning of the gradient, leading to unnecessary densifications as illustrated in Fig. 2. Suppose a Gaussian covers darker pixels, the optimal choice for Gaussian should be to deepen its color. Without the absolute value, i.e., $g_{i,u}^m$ would approach to zero due to the symmetry of the third term in Eq. (15) while $(g_{i,u}^m)_{abs}$ may be quite large which may guide anchor to densify. To mitigate this, we redefine the absolute gradient:

$$(g_{i,u}^m)_{abs} = \sum_{p=1}^{P}\sum_{n=1}^{3} \| \frac{\partial L_p^m}{\partial c_n^p} \| \cdot \frac{\partial c_n^p}{\partial \alpha_i} \cdot \frac{\partial \alpha_i}{\partial (\mu_i')_u^m} . \tag{16}$$

Anchor Operations: As mentioned in IV-A, we aim for the neural Gaussian's color to fluctuate slightly around the anchor's color. Therefore, during densification, if the gradient magnitude is moderately high, i.e., $(\frac{\partial L}{\partial \mu_i'})_{abs} \in [\tau_1, \tau_2]$, we increase the number of neural Gaussians, k_i. If the gradient is significantly large, i.e., $((\frac{\partial L}{\partial \mu_i'})_{abs} > \tau_2) \cup (\frac{\partial L}{\partial \mu_i'} > \tau)$, indicating a poor alignment between the neural Gaussian's color and the anchor's color, we generate new anchors instead. For a given anchor i, k_i is reduced if the ratio of neural Gaussian activations to the anchor's visible occurrences falls below τ_d. We record the opacities of activated neural Gaussians for each anchor. If accumulated opacities are under τ_α or $k_i < 1$, anchors are subsequently removed. In practical implementation, we allow MLPs to predict the attributes of ten neural Gaussians, recording the utilized Gaussians through an additional mask. This approach reduces the storage overhead associated with the offsets O_i^j.

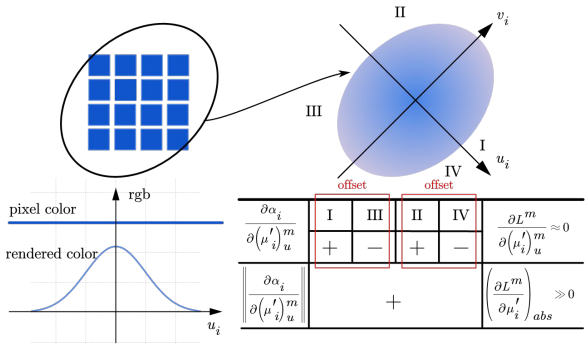

Fig. 2: Abs gradient elimination of numerical directions leads to unwanted densification.

We adjust the anchor's level $\mathbf{L}_i$ based on LOD bias θ_i and scale S_i, increasing it when $S_i < \tau_l$ and decreasing it when $\theta_i > \tau_\theta$, rather than empirically relying on $\frac{\partial L}{\partial \mu_i'}$ in [15].

D. Analytic Anti-Aliasing Filtering

In large scenes or aerial perspectives, changing the camera focal length is often necessary for rendering in order to view thumbnails. During optimization, the stochastic nature of MLP may generate a large number of tiny and intricate neural Gaussians. However, many of these Gaussians do not satisfy the Nyquist sampling theorem [27], leading to aliasing issues when the viewpoint changes. Mip-Splatting [27] proposed a 3D smoothing filter to ensure that the projected 2D Gaussian covers at least two pixels in at least one training view. Nevertheless, it employs some approximation during derivation thus requiring fine-tuning hyperparameters. We propose an analytical 3D smoothing filter that addresses this issue.

Given a desired circle with a radius $r_p = \frac{\sqrt{5}}{2}$ on the pixel plane, ensuring that at least two pixels are covered, and intrinsic parameters f_x, f_y, we derive the radius of sphere $\mathbf{r}$ centered at $\mathbf{x} = (x_c, y_c, z_c)$ in camera coordinate and tangent to the viewing cone [37]:

$$\mathbf{r} = \frac{||\mathbf{x}||_2 \cdot r_p / \sqrt{f_x f_y}}{||\mathbf{v_d}||_2 \cdot \sqrt{(\sqrt{||\mathbf{v_d}||_2^2 - 1} - r_p/\sqrt{f_x f_y})^2 + 1}} \tag{17}$$

where $\mathbf{v_d} = \frac{\sqrt{x_c^2+y_c^2+z_c^2}}{z_c}$. We define $\mathbf{r}_f = (\frac{\mathbf{r}}{3})^2$ as the scale of the 3D Gaussian low-pass filter to derive the filtered Gaussian kernel:

$$\mathcal{G}_i(\mathbf{x}) = \sqrt{\frac{\det(\mathbf{\Sigma}_i)}{\det(\mathbf{\Sigma}_i + \mathbf{r}_f \cdot I)}} \exp^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T (\mathbf{\Sigma}_i + \mathbf{r}_f \cdot I)^{-1} (\mathbf{x}-\mu_i)} . \tag{18}$$

Unlike [27], we derive an analytical 3D smoothing filter without needing an empirical coefficient. For stable training process, we calculate the size of the filter based on anchor positions instead of directly rely on neural gaussians. We further employ 2D mip filter to fit box filter as in [27].

V. Experiments

Experiments are conducted with an NVIDIA RTX 4090D 24GB GPU, an Intel Core i7-14700kf and 64GB RAM. For

TABLE I: Quantitative comparison on real-world custom dataset. We highlight best, second best and third best results.

Dataset	Mip-NeRF360 [31]				Tanks&Temples [52]				Deep Blending [53]			
Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	MB↓	PSNR ↑	SSIM ↑	LPIPS ↓	MB↓	PSNR ↑	SSIM ↑	LPIPS ↓	MB↓
Mip-NeRF360 [31]	27.69	0.792	0.237	10	23.14	0.841	0.183	-	29.40	0.901	0.245	-
iNGP [26]	27.24	0.760	0.348	48	22.22	0.759	0.256	50.4	29.25	0.883	0.368	50.4
3D-GS [2]	27.42	0.813	0.218	824	23.69	0.845	0.178	453	29.47	0.900	0.247	695
Mip-Splatting [27]	27.64	0.828	0.189	1014	23.78	0.859	0.156	598	29.40	0.903	0.239	859
Scaffold-GS [19]	27.73	0.811	0.227	179	24.03	0.850	0.175	80	30.12	0.905	0.257	56
Octree-GS [15]	27.70	0.814	0.219	144	24.26	0.856	0.161	88	30.11	0.904	0.256	60
Ours	27.56	0.811	0.217	134	24.03	0.857	0.163	80	29.64	0.903	0.253	73

TABLE II: Quantitative comparison on aerial perspective dataset[14]. - indicates that a CUDA out of memory issue is encountered.

Method		Amsterdam	Barcelona	Bilbao	Chicago	Hollywood	Pompidou	Quebec	Rome	Average
3DGS [2]	PSNR↑	27.62	27.39	28.76	28.14	26.18	-	28.70	27.52	27.76
	SSIM↑	0.913	0.915	0.915	0.928	0.868	-	0.931	0.914	0.912
	LPIPS↓	0.101	0.089	0.099	0.085	0.134	-	0.094	0.102	0.101
	MB↓	1457	2002	1307	1493	1667	-	1427	1596	1564
	GS(k)↓	2293	2940	2046	2624	2714	-	2054	2262	2419
Mip-Splatting [27]	PSNR↑	28.35	28.04	29.32	28.28	26.79	-	29.46	28.44	28.38
	SSIM↑	0.926	0.925	0.924	0.934	0.884	-	0.944	0.927	0.923
	LPIPS↓	0.083	0.076	0.089	0.079	0.122	-	0.081	0.087	0.088
	MB↓	1745	2201	1570	1766	1854	-	1584	1788	1787
	GS(k)↓	2921	3388	2487	3116	3099	-	2351	2740	2872
Scaffold-GS [19]	PSNR↑	28.06	27.53	29.23	28.35	26.29	27.12	28.77	27.97	27.92
	SSIM↑	0.917	0.913	0.917	0.923	0.864	0.916	0.929	0.917	0.912
	LPIPS↓	0.098	0.094	0.102	0.090	0.163	0.098	0.098	0.100	0.105
	MB↓	212.6	183.1	171.1	165.7	142.4	185.6	153.5	186.8	175.1
	GS(k)↓	1526	1129	1114	1318	992	1293	920	1275	1196
Octree-GS [15]	PSNR↑	27.77	28.11	29.15	28.77	26.61	26.88	28.84	28.52	28.08
	SSIM↑	0.917	0.924	0.919	0.932	0.878	0.917	0.939	0.930	0.920
	LPIPS↓	0.095	0.077	0.095	0.080	0.137	0.096	0.080	0.080	0.092
	MB↓	303.9	343.1	274.5	295.2	256.9	310.9	297.3	351.6	304.0
	GS(k)↓	1139	1417	1129	1208	1279	1078	1286	1613	1267
Ours	PSNR↑	28.18	28.07	29.23	28.87	26.64	27.19	28.86	28.27	28.16
	SSIM↑	0.921	0.923	0.922	0.931	0.879	0.917	0.937	0.922	0.920
	LPIPS↓	0.094	0.079	0.090	0.080	0.143	0.099	0.084	0.093	0.095
	MB↓	181.7	192.8	193.3	191.1	164.1	177.5	171.0	180.1	181.5
	GS(k)↓	1385	1240	1372	1159	922	729	1006	790	1075

TABLE III: Single-scale training and multi-scale testing on BungeeNeRF dataset. T means training resolution. We conduct training at a resolution of 1600x900, while the original image resolution is 1920x1080, thus we only report the rendering results at 1920x1080 (denoted as Full) as a case of zoom-in.

Res	Full			T			1/2			1/4			1/8			Avg		
Method|Metrics	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Mip-Splatting[27]	27.05	0.895	0.129	28.38	0.923	0.088	29.76	0.950	0.052	30.44	0.959	0.044	29.65	0.954	0.047	29.23	0.939	0.068
Scaffold-GS[19]	26.45	0.876	0.152	27.92	0.912	0.105	27.55	0.917	0.074	23.28	0.804	0.121	19.81	0.624	0.197	24.27	0.805	0.136
Octree-GS[15]	26.56	0.885	0.135	28.08	0.920	0.092	26.66	0.912	0.076	21.76	0.764	0.138	18.59	0.566	0.217	23.39	0.782	0.142
Ours	26.84	0.887	0.138	28.16	0.920	0.095	29.52	0.949	0.056	29.98	0.959	0.046	29.32	0.957	0.046	28.91	0.938	0.071

Fig. 3: Qualitative comparison on Mip-NeRF 360 dataset[31]. Previous anchor-based methods tend to produce blurring in high-frequency details, while our approach maintains better quality.

a fair comparison, we set densification to stop after 15k iterations and perform a total of 30k rounds. We do not impose restrictions on the resolution of views, except that images with a resolution greater than 1.6k will be resized. For hash table, we set resolution level $L = 16$, scale ratio $s = 2.0$ and base resolution $\Phi = 16$ with 2 dimension feature per level. The maximum capacity of per level N is selected from $2^{17}, 2^{18}, 2^{19}$ according to the complexity of the scene. For prune policy, we set $\tau_d = 0.15 \cdot \frac{iteration}{15000}$, $\tau_\alpha = 0.005$. We opt "round" as rounding type, thus we set $\tau_\theta = 0.5$, $c_\theta = 2$ and initial LOD bias is 0.25. Same as previous work[15, 19], except for conventional L1 loss and SSIM[54] loss, we add a regularization on neural Gaussian scale $\mathbf{L}_{vol} = \sum_{i=1}^{N_{ng}} Prod(S_i)$ where $Prod(\cdot)$ means multiply each value in a vector, N_{ng} is the number of activated neural Gaussians.

A. Datasets and Baseline

We carry out experiments on three public datasets, including 9 scenes in MipNeRF-360 [31], two scenes in tanks & temples [52], two scenes in deep blending [53] and 8 scenes from BungeeNeRF [14]. The first three datasets involve indoor and outdoor scenes shot from common perspectives. BungeeNeRF dataset [14] contains large-scale urban scenes

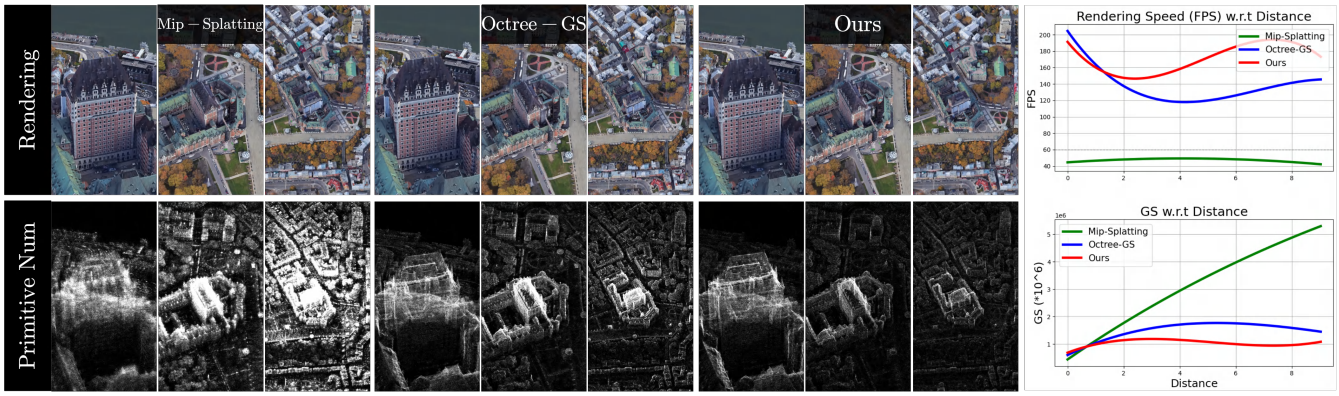

Fig. 4: Visualization of changes in FPS and number of Gaussians when moving away from observation area tested on Quebec from [14]. The brighter the second row of images, the greater the number of Gaussians within the frustum.

shot from aerial perspectives. Along with PSNR, SSIM [54] and LPIPS [55] for visual quality evaluation, we report average primitives in frustum and storage size.

We choose MipNeRF-360 [31], iNGP [26], 3DGS [2], Mip-Splatting [27], Scaffold-GS [19] and Octree-GS [15] as comparison methods. We extract results of MipNeRF-360 from paper because of the training difficulty. NeRFAcc [56] is used to test iNGP [26]. The remains are tested using official implementations.

B. Results

Table. I shows the results of indoor and outdoor scenes captured from a conventional perspective. Our approach strikes a balance between visual quality and storage space. Shown in Fig. 3, our method can better model high frequency details, which proves that Eq. (16) is effective. Table. II and Fig. 4 exhibits the evaluations on more challenging dataset[14] which contains urban scenes with large perspective changes. Our method effectively keeps the number of anchors involved in the rendering process due to the adoption of LOD structure. It is worth noting that latest Mip-Splatting[27] integrates absolute gradient[57] which increase the performance. However, this also brings up the problem of the rapid growth of primitives. Our mehod achieves competitive results while keep the number of parameters under control. We further partition the test images into four groups based on distance variations to investigate the impact of LOD. The results demonstrate the effectiveness of LOD.

Table.III shows the evaluation of the zoom rendering performance with single-scale training and multi-scale testing on BungeeNeRF [14] dataset. The filter reduces aliasing thus achieving better performance. Although Mip-splatting outperforms in most cases, it requires 10x more memory usage on average compared to the proposed method.

C. Ablation

Base Color: As illustrated in Sec. IV-A, we introduce a base color for each anchor to overcome the difficulty of learning high-frequency signals. Tab. V shows that directly using the output of MLP as color will cause insufficient representation ability.

TABLE IV: Group evaluation results on BungeeNeRF dataset [14]. From 1 to 4, distance is getting farther.

Dataset	distance-1		distance-2		distance-3		distance-4	
Method|Metrics	PSNR↑	GS(k)	PSNR↑	GS(k)	PSNR↑	GS(k)	PSNR↑	GS(k)
3DGS[2]	29.84	598	28.81	1430	27.63	2787	24.98	4735
Mip-Splatting[27]	30.29	809	29.71	1801	28.56	3332	25.20	5400
Scaffold-GS	30.14	331	28.71	757	27.72	1359	25.36	2250
Octree-GS	30.69	634	28.84	1156	27.82	1573	25.25	1682
Ours	30.89	681	29.01	1036	27.85	1264	25.19	1300

TABLE V: Ablation study on Amsterdam sequence from [14].

Component|Metrics	PSNR↑	SSIM↑	LPIPS↓	MB↓
w/o Base Color	28.06	0.918	0.097	170.8
w/o LOD Bias	27.95	0.919	0.096	181.5
w/o Analytic Filter	28.19	0.921	0.094	181.5
w/o Anchor Management	28.24	0.921	0.093	216.1
Full Model	28.18	0.921	0.094	181.7

LOD Bias: To evaluate the effectiveness of LOD bias, we set 0 as its initial value. This ensures that Eq. (12) does not select additional anchors. The results show that LOD bias can lead to better visual quality owing to smooth transitions.
Analytic Filter: We derive the 3D filter without empirical parameters. Compare the results of the proposed analytical filter with the original formula in Mip-Splatting[27], the results are almost the same. This demonstrates the correctness of our proposed analytical filter.
Adaptive Anchor Management: As for anchor management strategy, it reduces the storage of the neural Gaussians that contribute little to the rendering. From Tab.V, it can be seen that our adaptive anchor management policy reduces storage requirements by about 16% with a slight performance degradation. More parameter have stronger fitting ability, such results are in line with expectations.

VI. Conclusions

In this paper, to tackle the challenges of storage overhead and rendering speed arising from large-scale reconstruction, we propose an anchor-based 3DGS with capacity constrained multi-resolution hash encoding and adaptive anchor management. Additionally, we support LOD and anti-aliasing for large scenes with varying rendering scale. Future works include finding a space warping that allows for more efficient utilization of features and extending to larger scale scenes.

REFERENCES

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." In: *ACM Trans. Graph.* 42.4 (2023), pp. 139–1.

[3] W. Yifan, F. Serena, S. Wu, C. Öztireli, and O. Sorkine-Hornung. "Differentiable surface splatting for point-based geometry processing". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–14.

[4] Q. Zhang, S.-H. Baek, S. Rusinkiewicz, and F. Heide. "Differentiable point-based radiance fields for efficient view synthesis". In: *SIGGRAPH Asia 2022 Conference Papers*. 2022, pp. 1–12.

[5] L. Xu, V. Agrawal, W. Laney, T. Garcia, A. Bansal, C. Kim, S. Rota Bulò, L. Porzi, P. Kontschieder, A. Božič, D. Lin, M. Zollhöfer, and C. Richardt. "VR-NeRF: High-Fidelity Virtualized Walkable Spaces". In: *SIGGRAPH Asia Conference Proceedings*. 2023.

[6] Y. Jiang, C. Yu, T. Xie, X. Li, Y. Feng, H. Wang, M. Li, H. Lau, F. Gao, Y. Yang, et al. "VR-GS: a physical dynamics-aware interactive gaussian splatting system in virtual reality". In: *ACM SIGGRAPH 2024 Conference Papers*. 2024, pp. 1–1.

[7] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. "Nice-slam: Neural implicit scalable encoding for slam". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12786–12796.

[8] H. Wang, J. Wang, and L. Agapito. "Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 13293–13302.

[9] M. M. Johari, C. Carta, and F. Fleuret. "Eslam: Efficient dense slam system based on hybrid representation of signed distance fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 17408–17419.

[10] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten. "SplaTAM: Splat Track & Map 3D Gaussians for Dense RGB-D SLAM". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 21357–21366.

[11] X. Lang, L. Li, H. Zhang, F. Xiong, M. Xu, Y. Liu, X. Zuo, and J. Lv. "Gaussian-LIC: Photo-realistic LiDAR-Inertial-Camera SLAM with 3D Gaussian Splatting". In: *arXiv preprint arXiv:2404.06926* (2024).

[12] X. Zhou, Z. Lin, X. Shan, Y. Wang, D. Sun, and M.-H. Yang. "Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 21634–21643.

[13] Y. Yan, H. Lin, C. Zhou, W. Wang, H. Sun, K. Zhan, X. Lang, X. Zhou, and S. Peng. "Street Gaussians: Modeling Dynamic Urban Scenes with Gaussian Splatting". In: *ECCV*. 2024.

[14] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin. "Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering". In: *European conference on computer vision*. Springer. 2022, pp. 106–122.

[15] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai. "Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians". In: *arXiv preprint arXiv:2403.17898* (2024).

[16] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis. "A hierarchical 3d gaussian representation for real-time rendering of very large datasets". In: *ACM Transactions on Graphics (TOG)* 43.4 (2024), pp. 1–15.

[17] Y. Chen and G. H. Lee. "DoGaussian: Distributed-Oriented Gaussian Splatting for Large-Scale 3D Reconstruction Via Gaussian Consensus". In: *arXiv preprint arXiv:2405.13943* (2024).

[18] H. Turki, D. Ramanan, and M. Satyanarayanan. "Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12922–12931.

[19] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai. "Scaffold-gs: Structured 3d gaussians for view-adaptive rendering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 20654–20664.

[20] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park. "Compact 3d gaussian representation for radiance field". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 21719–21728.

[21] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang. "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps". In: *arXiv preprint arXiv:2311.17245* (2023).

[22] S. Girish, K. Gupta, and A. Shrivastava. "Eagles: Efficient accelerated 3d gaussians with lightweight encodings". In: *arXiv preprint arXiv:2312.04564* (2023).

[23] Y. Liu, H. Guan, C. Luo, L. Fan, J. Peng, and Z. Zhang. "Citygaussian: Real-time high-quality large-scale scene rendering with gaussians". In: *arXiv preprint arXiv:2404.01133* (2024).

[24] J. Cui, J. Cao, Y. Zhong, L. Wang, F. Zhao, P. Wang, Y. Chen, Z. He, L. Xu, Y. Shi, et al. "LetsGo: Large-Scale Garage Modeling and Rendering via LiDAR-Assisted Gaussian Primitives". In: *arXiv preprint arXiv:2404.09748* (2024).

[25] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of detail for 3D graphics*. Elsevier, 2002.

[26] T. Müller, A. Evans, C. Schied, and A. Keller. "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM transactions on graphics (TOG)* 41.4 (2022), pp. 1–15.

[27] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger. "Mip-splatting: Alias-free 3d gaussian splatting". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 19447–19456.

[28] M. Kazhdan and H. Hoppe. "Screened poisson surface reconstruction". In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), pp. 1–13.

[29] J. Behley and C. Stachniss. "Efficient surfel-based SLAM using 3D laser range data in urban environments." In: *Robotics: science and systems*. Vol. 2018. 2018, p. 59.

[30] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 5855–5864.

[31] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. "Mip-nerf 360: Unbounded anti-aliased neural radiance fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5470–5479.

[32] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt. "Neural sparse voxel fields". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15651–15663.

[33] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. "Plenoxels: Radiance fields without neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5501–5510.

[34] C. Sun, M. Sun, and H.-T. Chen. "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5459–5469.

[35] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. "Plenoctrees for real-time rendering of neural radiance fields". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5752–5761.

[36] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. "Zip-nerf: Anti-aliased grid-based neural radiance fields". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19697–19705.

[37] W. Hu, Y. Wang, L. Ma, B. Yang, L. Gao, X. Liu, and Y. Ma. "Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19774–19783.

[38] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. "Tensorf: Tensorial radiance fields". In: *European conference on computer vision*. Springer. 2022, pp. 333–350.

[39] S. Fridovich-Keil, G. Meanti, F. R. Warburg, B. Recht, and A. Kanazawa. "K-planes: Explicit radiance fields in space, time, and appearance". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12479–12488.

[40] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai. "Hac: Hash-grid assisted context for 3d gaussian splatting compression". In: *European Conference on Computer Vision*. Springer. 2024, pp. 422–438.

[41] Y. Chen, Q. Wu, M. Harandi, and J. Cai. "How far can we compress instant-ngp-based nerf?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 20321–20330.

[42] L. Williams. "Pyramidal parametrics". In: *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*. 1983, pp. 1–11.

[43] H. Turki, M. Zollhöfer, C. Richardt, and D. Ramanan. "Pynerf: Pyramidal neural radiance fields". In: *Advances in Neural Information Processing Systems* 36 (2024).

[44] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. "Neural geometric level of detail: Real-time rendering with implicit 3d shapes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11358–11367.

[45] P. Weier, T. Zirr, A. Kaplanyan, L.-Q. Yan, and P. Slusallek. "Neural Prefiltering for Correlation-Aware Levels of Detail". In: *ACM Transactions on Graphics (TOG)* 42.4 (2023), pp. 1–16.

[46] S. Bako, P. Sen, and A. Kaplanyan. "Deep appearance prefiltering". In: *ACM Transactions on Graphics* 42.2 (2023), pp. 1–23.

[47] Z. Yan, W. F. Low, Y. Chen, and G. H. Lee. "Multi-scale 3d gaussian splatting for anti-aliased rendering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 20923–20931.

[48] Z. Liang, Q. Zhang, W. Hu, Y. Feng, L. Zhu, and K. Jia. "Analytic-Splatting: Anti-Aliased 3D Gaussian Splatting via Analytic Integration". In: *arXiv preprint arXiv:2403.11056* (2024).

[49] J. L. Schonberger and J.-M. Frahm. "Structure-from-motion revisited". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4104–4113.

[50] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross. "EWA splatting". In: *IEEE Transactions on Visualization and Computer Graphics* 8.3 (2002), pp. 223–238.

[51] Z. Ye, W. Li, S. Liu, P. Qiao, and Y. Dou. “AbsGS: Recovering fine details in 3D Gaussian Splatting”. In: *ACM Multimedia 2024*. 2024.
[52] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. “Tanks and temples: Benchmarking large-scale scene reconstruction”. In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), pp. 1–13.
[53] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow. “Deep blending for free-viewpoint image-based rendering”. In: *ACM Transactions on Graphics (ToG)* 37.6 (2018), pp. 1–15.
[54] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.
[55] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 586–595.
[56] R. Li, H. Gao, M. Tancik, and A. Kanazawa. “Nerfacc: Efficient sampling accelerates nerfs”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 18537–18546.
[57] Z. Yu, T. Sattler, and A. Geiger. “Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes”. In: *arXiv preprint arXiv:2404.10772* (2024).