Mip-Splatting: Alias-free 3D Gaussian Splatting

Zehao Yu[1,2] Anpei Chen[†,1,2] Binbin Huang[3] Torsten Sattler[4] Andreas Geiger[1,2]

[1]University of Tübingen [2]Tübingen AI Center [3]ShanghaiTech University
[4]Czech Technical University in Prague

https://niujinshuchong.github.io/mip-splatting

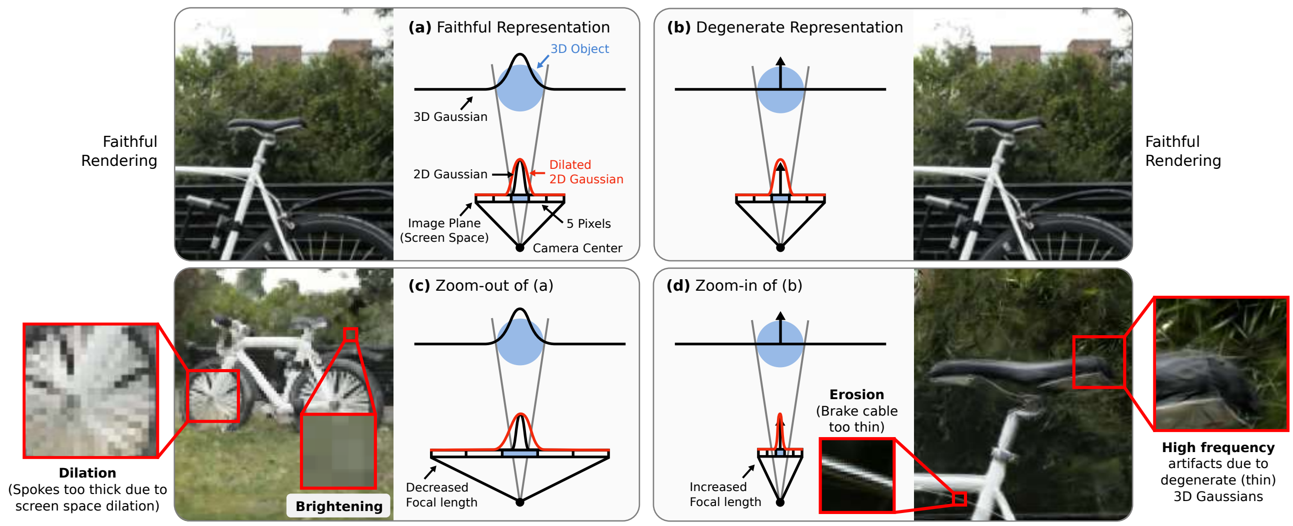

Figure 1. **3D Gaussian Splatting** [18] renders images by representing **3D Objects** as **3D Gaussians** which are projected onto the image plane followed by **2D Dilation** in screen space as shown in (a). Its intrinsic shrinkage bias leads to degenerate 3D Gaussians that exceed the sampling limit as illustrated by the δ function in (b) while rendering similarly in 2D due to the dilation operation. However, when changing the sampling rate (via the focal length or camera distance), we observe strong dilation effects (c) and high frequency artifacts (d).

Abstract

Recently, 3D Gaussian Splatting has demonstrated impressive novel view synthesis results, reaching high fidelity and efficiency. However, strong artifacts can be observed when changing the sampling rate, e.g., by changing focal length or camera distance. We find that the source for this phenomenon can be attributed to the lack of 3D frequency constraints and the usage of a 2D dilation filter. To address this problem, we introduce a 3D smoothing filter to constrains the size of the 3D Gaussian primitives based on the maximal sampling frequency induced by the input views. It eliminates high-frequency artifacts when zooming in. Moreover, replacing 2D dilation with a 2D Mip filter, which simulates a 2D box filter, effectively mitigates aliasing and dilation issues. Our evaluation, including scenarios such a training on single-scale images and testing on multiple scales, validates the effectiveness of our approach.

† Corresponding author.

1. Introduction

Novel View Synthesis (NVS) plays a critical role in computer graphics and computer vision, with various applications including virtual reality, cinematography, robotics, and more. A particularly significant advancement in this field is the Neural Radiance Field (NeRF) [28], introduced by Mildenhall et al. in 2020. NeRF utilizes a multilayer perceptron (MLP) to represent geometry and view-dependent appearance effectively, demonstrating remarkable novel view rendering quality. Recently, 3D Gaussian Splatting (3DGS) [18] has gained attention as an appealing alternative to both MLP [28] and feature grid-based representations [4, 11, 24, 32, 46]. 3DGS stands out for its impressive novel view synthesis results, while achieving real-time rendering at high resolutions. This effectiveness and efficiency, coupled with the potential integration into the standard rasterization pipeline of GPUs represents a significant step towards practical usage of NVS methods.

→ pre-computed Gaussians

→ 3DGS

Specifically, 3DGS represents complex scenes as a set

But MLP can be more memory-efficient

Q3: 2D Gaussians are too small
↓
Increase the scale

of 3D Gaussians, which are rendered to screen space through splatting-based rasterization. The attributes of each 3D Gaussian, i.e., position, size, orientation, opacity, and color, are optimized through a multi-view photometric loss. Thereafter, a 2D dilation operation is applied in screen space for low-pass filtering. Although 3DGS has demonstrated impressive NVS results, it produces artifacts when camera views diverge from those seen during training, such as zoom in and zoom out, as illustrated in Figure 1. We find that the source for this phenomenon can be attributed to the lack of 3D frequency constraints and the usage of a 2D dilation filter. Specifically, zooming out leads to a reduced size of the projected 2D Gaussians in screen space, while applying the same amount of dilation results in dilation artifacts. Conversely, zooming in causes erosion artifacts since the projected 2D Gaussians expand, yet dilation remains constant, causing erosion and resulting in incorrect gaps between Gaussians in the 2D projection.

artifacts of 3DGS

future work in 3DGS

To resolve these issues, we propose to regularize the 3D representation in 3D space. Our key insight is that the highest frequency that can be reconstructed of a 3D scene is inherently constrained by the sampling rates of the input images. We first derive the multi-view frequency bounds of each Gaussian primitive based on the training views according to the Nyquist-Shannon Sampling Theorem [33, 45]. By applying a low-pass filter to the 3D Gaussian primitives in 3D space during the optimization, we effectively restrict the maximal frequency of the 3D representation to meet the Nyquist limit. Post-training, this filter becomes an intrinsic part of the scene representation, remaining constant regardless of viewpoint changes. Consequently, our method eliminates the artifacts presents in 3DGS [18] when zooming in, as shown in the 8× higher resolution image in Figure 2.

Nonetheless, rendering the reconstructed scene at lower sampling rates (e.g., zooming out) results in aliasing. Previous work [1–3, 17] address aliasing by employing cone tracing and applying pre-filtering to the input positional or feature encoding, which is not applicable to 3DGS. Thus, we introduce a 2D Mip filter (à la "mipmap") specifically designed to ensure alias-free reconstruction and rendering across different scales. Our 2D Mip filter mimics the 2D box filter inherent to the actual physical imaging process [29, 37, 48], by approximating it with a 2D Gaussian low pass filter. In contrast to previous work [1–3, 17] that rely on the MLP's ability to interpolate multi-scale signals during training with multi-scale images, our closed-form modification to the 3D Gaussian representation results in excellent out-of-distribution generalization: Training at a single sampling rate enables faithful rendering at various sampling rates different from those used during training as demonstrated by the 1/4× down-sampled image in Figure 2.

In summary, we make the following contributions:

- We analyze and identify the root of 3DGS's artefacts when changing sampling rates.

① Lack of 3D frequency constraints
② Rely on a 2D dilation filter

Main Contributions

- We introduce a **3D smoothing filter** for 3DGS to effectively regularize the maximum frequency of 3D Gaussian primitives, resolving the artifacts observed in out-of-distribution renderings of prior methods [18, 59].
- We replace the 2D dilation filter with a **2D Mip filter** to address aliasing and dilation artifacts.
- Experiments on challenging benchmark datasets [2, 28] demonstrate the effectiveness of Mip-Splatting when modifying the sampling rate.

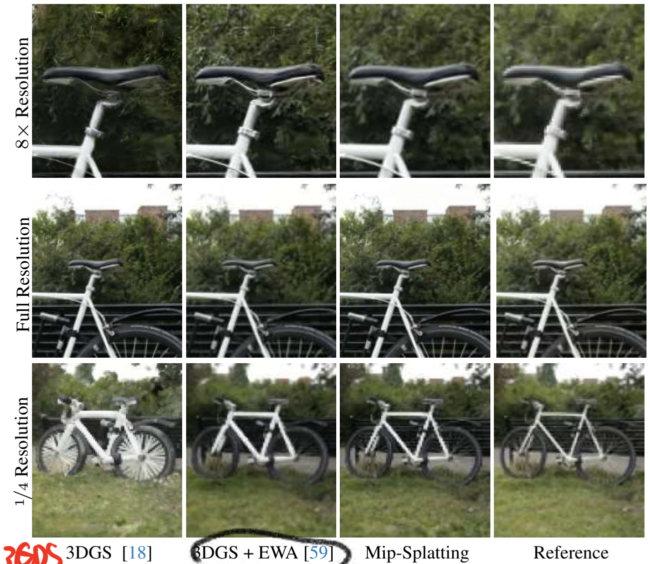

Figure 2. We trained all the models on single-scale (full resolution here) images and rendered images with different resolutions by changing focal length. While all methods show similar performance at training scale, we observe strong artifacts in previous work [18, 59] when changing the sampling rate. By contrast, our Mip-Splatting renders faithful images across different scales.

2. Related Work

Novel View Synthesis: NVS is the process of generating new images from viewpoints different from those of the original captures [12, 22]. NeRF [28], which leverages volume rendering [10, 21, 25, 26], has become a standard technique in the field. NeRF utilizes MLPs [5, 27, 34] to model scenes as continuous functions, which, despite their compact representation, impede rendering speed due to the expensive MLP evaluation that is required for each ray point. Subsequent methods [16, 40, 41, 52, 54] distill a pretrained NeRF into a sparse representation, enabling real-time rendering of NeRFs. Further advancements have been made to improve the training and rendering of NeRF with advanced scene representations [4, 6, 11, 18, 19, 24, 32, 46, 51]. In particular, 3D Gaussians Splatting (3DGS) [18] demonstrated impressive novel view synthesis results, while achieving real-time rendering at high-definition resolutions.

The limitation of 3DGS

Importantly, 3DGS represents the scene explicitly as a collection of 3D Gaussians and uses rasterization instead of ray tracing. Nevertheless, 3DGS focuses on in-distribution evaluation where training and testing are conducted at similar sampling rates (focal length/scene distance). In this paper, we study the out-of-distribution generalization of 3DGS, training models at a single scale and evaluating it across multiple scales.

Primitive-based Differentiable Rendering: Primitive-based rendering techniques, which rasterize geometric primitives onto the image plane, have been explored extensively due to their efficiency [13, 14, 38, 44, 59, 60]. Differentiable point-based rendering methods [20, 36, 39, 43, 49, 53, 57] offer great flexibility in representing intricate structures and are thus well-suited for novel view synthesis. Notably, Pulsar [20] stands out for its efficient sphere rasterization. The more recent 3D Gaussian Splatting (3DGS) work [18] utilizes anisotropic Gaussians [59] and introduces a tile-based sorting for rendering, achieving remarkable frame rates. Despite its impressive results, 3DGS exhibits strong artifacts when rendering at a different sampling rate. We address this issue by introducing a 3D smoothing filter to constrain the maximal frequencies of the 3D Gaussian primitive representation, and a 2D Mip filter that approximates the box filter of the physical imaging process for alias-free rendering.

solve the limitation

Anti-aliasing in Rendering: There are two principal strategies to combat aliasing: *super-sampling*, which increases the number of samples [7], and *prefiltering*, which applies low-pass filtering to the signal to meet the Nyquist limit [8, 15, 31, 47, 50, 59]. For example, EWA splatting [59] applies a Gaussian low pass filter to the projected 2D Gaussian in screen space to produce a band limited output respecting the Nyquist frequency of the image. While we also apply a band-limited filter to the Gaussian primitives, our band-limited filter is applied in 3D space and the filter size is fully determined by the training images not the images to be rendered. While our 2D Mip filter is also a Gaussian low pass filter in screen space, it approximates the box filter of the physical imaging process, approximating a single pixel. Conversely, the EWA filter limits the frequency signal's bandwidth to the rendered image, and the size of the filter is chosen empirically. A critical difference to [59] is that we tackle the reconstruction problem, optimizing the 3D Gaussian representation via inverse rendering while EWA splatting only considers the rendering problem.

Both filters are

Recent neural rendering methods integrate pre-filtering to mitigate aliasing [1–3, 17, 58]. Mip-NeRF [1], for instance, introduced an integrated position encoding (IPE) to attenuate high-frequency details. A similar idea is adapted for feature grid-based representations [3, 17, 58]. Note that these approaches require multi-scale images extracted from the original data for supervision. In contrast, our approach is based on 3DGS [18] and determines the necessary low-pass filter size based on pixel size, allowing for alias-free rendering at scales unobserved during training.

3. Preliminaries

In this section, we first review the sampling theorem in Section 3.1, laying the foundation for understanding the aliasing problem. Subsequently, we introduce 3D Gaussian Splatting (3DGS) [18] and its rendering process in Section 3.2.

3.1. Sampling Theorem

The Sampling Theorem, also known as the Nyquist-Shannon Sampling Theorem [33, 45], is a fundamental concept in signal processing and digital communication that describes the conditions under which a continuous signal can be accurately represented or reconstructed from its discrete samples. To accurately reconstruct a continuous signal from its discrete samples without loss of information, the following conditions must be met:

3D Gaussian

Condition 1 *The continuous signal must be band-limited and may not contain any frequency components above a certain maximum frequency* ν.

Camera → sampling rate

Condition 2 *The sampling rate* $\hat{\nu}$ *must be at least twice the highest frequency present in the continuous signal:* $\hat{\nu} \geq 2\nu$. → image

In practice, to satisfy the constraints when reconstructing a signal from discrete samples, a low-pass or anti-aliasing filter is applied to the signal before sampling. The filter eliminates any frequency components above $\frac{\hat{\nu}}{2}$ and attenuates high-frequency content that could lead to aliasing.

3.2. 3D Gaussian Splatting

Prior works [18, 59] propose to represent a 3D scene as a set of scaled 3D Gaussian primitives $\{\mathcal{G}_k | k = 1, \cdots, K\}$ and render an image using volume splatting. The geometry of each scaled 3D Gaussian $\mathcal{G}_k$ is parameterized by an opacity (scale) $\alpha_k \in [0, 1]$, center $\mathbf{p}_k \in \mathbb{R}^{3\times 1}$ and covariance matrix $\mathbf{\Sigma}_k \in \mathbb{R}^{3\times 3}$ defined in world space:

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

To constrain $\mathbf{\Sigma}_k$ to the space of valid covariance matrices, a semi-definite parameterization $\mathbf{\Sigma}_k = \mathbf{O}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{O}_k^T$ is used. Here, $\mathbf{s} \in \mathbb{R}^3$ is a scaling vector and $\mathbf{O} \in \mathbb{R}^{3\times 3}$ is a rotation matrix, parameterized by a quaternion [18].

To render an image for a given view point defined by rotation $\mathbf{R} \in \mathbb{R}^{3\times 3}$ and translation $\mathbf{t} \in \mathbb{R}^3$, the 3D Gaussians $\{\mathcal{G}_k\}$ are first transformed into camera coordinates:

$$\mathbf{p}'_k = \mathbf{R}\,\mathbf{p}_k + \mathbf{t}, \quad \mathbf{\Sigma}'_k = \mathbf{R}\,\mathbf{\Sigma}_k\,\mathbf{R}^T \tag{2}$$

Afterwards, they are projected to ray space via a local affine transformation

$$\mathbf{\Sigma}_k'' = \mathbf{J}_k \, \mathbf{\Sigma}_k' \, \mathbf{J}_k^T \tag{3}$$

where the Jacobian matrix $\mathbf{J}_k$ is an affine approximation to the projective transformation defined by the center of the 3D Gaussian $\mathbf{p}_k'$. By skipping the third row and column of $\mathbf{\Sigma}_k''$, we obtain a 2D covariance matrix $\mathbf{\Sigma}_k^{2D}$ in ray space, and we use $\mathcal{G}_k^{2D}$ to refer to the corresponding scaled 2D Gaussian, see [18] for details.

Finally, 3DGS [18] utilizes spherical harmonics to model view-dependent color $\mathbf{c}_k$ and renders image via alpha blending according to the primitive's depth order $1, \ldots, K$:

$$\mathbf{c}(\mathbf{x}) = \sum_{k=1}^{K} \mathbf{c}_k \, \alpha_k \, \mathcal{G}_k^{2D}(\mathbf{x}) \prod_{j=1}^{k-1} (1 - \alpha_j \, \mathcal{G}_j^{2D}(\mathbf{x})) \tag{4}$$

Dilation: To avoid degenerate cases where the projected 2D Gaussians are too small in screen space, i.e., smaller than a pixel, the projected 2D Gaussians are dilated as follows:

$$\mathcal{G}_k^{2D}(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p}_k)^T \left(\mathbf{\Sigma}_k^{2D} + s\,\mathbf{I}\right)^{-1} (\mathbf{x}-\mathbf{p}_k)} \tag{5}$$

where $\mathbf{I}$ is a 2D identity matrix and s is a scalar dilation hyperparameter. Note that this operator adjusts the scale of the 2D Gaussian while leaving its maximum unchanged. As this effect is similar to that of dilation operators in morphology, we called it a 2D screen space dilation operation[*].

Reconstruction: As the rendering process is fast and differentiable, the 3D Gaussian parameters can be efficiently optimized using a multi-view loss. During optimization, 3D Gaussians are adaptively added and deleted to better represent the scene. We refer the reader to [18] for details.

4. Sensitivity to Sampling Rate

Based on experience

In traditional forward splatting, the centers $\mathbf{p}_k$ and colors $\mathbf{c}_k$ of Gaussian primitives are predetermined, whereas the 3D Gaussian covariance $\mathbf{\Sigma}_k$ are chosen empirically [42, 59]. In contrast, 3DGS [18], optimizes all parameters jointly through an inverse rendering framework by backpropagating a multi-view photometric loss.

We observe that this optimization suffers from ambiguities as illustrated in Figure 1 which shows a simple example involving one object and an image sensor with 5 pixels. Consider the 3D object in (a), its approximation by a 3D Gaussian and its projection into screen space (blue pixel). Due to screen space dilation (Eq. 5) with a Gaussian kernel (size $\approx$ 1 pixel), the degenerate 3D Gaussian represented by a Dirac δ function in (b) leads to a similar image. In order to represent high frequency details in real world scenes, the dilated 2D Gaussians would become small, since the smaller the Gaussian, the higher the frequency it represents, resulting in systematically underestimation of its scale.

While this does not affect rendering at similar sampling rates (cf. Figure 1 (a) vs. (b)), it leads to erosion effects when zooming in or moving the camera closer. This is because the dilated 2D Gaussians become smaller in screen space. In this case, the rendered image exhibits high-frequency artifacts, rendering object structures thinner than they actually appear as illustrated in Figure 1 (d).

Conversely, screen space dilation also negatively affects rendering when decreasing the sampling rate as illustrated in Figure 1 (c) which shows a zoomed-out version of (a). In this case, dilation spreads radiance in a physically incorrect way across pixels. Note that in (c), the area covered by the projection of the 3D object is smaller than a pixel, yet the dilated Gaussian is not attenuated, accumulating more light than what physically reaches the pixel. This leads to increased brightness and dilation artifacts which strongly degrade the appearance of the bicycle wheels' spokes.

The aforementioned scale ambiguity becomes particularly problematic in representations involving millions of Gaussians. However, simply discarding screen space dilation results in optimization challenges for complex scenes, such as those present in the Mip-NeRF 360 dataset [2], where a large number of small Gaussians are created by the density control mechanism [18], exceeding GPU capacity. Moreover, even if a model can be successfully trained without dilation, decreasing the sampling rate results in aliasing effects due to the lack of anti-aliasing [59].

5. Mip Gaussian Splatting → Modifications on 3DGS

To overcome these challenges, we make two modifications to the original 3DGS model. In particular, we introduce a 3D smoothing filter that limits the frequency of the 3D representation to below half the maximum sampling rate determined by the training images, eliminating high frequency artifacts when zooming in. Moreover, we demonstrate that replacing 2D screen space dilation with a 2D Mip filter, which approximates the box filter inherent to the physical imaging process, effectively mitigates aliasing and dilation issues. In combination, Mip-Splatting enables alias-free renderings[†] across various sampling rates. We now discuss the the 3D smoothing and the 2D Mip filters in detail.

5.1. 3D Smoothing Filter → tackle the problem of high-frequency artifacts

3D radiance field reconstruction from multi-view observations is a well-known ill-posed problem as multiple distinctly different reconstructions can result in the same 2D projections [2, 55, 56]. Our key insight is that the highest frequency of a reconstructed 3D scene is limited by

---

[*]The dilation operation is not mentioned in original paper.

[†]Note that we use alias to refer to multiple artifacts discussed in the paper, including dilation, erosion, oversmoothing, high-frequency artifacts and aliasing itself.

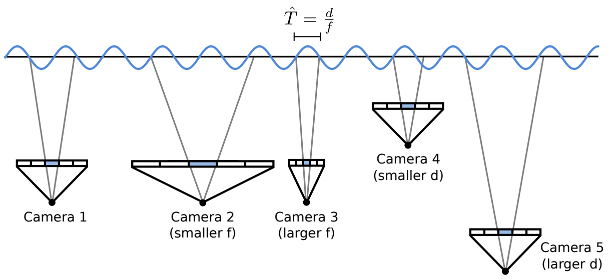

Figure 3. **Sampling limits.** A pixel corresponds to sampling interval $\hat{T}$. We band-limit the 3D Gaussians by the maximal sampling rate (i.e., minimal sampling interval) among all observations. This example shows 5 cameras at different depths d and with different focal lengths f. Here, camera 3 determines the minimal $\hat{T}$ and hence the maximal sampling rate $\hat{\nu}$.

the sampling rate defined by the training views. Following Nyquist's theorem 3.1, we aim to constrain the maximum frequency of the 3D representation during optimization.

Multiview Frequency Bounds: Multi-view images are 2D projections of a continuous 3D scene. The discrete image grid determines where we sample points from the continuous 3D signal. This sampling rate is intrinsically related to the image resolution, camera focal length, and the scene's distance from the camera. For an image with focal length f in pixel units, the sampling interval in screen space is 1. When this pixel interval is back-projected to the 3D world space, it results in a world space sampling interval $\hat{T}$ at a given depth d, with sampling frequency $\hat{\nu}$ as its inverse:

$$\hat{T} = \frac{1}{\hat{\nu}} = \frac{d}{f} \tag{6}$$

→ focal length

As posited by Nyquist's theorem Section 3.1, given samples drawn at frequency $\hat{\nu}$, reconstruction algorithms are able to reconstruct components of the signal with frequencies up to $\frac{\hat{\nu}}{2}$, or $\frac{f}{2d}$. Consequently, a primitive smaller than $2\hat{T}$ may result in aliasing artifacts during the splatting process, since its size is below twice the sampling interval.

primitive of 3DGS

To simplify, we approximate depth d using the center of the primitive $\mathbf{p}_k$ and disregard the impact of occlusion for sampling interval estimation. Since the sampling rate of a primitive is depth-dependent and differs across cameras, we determine the maximal sampling rate for primitive k as

$$\mathbb{1}_n(\mathbf{p}_k) = \begin{cases} 1, & \text{true} \\ 0, & \text{else} \end{cases}$$

$$\hat{\nu}_k = \max\left(\left\{\mathbb{1}_n(\mathbf{p}_k) \cdot \frac{f_n}{d_n}\right\}_{n=1}^{N}\right) \tag{7}$$

where N is the total number of images, $\mathbb{1}_n(\mathbf{p})$ is an indicator function that assesses the visibility of a primitive. It is true if the Gaussian center $\mathbf{p}_k$ falls within the view frustum of the n-th camera. Intuitively, we choose the sampling rate such that there exists at least one camera that is able to reconstruct the respective primitive. This process is illustrated in Figure 3 for $N = 5$. In our implementation,

Ensure at least one camera view from which the primitive can be adequately reconstructed

we recompute the maximal sampling rate of each Gaussian primitive every m iterations as we found the 3D Gaussians centers remain relatively stable throughout the training.

3D Smoothing: Given the maximal sampling rate $\hat{\nu}_k$ for a primitive, we aim to constrain the maximal frequency of the 3D representation. This is achieved by applying a Gaussian low-pass filter $\mathcal{G}_{\text{low}}$ to each 3D Gaussian primitive $\mathcal{G}_k$ before projecting it onto screen space:

$$\mathcal{G}_k(\mathbf{x})_{\text{reg}} = (\mathcal{G}_k \otimes \mathcal{G}_{\text{low}})(\mathbf{x}) \tag{8}$$

This operation is efficient as convolving two Gaussians with covariance matrices $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ results in another Gaussian with variance $\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2$. Hence,

$$\mathcal{G}_k(\mathbf{x})_{\text{reg}} = \boxed{\sqrt{\frac{|\mathbf{\Sigma}_k|}{|\mathbf{\Sigma}_k + \frac{s}{\hat{\nu}_k^2} \cdot \mathbf{I}|}}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p}_k)^T \left(\mathbf{\Sigma}_k + \frac{s}{\hat{\nu}_k^2} \cdot \mathbf{I}\right)^{-1} (\mathbf{x}-\mathbf{p}_k)} \tag{9}$$

Q6 Σ_k

Here, s is a scalar hyperparameter to control the size of the filter. Note that the scale $\frac{s}{\hat{\nu}_k}$ of the 3D filters for each primitive are different as they depend on the training views in which they are visible. By employing 3D Gaussian smoothing, we ensure that the highest frequency component of any Gaussian does not exceed half of its maximal sampling rate for at least one camera. Note that $\mathcal{G}_{\text{low}}$ becomes an intrinsic part of the 3D representation, remaining constant post-training.

5.2. 2D Mip Filter

→ Address aliasing issues that occurs when zooming out

While our 3D smoothing filter effectively mitigates high-frequency artifacts [18, 59], rendering the reconstructed scene at lower sampling rates (e.g., zooming out or moving the camera further away) would still lead to aliasing. To overcome this, we replace the screen space dilation filter of 3DGS by a 2D Mip filter.

More specifically, we replicate the physical imaging process [37, Section 8], where photons hitting a pixel on the camera sensor are integrated over the pixel's area. While an ideal model would use a 2D box filter in image space, we approximate it with a 2D Gaussian filter for efficiency

$$\mathcal{G}_k^{2D}(\mathbf{x})_{\text{mip}} = \boxed{\sqrt{\frac{|\mathbf{\Sigma}_k^{2D}|}{|\mathbf{\Sigma}_k^{2D} + s\mathbf{I}|}}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p}_k)^T \left(\mathbf{\Sigma}_k^{2D} + s\mathbf{I}\right)^{-1} (\mathbf{x}-\mathbf{p}_k)} \tag{10}$$

2D Gaussian

where s is chosen to cover a single pixel in screen space.

While our Mip filter shares similarities with the EWA filter [59], their underlying principles are distinct. Our Mip filter is designed to replicate the box filter in the imaging process, targeting an exact approximation of a single pixel. Conversely, the EWA filter's role is to limit the frequency signal's bandwidth, and the size of the filter is chosen empirically. The EWA paper [15, 59] even advocates for an identity covariance matrix, effectively occupying a 3x3 pixel region on the screen. However, this approach leads to overly

approximate the effect of a box filter

	PSNR ↑					SSIM ↑					LPIPS ↓				
	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res	Avg.
NeRF w/o $\mathcal{L}_{\text{area}}$ [1, 28]	31.20	30.65	26.25	22.53	27.66	0.950	0.956	0.930	0.871	0.927	0.055	0.034	0.043	0.075	0.052
NeRF [28]	29.90	32.13	33.40	29.47	31.23	0.938	0.959	0.973	0.962	0.958	0.074	0.040	0.024	0.039	0.044
MipNeRF [1]	32.63	34.34	35.47	35.60	34.51	0.958	0.970	0.979	0.983	0.973	0.047	0.026	0.017	0.012	0.026
Plenoxels [11]	31.60	32.85	30.26	26.63	30.34	0.956	0.967	0.961	0.936	0.955	0.052	0.032	0.045	0.077	0.051
TensoRF [4]	32.11	33.03	30.45	26.80	30.60	0.956	0.966	0.962	0.939	0.956	0.056	0.038	0.047	0.076	0.054
Instant-NGP [32]	30.00	32.15	33.31	29.35	31.20	0.939	0.961	0.974	0.963	0.959	0.079	0.043	0.026	0.040	0.047
Tri-MipRF [17]*	32.65	34.24	35.02	35.53	34.36	0.958	0.971	0.980	0.987	0.974	0.047	0.027	0.018	0.012	0.026
3DGS [18]	28.79	30.66	31.64	27.98	29.77	0.943	0.962	0.972	0.960	0.960	0.065	0.038	0.025	0.031	0.040
3DGS [18] + EWA [59]	31.54	33.26	33.78	33.48	33.01	0.961	0.973	0.979	0.983	0.974	0.043	0.026	0.021	0.019	0.027
Mip-Splatting (ours)	32.81	34.49	35.45	35.50	34.56	0.967	0.977	0.983	0.988	0.979	0.035	0.019	0.013	0.010	0.019

Table 1. **Multi-scale Training and Multi-scale Testing on the Blender dataset [28].** Our approach achieves state-of-the-art performance in most metrics. It significantly outperforms 3DGS [18] and 3DGS + EWA [59]. $*$ indicates that we retrain the model.

smooth results when zooming out as we will show in our experiments.

6. Experiments

We first present the implementation details of Mip-Splatting. We then assess its performance on the Blender dataset [28] and the challenging Mip-NeRF 360 dataset [2]. Finally, we discuss the limitations of our approach.

6.1. Implementation

We build our method upon the popular open-source 3DGS code base [18][‡]. Following [18], we train our models for 30K iterations across all scenes and use the same loss function, Gaussian density control strategy, schedule and hyper-parameters. For efficiency, we recompute the sampling rate of each 3D Gaussian every $m = 100$ iterations. We choose the variance of our 2D Mip filter as 0.1, approximating a single pixel, and the variance of our 3D smoothing filter as 0.2, totaling 0.3 for a fair comparison with 3DGS [18] and 3DGS + EWA [59] which replaces the dilation of 3DGS with the EWA filter.

hyper-parameter

6.2. Evaluation on the Blender Dataset

→ down-scale resolution

Multi-scale Training and Multi-scale Testing: Following previous work [1, 17], ~~we train~~ our model with multi-scale data ~~and evaluate~~ on multi-scale data. Similar to [1, 17] where rays of full resolution images are sampled more frequently compared to lower resolution images, we sample 40 percent of full resolution images and 20 percent from other image resolutions each. Our quantitative evaluation is shown in Table 1. Our approach attains comparable or superior performance compared to state-of-the-art methods such as Mip-NeRF [1] and Tri-MipRF [17]. Notably, our method outperforms 3DGS [18] and 3DGS + EWA [59] by a substantial margin, owing to its 2D Mip filter.

Only

Single-scale Training and Multi-scale Testing: Contrary to prior work that evaluates models trained on single-scale data at the same scale, we consider an important new setting that involves training on full-resolution images and rendering at various resolutions (i.e. 1×, 1/2, 1/4, and 1/8) to mimic zoom-out effects. In the absence of a public benchmark for this setting, we trained all baseline methods ourselves. We use NeRFAcc [23]'s implementation for NeRF [28], Instant-NGP [32], and TensoRF [4] for its efficiency. Official implementations were employed for Mip-NeRF [1], Tri-MipRF [17], and 3DGS [18]. The quantitative results, as presented in Table 2, indicate that our method significantly outperforms all existing state-of-the-art methods. A qualitative comparison is provided in Figure 4. Methods based on 3DGS [18] capture fine details more effectively than Mip-NeRF [1] and Tri-MipRF [17], but only at the original training scale. Notably, our method surpasses both 3DGS [18] and 3DGS + EWA [59] in rendering quality at lower resolutions. In particular, 3DGS [18] exhibits dilation artifacts. EWA splatting [59] uses a large low pass filter to limit the frequency of the rendered images, resulting in oversmoothed images, which becomes particularly apparent at lower resolutions.

the goal

Only

6.3. Evaluation on the Mip-NeRF 360 Dataset

→ up-scale resolution

Single-scale Training and Multi-scale Testing: To simulate zoom-in effects, we train models on data downsampled by a factor of 8 and rendered at successively higher resolutions (1×, 2×, 4×, and 8×). In the absence of a public benchmark for this setting, we trained all baseline methods ourselves. We use the official implementation for Mip-NeRF 360 [1] and 3DGS [18] and use a community reimplementation for Zip-NeRF [3][§] as the code is not available. The results in Table 3 show that our method performs comparable to prior work at the training scale (1×) and significantly exceeds all state-of-the-art methods at higher resolutions. As depicted in Figure 5, our method generates high fidelity imagery without high-frequency artifacts. Notably, both Mip-NeRF 360 [2] and Zip-NeRF [3] exhibit subpar performance at increased resolutions, likely due to their MLPs' inability to extrapolate to out-of-distribution frequencies. While 3DGS [18] introduces notable erosion

[‡]https://github.com/graphdeco-inria/gaussian-splatting

[§]https://github.com/SuLvXiangXin/zipnerf-pytorch

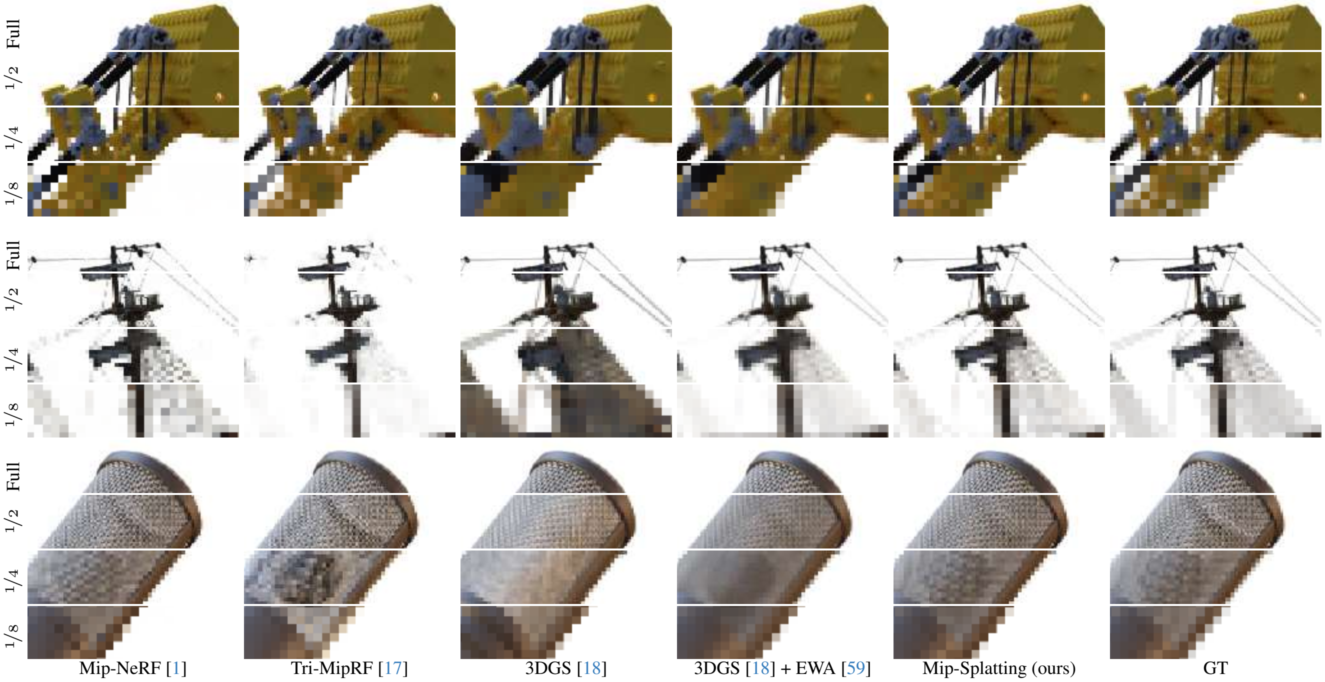

Figure 4. **Single-scale Training and Multi-scale Testing on the Blender Dataset [28].** All methods are trained at full resolution and evaluated at different (smaller) resolutions to mimic zoom-out. Methods based on 3DGS capture fine details better than Mip-NeRF [1] and Tri-MipRF [17] at training resolution. Mip-Splatting surpasses both 3DGS [18] and 3DGS + EWA [59] at lower resolutions.

	PSNR ↑					SSIM ↑					LPIPS ↓				
	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res	Avg.
NeRF [28]	31.48	32.43	30.29	26.70	30.23	0.949	0.962	0.964	0.951	0.956	0.061	0.041	0.044	0.067	0.053
MipNeRF [1]	33.08	33.31	30.91	27.97	31.31	0.961	0.970	0.969	0.961	0.965	0.045	0.031	0.036	0.052	0.041
TensoRF [4]	32.53	32.91	30.01	26.45	30.48	0.960	0.969	0.965	0.948	0.961	0.044	0.031	0.044	0.073	0.048
Instant-NGP [32]	33.09	33.00	29.84	26.33	30.57	0.962	0.969	0.964	0.947	0.961	0.044	0.033	0.046	0.075	0.049
Tri-MipRF [17]	32.89	32.84	28.29	23.87	29.47	0.958	0.967	0.951	0.913	0.947	0.046	0.033	0.046	0.075	0.050
3DGS [18]	33.33	26.95	21.38	17.69	24.84	0.969	0.949	0.875	0.766	0.890	0.030	0.032	0.066	0.121	0.063
3DGS [18] + EWA [59]	33.51	31.66	27.82	24.63	29.40	0.969	0.971	0.959	0.940	0.960	0.032	0.024	0.033	0.047	0.034
Mip-Splatting (ours)	33.36	34.00	31.85	28.67	31.97	0.969	0.977	0.978	0.973	0.974	0.031	0.019	0.019	0.026	0.024

Table 2. **Single-scale Training and Multi-scale Testing on the Blender Dataset [28].** All methods are trained on full-resolution images and evaluated at four different (smaller) resolutions, with lower resolutions simulating zoom-out effects. While Mip-Splatting yields comparable results at training resolution, it significantly surpasses previous work at all other scales.

EWA rendering

artifacts due to dilation operations, 3DGS + EWA [59] performs better while still yielding pronounced high-frequency artifacts. In contrast, our method avoids such artifacts, yielding aesthetically pleasing images that more closely resemble ground truth. It's important to remark that rendering at higher resolutions is a super-resolution task, and models should not hallucinate high-frequency details absent from the training data.

Single-scale Training and Same-scale Testing: We further evaluate our method on the Mip-NeRF 360 dataset [2] following the widely used setting, where models are trained and tested at the same scale, with indoor scenes downsampled by a factor of two and outdoor scenes by four. As shown in Table 4, our method performs on par with 3DGS [18] and 3DGS + EWA [59] in this challenging benchmark, without any decrease in performance. This confirms our method's effectiveness to handle various settings.

6.4. Limitations

Our method employs a Gaussian filter as an approximation to a box filter for efficiency. However, this approximation introduces errors, particularly when the Gaussian is small 1 in screen space. This issue correlates with our experimental findings, where increased zooming out leads to larger errors, as evidenced in Table 2. Additionally, there is a slight increase in training overhead as the sampling rate for each 3D Gaussian must be calculated every $m = 100$ iterations. Currently, this computation is performed using PyTorch [35] and a more efficient CUDA implementation could potentially reduce this overhead. Designing a better data structure for precomputing and storing the sampling rate, as it depends solely on the camera poses and intrin-

Figure 5. **Single-scale Training and Multi-scale Testing on the Mip-NeRF 360 Dataset [2].** All models are trained on images downsampled by a factor of eight and rendered at full resolution to demonstrate zoom-in/moving closer effects. In contrast to prior work, Mip-Splatting renders images that closely approximate ground truth. Please also note the high-frequency artifacts of 3DGS + EWA [59].

	PSNR ↑					SSIM ↑					LPIPS ↓				
	1× Res.	2× Res.	4× Res.	8× Res.	Avg.	1× Res.	2× Res.	4× Res.	8× Res.	Avg.	1× Res.	2× Res.	4× Res.	8× Res.	Avg.
Instant-NGP [32]	26.79	24.76	24.27	24.27	25.02	0.746	0.639	0.626	0.698	0.677	0.239	0.367	0.445	0.475	0.382
mip-NeRF 360 [2]	29.26	25.18	24.16	24.10	25.67	0.860	0.727	0.670	0.706	0.741	0.122	0.260	0.370	0.428	0.295
zip-NeRF [3]	29.66	23.27	20.87	20.27	23.52	0.875	0.696	0.565	0.559	0.674	0.097	0.257	0.421	0.494	0.318
3DGS [18]	29.19	23.50	20.71	19.59	23.25	0.880	0.740	0.619	0.619	0.715	0.107	0.243	0.394	0.476	0.305
3DGS [18] + EWA [59]	29.30	25.90	23.70	22.81	25.43	0.880	0.775	0.667	0.643	0.741	0.114	0.236	0.369	0.449	0.292
Mip-Splatting (ours)	29.39	27.39	26.47	26.22	27.37	0.884	0.808	0.754	0.765	0.803	0.108	0.205	0.305	0.392	0.252

Table 3. **Single-scale Training and Multi-scale Testing on the Mip-NeRF 360 Dataset [2].** All methods are trained on the smallest scale (1×) and evaluated across four scales (1×, 2×, 4×, and 8×), with evaluations at higher sampling rates simulating zoom-in effects. While our method yields comparable results at the training resolution, it significantly surpasses all previous work at all other scales.

	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [9, 28]	23.85	0.605	0.451
mip-NeRF [1]	24.04	0.616	0.441
NeRF++ [56]	25.11	0.676	0.375
Plenoxels [11]	23.08	0.626	0.463
Instant NGP [32, 52]	25.68	0.705	0.302
mip-NeRF 360 [2, 30]	27.57	0.793	0.234
Zip-NeRF [3]	28.54	0.828	0.189
3DGS [18]	27.21	0.815	0.214
3DGS [18]*	27.70	0.826	0.202
3DGS [18] + EWA [59]	27.77	0.826	0.206
Mip-Splatting (ours)	27.79	0.827	0.203

Table 4. **Single-scale Training and Same-scale Testing on the Mip-NeRF 360 dataset [2].** In the standard in-distribution setting, our approach demonstrates performance on par with many established techniques. $*$ indicates that we retrain the model.

sics, is an avenue for future work. As mentioned before, the sampling rate computation is the only prerequisite during training and the 3D smoothing filter can be fused with the Gaussian primitives per Eq. 9, thereby eliminating any additional overhead during rendering.

7. Conclusion

We introduced Mip-Splatting, a technique improving 3DGS with a 3D smoothing filter and a 2D Mip filter for alias-free rendering at any scale. Our 3D smoothing filter effectively limits the maximal frequency of Gaussian primitives to match the sampling constraints imposed by the training images, while the 2D Mip filter approximates the box filter to simulate the physical imaging process. Mip-Splatting significantly outperforms state-of-the-art methods in out-of-distribution scenarios, when testing at sampling rates different from training, resulting in better generalization to out-of-distribution camera poses and zoom factors.

Acknowledgement

ZY, AC and AG are supported by the ERC Starting Grant LEGO-3D (850533) and DFG EXC number 2064/1 - project number 390727645. TS is supported by a Czech Science Foundation (GACR) EXPRO grant (UNI-3D, grant no. 23-07973X). We also thank Christian Reiser for insightful discussions during the preparation of the draft.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021. 2, 3, 6, 7, 8

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2, 4, 6, 7, 8

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 2, 3, 6, 8

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. 2022. 1, 2, 6, 7

[5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. 2019. 2

[6] Zhang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, and Yi Xu. Neurbf: A neural fields representation with adaptive radial basis functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4182–4194, 2023. 2

[7] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, 1986. 3

[8] Franklin C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, page 207–212, New York, NY, USA, 1984. Association for Computing Machinery. 3

[9] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. 8

[10] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4): 65–74, 1988. 2

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1, 2, 6, 8

[12] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 453–464. 2023. 2

[13] Markus Gross and Hanspeter Pfister. *Point-based graphics*. Elsevier, 2011. 3

[14] Jeffrey P Grossman and William J Dally. Point sample rendering. In *Rendering Techniques' 98: Proceedings of the Eurographics Workshop in Vienna, Austria, June 29—July 1, 1998 9*, pages 181–192. Springer, 1998. 3

[15] Paul S Heckbert. Fundamentals of texture mapping and image warping. 1989. 3, 5

[16] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021. 2

[17] Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 2, 3, 6, 7

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2, 3, 4, 5, 6, 7, 8

[19] Jonas Kulhanek and Torsten Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. *arXiv preprint arXiv:2304.09987*, 2023. 2

[20] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021. 3

[21] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG)*, 9(3):245–261, 1990. 2

[22] Marc Levoy and Pat Hanrahan. Light field rendering. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 441–452. 2023. 2

[23] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18537–18546, 2023. 6

[24] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 1, 2

[25] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 2

[26] Nelson Max and Min Chen. Local and global illumination in the volume rendering integral. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2005. 2

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. 2019. 2

[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 *ECCV*, 2020. 1, 2, 6, 7, 8

[29] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P. Srinivasan, and Jonathan T. Barron. NeRF in the dark: High dynamic range view synthesis from noisy raw images. *CVPR*, 2022. 2

[30] Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ricardo Martin-Brualla, and Jonathan T. Barron. MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF, 2022. 8

[31] Klaus Mueller, Torsten Moller, J Edward Swan, Roger Crawfis, Naeem Shareef, and Roni Yagel. Splatting errors and antialiasing. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):178–191, 1998. 3

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 2, 6, 7, 8

[33] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 1928. 2, 3

[34] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. 2019. 2

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 7

[36] Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[37] Steve Hollasch Peter Shirley, Trevor David Black. Ray tracing in one weekend. https://raytracing.github.io/books/RayTracingInOneWeekend.html, 2023. 2, 5

[38] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000. 3

[39] Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. Dynamic point fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7964–7976, 2023. 3

[40] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. 2021. 2

[41] Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P. Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T. Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *SIGGRAPH*, 2023. 2

[42] Liu Ren, Hanspeter Pfister, and Matthias Zwicker. Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering. In *Computer Graphics Forum*, pages 461–470. Wiley Online Library, 2002. 4

[43] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *arXiv preprint arXiv:2110.06635*, 2021. 3

[44] Miguel Sainz and Renato Pajarola. Point-based rendering techniques. *Computers & Graphics*, 28(6):869–879, 2004. 3

[45] Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 1949. 2, 3

[46] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[47] J Edward Swan, Klaus Mueller, Torsten Moller, N Shareel, Roger Crawfis, and Roni Yagel. An anti-aliasing technique for splatting. In *Proceedings. Visualization'97 (Cat. No. 97CB36155)*, pages 197–204. IEEE, 1997. 3

[48] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022. 2

[49] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[50] Lance Williams. Pyramidal parametrics. page 1–11, New York, NY, USA, 1983. Association for Computing Machinery. 3

[51] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2

[52] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv*, 2023. 2, 8

[53] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 38(6), 2019. 3

[54] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 2

[55] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 4

[56] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 4, 8

[57] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J. Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[58] Yiyu Zhuang, Qi Zhang, Ying Feng, Hao Zhu, Yao Yao, Xiaoyu Li, Yan-Pei Cao, Ying Shan, and Xun Cao. Anti-aliased neural implicit surfaces with encoding level of detail. *arXiv preprint arXiv:2309.10336*, 2023. 3

[59] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 2, 3, 4, 5, 6, 7, 8

[60] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 3