EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Aliasing in 3D Reconstructions Mip-NeRF and Mip-Splatting

César Díaz Blanco

March 23, 2025

Seminar Winter 2024/2025
Graphics, Computer Vision and Machine Learning

Aliasing in 3D Reconstructions Mip-NeRF and Mip-Splatting

César Díaz Blanco

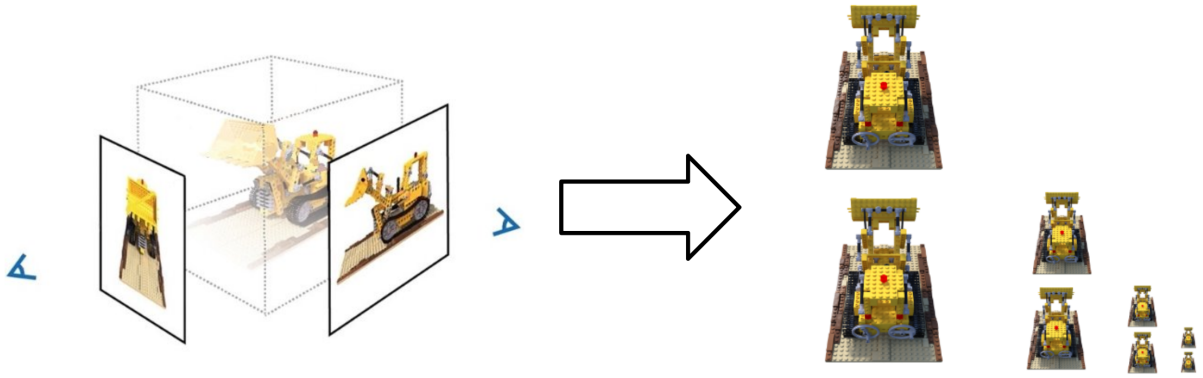

Figure 1: Mip-NeRF and Mip-Splatting synthesize novel views from a scene at different scales and focal lengths to those used during training.

Abstract

Novel view synthesis (NVS) aims to generate images from unseen camera poses. Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) have showed impressive results but struggle with aliasing when rendering views at scales or focal lengths different from the training set. NeRF exhibits blurring or aliasing under varying sampling rates, while 3DGS suffers from structural erosion or thickening. This report details how Mip-NeRF and Mip-Splatting address these limitations through anti-aliasing strategies grounded in signal processing and classical computer techniques. Mip-NeRF replaces NeRF's ray tracing with cone tracing reducing errors by 60% over NeRF while halving storage. Mip-Splatting introduces a low-pass filter to constrain the size of the 3D Gaussian primitives based on the maximal sampling frequency in the training set. Moreover, it replaces 3DGS' dilation on the projected gaussians with a low-pass filter which simulates a 2D box filter. Both methods outperform their predecessors in visual quality, with Mip-Splatting excelling in efficiency and robustness across scales.

1 Introduction

Novel view synthesis (NVS) is the task of synthesizing a target image with an arbitrary target camera pose from given source images and their camera poses. Recent progress has been successful with the aid of machine learning with neural radiance fields (NeRFs) [MST+20] and 3D Gaussian Splatting (3DGS) [KKLD23] at the forefront.

The results for both methods are remarkable when synthesizing novel views at the same distance or focal length as the training images since the sample rate is the same. However, when synthesizing views at different focal lengths or distances to the scene compared to the training images, these methods fail to produce faithful renderings. In the case of NeRF, the rendered images are either excessively blurred for higher sampling rates at closer views or aliased for lower sampling rates at distant views. For 3DGS, erosion of fine structures is present at higher sampling rates and thickening of fine structures at lower sampling rates.

Mip-NeRF [BMT+21] proposes to do volume rendering along cones and its sections, frustum or frusta in plural, instead of points along rays.

MipSplatting [YCH+24] proposes to enforce a scene-dependent-constraint on the 3D gaussians and a 2D box filter on the projected 2D gaussians. Mip-NeRF improves upon NeRF in single-scale settings and decidedly outperforms it in multi-scale settings reducing errors by 17% and 60% respectively relative to NeRF. Furthermore, its storage is half of its predecessor as it doesn't require two separate multilayer perceptrons (MLP) to represent coarse and fine details. Likewise, Mip-Splatting outperforms 3DGS in single-scale settings and multi-scale settings by reducing errors relative to its predecessor by 60% for the latter configuration. Both methods have similar rendering times as their predecessors: Mip-Splatting achieves real-time rendering while Mip-NeRF doesn't.

In this report, first in section 2, we will present the works introducing the underlying representations: NeRF and 3DGS. In section 3 we will briefly explain the theory behind aliasing. Then in sections 4 and 5, we will focus on the methods used by the authors of Mip-NeRF and MipSplatting to overcome their predecessors' sampling artifacts by re-framing classical approaches against aliasing. In section 6.3, we will highlight a new type of aliasing that emerges when sampling at higher rates than those shown during training. This phenomenon contrasts with classical computer graphics where aliasing is only present when sampling at lower rates. Finally, in section 6.2, we will compare Mip-NeRF and Mip-Splatting storage, training, and rendering time against their predecessors and compare Mip-NeRF, 3DGS, and Mip-Splatting on the basis of their visual quality, peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS).

2 Preliminaries

The process of synthesizing an image using a computer program is called rendering. Different techniques exist such as rasterization and ray-tracing. Ray-tracing is a general technique for modeling light transport and a specific algorithm inside this family is volume rendering. Volume rendering, as we will see in section 2.1, details a method to compute the mapping of scene geometry to pixels and their colors. On the other hand, rasterization refers only to the process of mapping scene geometry to pixels. In this sense, NeRF [MST+20] is totally reliant on the volume rendering algorithm as it renders an image by querying a MLP at points along the view-rays and integrating their colors. On the other hand, 3DGS [KKLD23] represents a scene as a point cloud of 3D gaussian primitives. The projection of these gaussians into 2D gaussians is similar to a rasterization process but it still relies on the volume rendering algorithm to compute the pixel's color. Both of these methods are explained in detail in sections 2.2 and 2.3.

2.1 Volume Rendering Equation

The theory behind both NeRF [MST+20] and 3DGS [KKLD23] is volume rendering. The core idea as in surface rendering, is to cast a camera ray to the scene and keep track of the ray hits which are then used to determine the pixel's color value. The volume rendering equation returns the expected color by integrating along the ray from t_0, the first point just outside the camera, to t_f, an arbitrary last point along the ray:

$$\int_{t_0}^{t_f} T(t)\sigma(t)\mathbf{c}(t)dt \tag{1}$$

Where $T(t)$ is the transmittance or probability of the ray not being absorbed yet from its origin to point t, $\sigma(t)$ is the volume density or probability of ray being absorbed at point t, and $\mathbf{c}(t)$ is the RGB color at point t.

The volume rendering equation is approximated by splitting the ray into n segments with endpoints $\{t_1, t_2, ..., t_{n+1}\}$ with constant lengths $\delta_i = t_{i+1} - t_i$. Density and color can be treated as constant in these segments but not transmittance since the segments are not necessarily aligned with surfaces. The approximation is as follows:

$$\int_{t_0}^{t_f} T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n} \left(\int_{t_i}^{t_{i+1}} T(t)dt \right) \sigma_i \mathbf{c}_i \tag{2}$$

To approximate the transmittance integral, it is useful to define the segment's opacity as $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ which indicates how much light is contributed by ray segment i. Thus, the transmittance up to segment i must be the product of previous segments *not* contributing to the integration:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{3}$$

Finally, in both NeRF and 3DGS, the volume rendering equation is approximated by summing over each segment i's product of: the probability T_i that no previous segments have absorbed the ray, the contribution α_i of this segment, and the RGB color $\mathbf{c}_i$ at this segment:

$$\sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i \tag{4}$$

2.2 NeRF

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis [MST+20] uses the weights of a MLP to infer the volume density at a given point and the color it emits at a given angle. Then, to synthesize a novel view, a pixel is rendered through the approximated volume rendering equation (4). The input to the MLP is a 5D coordinate with the first three coordinates corresponding to a point $\mathbf{x} = (x, y, z)$ along the ray segment i and the last two to the viewing direction $\mathbf{d} = (\theta, \phi)$. Its output is the corresponding volume density σ and the view-dependent RGB color $\mathbf{c}$:

$$F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma) \tag{5}$$

The authors of NeRF introduce a positional encoding to map continuous input coordinates into a higher dimensional space to enable the MLP to more easily approximate higher frequency functions. This is done by γ which maps the input 5D coordinates to a higher dimensional space $\mathbb{R}^{2L}$:

$$\gamma(\mathbf{x}) = \Big[\sin(\mathbf{x}), \cos(\mathbf{x}), \ldots, \sin\big(2^{L-1}\mathbf{x}\big), \cos\big(2^{L-1}\mathbf{x}\big) \Big]^{\mathrm{T}} \tag{6}$$

The input to NeRF is a set of images with the corresponding camera parameters: number of pixels in x and y dimension, focal length along x and y dimension, and camera position and rotation in world coordinates. During training, a ground truth image and its camera parameters are sampled from the training set, an image is rendered pixel by pixel with equation (4) by querying the MLP with the points along the rays produced according to the corresponding camera parameters, and the MLP is optimized by minimizing the total squared difference between the rendered and ground truth RGB values per pixel.

In practice, two MLPs, coarse and fine, are needed to do hierarchical volume sampling as free space and occluded regions that do not contribute to the rendered image are otherwise sampled repeatedly. The coarse MLP is sampled at $N_c = 64$ equally spaced points along the ray. From these, a piecewise-constant probability density function (PDF) with domain along the ray is produced with values at each segment as follows:

$$w_i = T_i \alpha_i \quad \hat{w}_i = \frac{w_i}{\sum_{j=1}^{N_c} w_j} \tag{7}$$

To prioritize regions with higher volume density $N_f = 128$ points are sampled from this distribution. Finally, the pixel will be rendered by querying the fine MLP at both the equally sampled N_c points and the hierarchical sampled N_f points.

2.3 3DGS

3D Gaussian Splatting for Real-Time Radiance Field Rendering [KKLD23] proposes to represent a scene as a pointcloud of 3D gaussians. These gaussians are projected along the plane perpendicular to the viewing axis into 2D gaussians in a rasterization step. Finally, to compute the color value at each pixel, the volume rendering equation (4) is computed by integrating the opacity and color of each gaussian from closest to farthest gaussians. The gaussians are defined in world space and have mean μ and 3D covariance matrix Σ:

$$G(x) \ = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{8}$$

Each gaussian also has spherical harmonics (SH) coefficients and opacity α as attributes to reproduce the scene's appearance. In computer graphics the set of Laplace's spherical harmonics is used to quickly encode or decode directional information; in the case of 3DGS, the color of each gaussian is encoded as the 16 coefficients of the 0th, 1st, 2nd, and 3rd bands of the Laplace's spherical harmonics functions.

In the rasterization step, the gaussians are transformed according to the view point defined by rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^3$ so that its camera coordinates are as follow:

$$\mu' = \mathbf{R}\mu + \mathbf{t}, \quad \Sigma' = \mathbf{R}\Sigma\mathbf{R}^\top \tag{9}$$

Recall that in standard camera coordinates the z-axis is the viewing axis, thus by discarding the last row and column of Σ' as well as the last entry of μ' the resulting 2D gaussian is defined in image space.

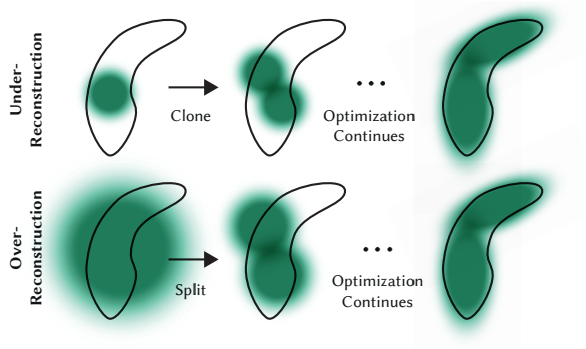

Figure 2: 3DGS' adaptive densification. Gaussians are cloned when geometry is insufficiently covered and split when geometry is represented by one large splat.

Once the rasterization step is finished, the gaussians are sorted in ascending order with respect to their depth which is the last entry of μ'. The color is computed with the approximated volume rendering equation (4) but instead of summing over segments i summing over the sorted gaussians. Finally, all gaussian parameters are optimized by minimizing a L1 and D-SSIM loss between the rendered and ground truth RGB values per pixel.

To ensure all geometries are well represented, new gaussians are created from gaussians with high positional gradients which signal areas that are either completely missing features, under-reconstruction, or would benefit from higher frequency details, over-reconstruction. The distinction between these two cases and the way 3DGS' adaptive densification scheme handles them is presented in Fig. 2 .

In practice, a few considerations must be taken into account for stable computation and memory efficiency. First, since covariance matrices must be symmetric and positive semi-definite, direct optimization of its entries is not trivial to setup. Instead, the authors of 3DGS propose to decompose the covariance matrix into a scaling and rotation matrix: $\Sigma = RSS^\top R^\top$. This is not enough since rotation matrices are also hard to optimize since they must satisfy that $RR^\top = \mathbb{1}$. Instead, the rotation is represented through a unit quaternion since keeping its norm as one is a constraint that is easy to enforce.

Since optimization is done with a graphics processing unit and its memory is limited, gaussians with opacity α close to zero are removed and projected 2D gaussians are dilated. The gaussians are dilated by increasing its size in all directions by $s = 0.3$ as follows:

$$G(x)_{2D} = e^{-\frac{1}{2}(x-\mu')^\top (\Sigma' + s\mathbb{1})^{-1}(x-\mu')} \tag{10}$$

This dilation is needed in complex scenes, where a large number of small gaussians are created by the adaptive density scheme. The dilation helps to represent details with the least gaussians possible and avoid redundancy.

3 Aliasing and mipmaps

Since novel view synthesis methods are ultimately trying to represent a signal, they are bound to Nyquist's Theorem [Nyq28]. This theorem describes the conditions under which a continuous signal can be accurately represented or reconstructed from its discrete samples without loss of information. It goes as follows:

Condition 1 *The sampling rate $\hat{\nu}$ must be at least twice the highest frequency present in the continuous signal: $\hat{\nu} \geq 2\nu$.*

In practice, to avoid introducing low frequency artifacts when reconstructing a signal at frequencies higher than twice the sampling rate, a low-pass filter is applied to the signal before sampling which smooths out any frequency components above $\hat{\nu}/2$.

In computer graphics, since the sampling rate constantly changes while navigating a scene, fast solutions are needed to avoid aliasing. One of the most widely used methods in rasterization is proposed by Lance Williams in [Wil83]. Williams proposes a "pyramidal parametric" prefiltering and sampling geometry on the texture images which minimizes aliasing effects and assures continuity within and between target images. The prefiltered images are called pyramids since they represent the signal at a progressively lower resolution: a factor of two smaller than the previous level. At rendering time, the pixel area covered by the texture is computed and the two nearest prefiltered images (lower and higher resolution) are used to compute an interpolated image which matches the texture's pixel area. For instance, if a texture uses 40×40 pixels, then the trilinear interpolation of the 64×64 and the 32×32 mipmaps is used for rendering. In the

original paper this interpolation between prefiltered images is refered as "mip" mapping; over time this term became synonym with aliasing solutions.

While the underlying representations studied in Pyramidal Parametrics, Mip-NeRF [BMT+21], and Mip-Splatting [YCH+24] are different and thus the approaches to these are different as well, the key idea behind them is the same as in signal processing: apply a low pass filter to the signal before sampling. When it comes to images, this can be understood as making sure that a pixel's color faithfully represents the area it covers in a scene. In practice, this could be achieved by re-framing the volume rendering equation to do cone tracing as shown in Fig. 3 and detailed in section 4 or by applying a box filter to the projected primitives as detailed in section 5.

4 Mip-NeRF

As seen in section 2.2, NeRF [MST+20] computes a pixel's color as the integration along a single infinitesimally narrow ray per pixel instead of all incoming radiance onto the pixel. This assumption results in aliased images when reconstructing the signal at frequencies different to those present in the training set. In section 4.1, we will explore how Mip-NeRF [BMT+21] ensures the pixel footprint is taken into account when computing equation (4). In sections 4.2 and 4.3, we will detail how the MLP input and hierarchical sampling change from its predecessor.

4.1 Cone tracing

Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields [BMT+21] proposes to compute a pixel's color by casting a cone as seen in Fig. 3b. The cone apex lies at the camera center $\mathbf{o}$ and since it should cover a pixel, its radius r at the image plane is dependent on the pixel width in world coordinates w_p; specifically $r = 2w_p/\sqrt{12}$.

Since Mip-NeRF also relies on the volume rendering equation, the next step is to split the cone along planes perpendicular to the cone direction; these segments are called frustum or frusta in plural. A naïve approach would query the MLP at points inside the frusta and average the inferred densities and colors. This approach would fail to be efficient as each segment would require many more queries to the MLP compared to the single query

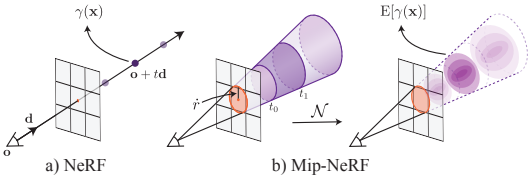

Figure 3: NeRF (a) casts rays and encodes the position of the sampled points along it. Mip-NeRF (b) casts cones and encodes a gaussian fitted to the frusta along it.

for a point along the ray in NeRF [MST+20]. The authors of Mip-NeRF instead approach this in two steps: fit the frusta to gaussians and take the expected positional encoding of the fitted gaussians as the MLP input. We explain the first step in the next paragraph and the second step in section 4.2.

To define the gaussian fitted to the cone it is helpful to think about the ray pointing in the direction of the cone; this is the same ray used in NeRF and the volume rendering equation. Recall that in section 2.1 points t_i are sampled along the ray; in cone tracing these determine the frusta's half length along the ray $t_\delta = (t_{i+1} - t_i)/2$ and the frusta's center along the ray $t_\mu = (t_{i+1} + t_i)/2$. Then, the fitted gaussians have mean depth μ_t along the ray, variance along the ray σ_t^2, and variance perpendicular to the ray σ_r^2:

$$\mu_t = t_\mu + \frac{2t_\mu t_\delta^2}{3t_\mu^2 + t_\delta^2}, \qquad \sigma_t^2 = \frac{t_\delta^2}{3} - \frac{4t_\delta^4(12t_\mu^2 - t_\delta^2)}{15(3t_\mu^2 + t_\delta^2)^2},$$
$$\sigma_r^2 = r^2\left(\frac{t_\mu^2}{4} + \frac{5t_\delta^2}{12} - \frac{4t_\delta^4}{15(3t_\mu^2 + t_\delta^2)}\right) \tag{11}$$

Recall that $r = 2w_p/\sqrt{12}$ is the cone radius at the image plane. Note that there is only one perpendicular variance since the cone is segmented along the plane perpendicular to the cone direction such that the cuts are circular. The previously defined gaussian is still in the cone coordinate system, to transform them to world coordinates the following transformation is done:

$$\boldsymbol{\mu} = \mathbf{o} + \mu_t \mathbf{d}, \qquad \boldsymbol{\Sigma} = \sigma_t^2\left(\mathbf{d}\mathbf{d}^{\mathrm{T}}\right) + \sigma_r^2\left(\mathbb{1} - \frac{\mathbf{d}\mathbf{d}^{\mathrm{T}}}{||\mathbf{d}||_2^2}\right) \tag{12}$$

Where $\mathbf{o}$ is the cone apex which lies at the camera center and $\mathbf{d}$ is the cone ray direction.

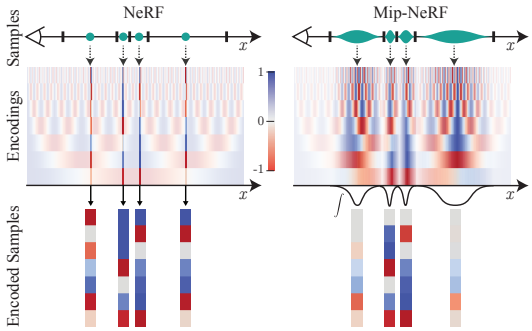

Figure 4: NeRF encodes all frequencies at the same amplitude while Mip-NeRF encodes frequencies based on the gaussian's variance.

4.2 Integrated positional encoding (IPE)

Similar to its predecessor, Mip-NeRF [BMT+21] uses a MLP to infer density and color. This means that it also needs to deal with the fact that MLPs learn a smooth function which cannot represent high frequency signals. Thus, Mip-NeRF also encodes the input to the MLP into a higher dimensional space: $\gamma(\mathbf{x})$ from equation (6). This time, however, a gaussian needs to be encoded rather than a point. To do so, the authors of Mip-NeRF rewrite the encoding $\gamma(\mathbf{x})$ in matrix form:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 0 & & 2^{L-1} & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & \cdots & 0 & 2^{L-1} & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & & 0 & 0 & 2^{L-1} \end{bmatrix}^{\mathrm{T}}, \quad \gamma(\mathbf{x}) = \begin{bmatrix} \sin(\mathbf{Px}) \\ \cos(\mathbf{Px}) \end{bmatrix} \tag{13}$$

Then, the choice to approximate the frusta as gaussians pays off as linear transformations of gaussians have a closed form. Its parameters are:

$$\boldsymbol{\mu}_\gamma = \mathbf{P}\boldsymbol{\mu}, \qquad \boldsymbol{\Sigma}_\gamma = \mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^\top \tag{14}$$

Then, since the original goal was to avoid sampling points from the frusta or gaussian, Mip-NeRF instead computes the expected positional encoding of the fitted gaussians and takes this as the input to the MLP. Again, the choice of gaussians is favorable as the expectations of the sine and cosine functions over a gaussian have closed forms:

$$\mathbb{E}_{x\sim\mathcal{N}(\mu,\sigma^2)}\left[\sin(x)\right] = \sin(\mu)\exp\left(-(\sigma^2/2)\right), \tag{15}$$

$$\mathbb{E}_{x\sim\mathcal{N}(\mu,\sigma^2)}\left[\cos(x)\right] = \cos(\mu)\exp\left(-(\sigma^2/2)\right) \tag{16}$$

A quick inspection of the input to the MLP, equations (15) and (16), already hints at the effectiveness of Mip-NeRF. Pixels which cover a lot of space will have big frusta and thus be approximated through gaussians with a high variance. The expectation value for these gaussians will be drastically reduced at high frequencies due to the exponential term. On the other hand, pixels which cover little space will be fitted with small gaussians whose expectation values could be similar for all frequencies. Fig. 4 shows the difference between NeRF's positional encoding and Mip-NeRF's integrated positional encoding for one-dimensional signals. This difference in the input enables the MLP to represent multiple scales.

4.3 MLP

Recall that in NeRF [MST+20] a separate MLP is needed to coarsely represent the scene and do hierarchical volume sampling. In Mip-NeRF [BMT+21], there is no need for a coarse MLP because the input already allows the network to reason about different scales. Compared to NeRF this represents a 50% storage reduction.

In Mip-NeRF, hierarchical sampling starts with querying the MLP with N_c gaussians fitted to equally spaced frusta of the same length along the traced cone. To construct a probability density function along the cone ray as done in NeRF and explained in section 2.2, the weights across the frusta's length are computed as $w_i = T_i\alpha_i$. Except this time, the weights are further modified to produce a non-zero, wide, and smooth upper envelope on the distribution:

$$w'_k = \frac{1}{2}\left(\max(w_{k-1}, w_k) + \max(w_k, w_{k+1})\right) + 0.01 \tag{17}$$

After normalization of these weights, N_f points are sampled from this distribution which determine the planes between the frusta used for computing the pixel's color. For fair comparison, N_c and N_f are set to 128 as to match the total number of MLP evaluations in NeRF. Finally, as in NeRF, the MLP is optimized by minimizing the total squared difference between the rendered and ground truth RGB values per pixel.

5 Mip-Splatting

To avoid subpixel 2D gaussian instances, 3DGS [KKLD23] applies a dilation operation onto all projected gaussians as seen in section 2.3. This effectively widens the gaussians' signal and ultimately

Figure 5: Left: Ground truth image of bike seat. Right: High frequency artifacts with 3DGS when sampling at higher rate.

stops the adaptive densification scheme from creating a large number of small gaussians and exceeding GPU capacity. While this method works as it still minimizes the loss against the training set, it fails to accurately represent the scene when rendering images at different distances or focal lengths to those seen during training as seen in Figs. 5 and 6.

The reason behind the erosion artifacts present when rendering images at higher sampling rates is that during optimization, highly anisotropic 3D gaussians are not penalized as their projection is dilated and ultimately produce a faithful rendering. However, when images are rendered at closer distances, the *fixed* dilation does not widen the projected Gaussians' signal proportionally to the camera's proximity. As a result, the Gaussians appear narrower, as shown in Fig. 5. In section 5.1 we will detail how Mip-Splatting [YCH$^+$24] solves this by applying a 3D smoothing operation to each gaussian.

When it comes to lower sampling rates, 3DGS has two shortcomings. First, as in NeRF: it assumes that a pixel's color can be computed from a single ray as it follows this principle when projecting the 3D gaussians into image space. Second, when images are rendered at farther distances, the *fixed* dilation does not widen the projected Gaussians' signal proportionally to the camera's distance. As a result, the Gaussians appear thicker, as shown in Fig. 6. In section 5.2 we present Mip-Splatting's method which replicates the physical imaging process where a pixel's color represents the integrated photons hitting a sensor.

5.1 3D Smoothing Filter

Recall from Nyquist's Theorem [Nyq28] that the sampling rate depends on the highest frequency present in the signal. In novel view synthesis the highest frequency of the reconstructed scene is determined by the training images. Next, for a pinhole camera model with focal length f in pixels pointing at objects at a depth d, the sampling rate ν is given as f/d. If the depth d is given in meters then this matches the general understanding of sampling rate: how many samples (in this case pixels) are taken from the original signal (in this case per meter).

Thus, the key idea of Mip-Splatting [YCH$^+$24] is to compute the maximal sample rate $\hat{\nu}_k$ of gaussian k and regularize its covariance matrix to smooth out any frequency components above $\hat{\nu}_k/2$. In Mip-Splatting, to get the maximal sampling rate $\hat{\nu}_k$ for gaussian k they compute the N possible sampling rates from N different training camera viewpoints making sure to discard instances where the gaussian is outside of the view frustum with indicator function $\mathbb{1}_n$. Then they take the maximum:

$$\hat{\nu}_k = \max\left(\left\{\mathbb{1}_n(G_k)\cdot\frac{f_n}{d_n}\right\}_{n=1}^{N}\right) \tag{18}$$

During training, the gaussians' mean position and thus depth will change. However, the authors of Mip-Splatting observe that it doesn't change much between iterations so that the maximal rate for each gaussian is updated every $m = 100$ iterations.

Figure 6: Left: Ground truth image of bike. Right: Dilation and brightening artifacts with 3DGS when sampling at lower rate.

To constrain the maximal frequency of the 3D representation, a Gaussian low-pass filter with covariance matrix $sI/\hat{\nu}_k$ is applied to each 3D gaussian by convolving the two gaussians. The choice of a gaussian low-pass filter is adequate since the convolution of two Gaussians has a closed form. The regularized gaussian is:

$$G(x)_{\text{reg}} = \sqrt{\frac{|\Sigma|}{|\Sigma + \frac{sI}{\hat{\nu}_k}|}}\, e^{-\frac{1}{2}(x-\mu)^\top (\Sigma + \frac{s\mathbb{1}}{\hat{\nu}_k})^{-1}(x-\mu)} \tag{19}$$

Where the mean μ and covariance Σ correspond to the kth gaussian and $s = 0.2$ is a hyperparameter to change the size of the filter. This achieves the same purpose as the dilation in 3DGS [KKLD23]: stop the redundancy of gaussians by limiting their size, but with a first principles approach following Nyquist's theorem [Nyq28]. Once training is finished, the regularized gaussian may be saved as the final representation to eliminate additional computation during rendering.

5.2 2D Mip Filter

Mip-Splatting [YCH+24] seeks to replicate the physical imaging process where a pixel's color represents the integrated photons hitting a sensor. While a classical approach would do this operation in image space, the authors of Mip-Splatting leverage the projected 2D gaussians and apply the filter to these 2D primitives for efficiency given its closed form:

$$G(x)_{\text{mip}} = \sqrt{\frac{|\Sigma_{2D}|}{|\Sigma_{2D} + s\mathbb{1}}}\, e^{-\frac{1}{2}(x-\mu)^\top (\Sigma_{2D} + s\mathbb{1})^{-1}(x-\mu)} \tag{20}$$

Where $s = 0.1$ is chosen to cover a single pixel in screen space. Thus the total variance of the mip filter and the 3D smoothing operation add to 0.3 which is is the variance used for the dilation operation in 3DGS [KKLD23].

The key difference to the dilation operation is that in 3DGS the dilation affected only the covariance and there was no prefactor adjustment for renormalization, thus the aliased results show a dilation and brightening show in Fig. 6. On the other hand, Mip-Splatting does account for normalization of the convolved gaussians.

6 Comparison

We already saw the algorithmic differences between Mip-NeRF [BMT+21] and Mip-Splatting [YCH+24] against their predecessors in sections 4 and 5. Following we will explore: the differences in storage, training and rendering time in section 6.1, improvements in visual quality in section 6.2 and aliasing at higher sampling rates in section 6.3.

6.1 Storage size, training, and rendering time

Recall from section 4.3 that Mip-NeRF [BMT+21] uses one MLP compared to NeRF's [MST+20] coarse and fine MLPs. Due to this Mip-NeRF uses half as much storage as NeRF. In Mip-Splatting [YCH+24], since the regularized gaussians from section 5.1 are saved as the final representation, there is no change in storage size.

Again, since Mip-NeRF queries a single MLP compared to NeRF's querying of coarse and fine MLPs, Mip-NeRF is 7% faster than its predecessor during training and rendering. Still training times are slow at around 48 hours per scene in the follow-up paper Mip-NeRF360 [BMV+22]. In Mip-Splatting during training, the maximal sampling rate for each gaussian is calculated every 100 iterations. However, this only represents a "slight increase in training overhead" according to the authors. Taking into consideration that training times in 3DGS [KKLD23] are around 40 minutes per scene, the maximal sampling rate computation overhead shouldn't be significant. Both of the training times are when using an A100 GPU.

During rendering, 3DGS does a dilation operation by changing the gaussians' diagonal covariance entries. Mip-Splatting does a low-filter operation which also modifies the gaussians' diagonal covariance. Thus they are both similar in their runtime complexity and both achieve real-time rendering: more than 30 rendered images per second. On the other hand Mip-NeRF360 takes around 10 seconds per rendered image when rendering with an A100 GPU.

Since performance is similar for both Mip-NeRF and Mip-Splatting when compared to their predecessors, if the reader was to choose any of these representations they should think about whether the scene to represent benefits more from a ray-tracing method or a rasterization method. Ray-tracing, and thus NeRF-like methods, are able to handle

scenes including reflections, refractions, translucency, specularity or any other effects where light is not absorbed. On the other hand, rasterization and thus 3DGS-like methods have a harder time approximating these effects but they are a lot faster since rendering time only depends on the number of primitives in view.

6.2 Side-by-side comparisons

We take the quantitative results from the Mip-Splatting [YCH+24] paper which includes evaluations on the Blender dataset for single scale training and multi-scale testing. Fig. 7 shows that erosion is common at lower resolutions for fine structures in Mip-NeRF [BMT+21]. Table 1 shows that both Mip-NeRF and Mip-Splatting outperform their predecessors and that Mip-Splatting outperforms all methods in this report.

6.3 Aliasing at higher sampling rates

As seen in sections 4.1 and 5.2, we see that their solutions for aliasing at lower sampling rates is similar to those in classical computer graphics. The idea is to replicate the physical imaging process and integrate the rays hitting the pixel's sensor to compute its color. In the case of Mip-NeRF [BMT+21] this is done by tracing over cones with radius relative to the pixel width and in Mip-Splatting [YCH+24] by applying a low-pass gaussian filter on the projected gaussians.

In classical computer graphics the optimal setup is a sampling rate $\hat{\nu}$ twice the highest frequency: $\hat{\nu} \geq 2\nu$ as the reconstructed signal will simply repeat the texture's pixels and represent the signal as it was recorded. However for NeRF [MST+20] and 3DGS [KKLD23], since their underlying representations are removed from the sampling (imaging) process, artifacts may emerge.

In the case of NeRF sampling at higher rates when rendering new views leads to blurry artifacts since its MLP is queried at individual points which may have never been directly optimized during training. Mip-NeRF solves this by optimizing the MLP over volumes thus initializing more regions which are then partially queried when rendering closer views to the scene. In the case of 3DGS, due to memory issues during optimization, gaussians need to have a lower limit to their size. Mip-Splatting rightly sets this limit according to their maximal sampling rate as measured from the training camera viewpoints.

7 Conclusion

In this report, we examined the shortcomings of NeRF [MST+20] and 3D Gaussian Splatting [KKLD23] when rendering at scales or focal lengths different from those seen during training. We revisited their foundations, and analyzed how Mip-NeRF [BMT+21] and Mip-Splatting [YCH+24] improve on their methods to stop aliasing.

Mip-NeRF mitigates aliasing by replacing traditional ray sampling with cone tracing. It fits gaussians to the cone's frusta, encodes these gaussians into a higher dimensional space to represent high-frequency details, and queries an MLP at their expected value to get the frusta's density and color. As a result, the MLP reasons about the size and shape of each conical frustum instead of just its centroid.

Mip-Splatting enhances the 3D Gaussian representation by introducing a scene-dependent 3D smoothing filter to limit the size of the gaussians and ensure they don't represent frequencies higher than those seen in the training set. Furthermore, it enhances the rasterization step by applying a gaussian low-pass filter to the projected gaussians effectively replicating the physical imaging process of several rays hitting a sensor.

Our comparative analysis shows that both methods yield substantial improvements over their predecessors, with Mip-Splatting standing out in this report for its superior visual quality, as well as its training and rendering efficiency.

Future work in Mip-Splatting, as the authors suggest, could explore faster ways to compute the per-gaussian 3D smoothing filter. In the case of Mip-NeRF, cone tracing is a solid argument for stopping aliasing at lower sampling rates. However, for higher sampling rates, a more principled approach as in Mip-Splatting which grounds its 3D smoothing filter in Nyquist's theorem [Nyq28] could be illuminating.

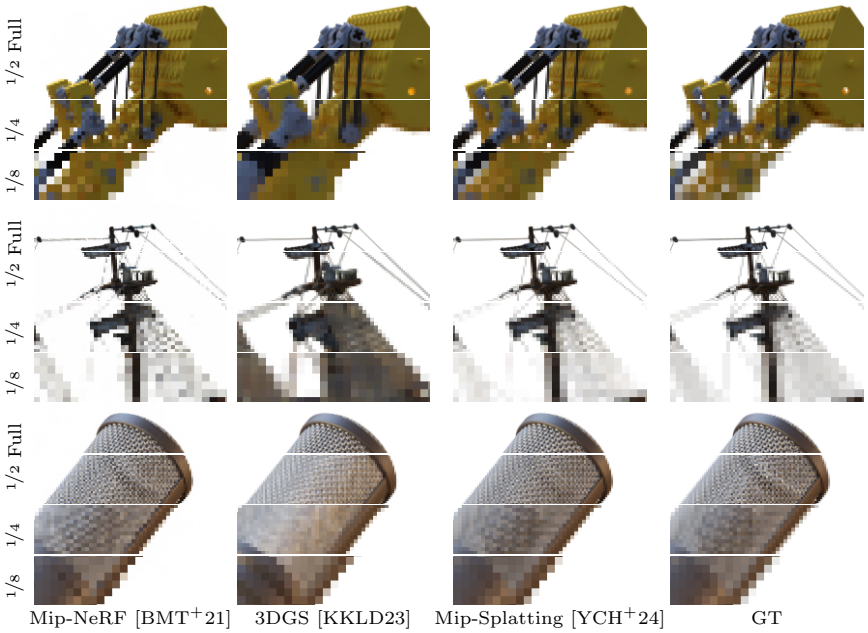

Figure 7: **Single-scale Training and Multi-scale Testing on the Blender Dataset [MST+20].** All methods are trained at full resolution and evaluated at different (smaller) resolutions to mimic zoom-out. Methods based on 3DGS capture fine details better than Mip-NeRF. Mip-Splatting surpasses 3DGS [KKLD23] 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 [MST+20]	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 [BMT+21]	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
3DGS [KKLD23]	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
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 1: **Single-scale Training and Multi-scale Testing on the Blender Dataset [MST+20].** 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.

References

[BMT+21] 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.

[BMV+22] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

[KKLD23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[MST+20] 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.

[Nyq28] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 1928.

[Wil83] Lance Williams. Pyramidal parametrics. In *Proceedings of the 10th An-*

nual Conference on Computer Graphics and Interactive Techniques, pages 1–11. Association for Computing Machinery, 1983.

[YCH+24] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.