AAA-Gaussians: Anti-Aliased and Artifact-Free 3D Gaussian Rendering

Michael Steiner*[1] Thomas Köhler*[1] Lukas Radl[1]
Felix Windisch[1] Dieter Schmalstieg[1,2] Markus Steinberger[1,3]
[1]Graz University of Technology [2]University of Stuttgart [3]Huawei Technologies
[1]{michael.steiner, t.koehler, lukas.radl, felix.windisch, steinberger}@tugraz.at
[2]dieter.schmalstieg@visus.uni-stuttgart.de

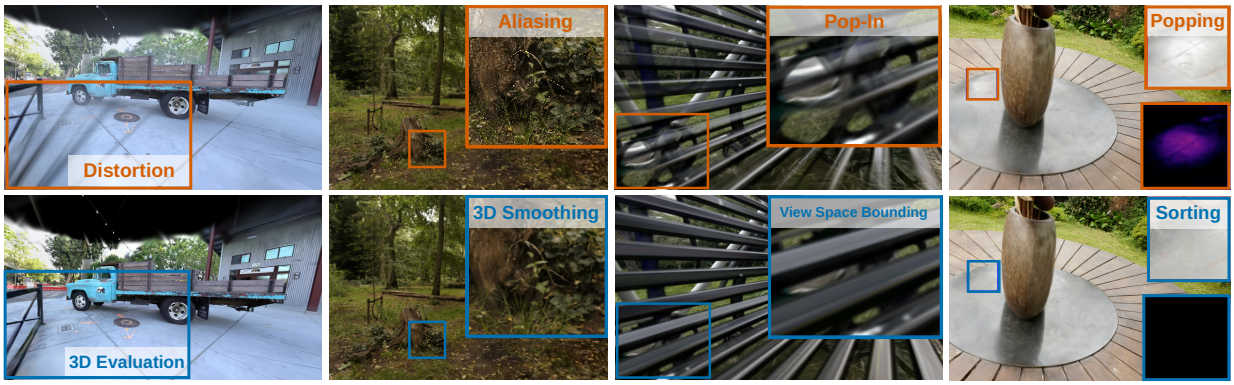

Figure 1. (Top row) 3DGS rasterization approaches encounter artifacts in out-of-distribution camera settings: (1) Distortions from 2D splat approximations in large field-of-view renderings. (2) 3D evaluation specific aliasing artifacts when zooming out. (3) Incorrect culling results in screen space when the camera is close to objects. (4) Popping due to depth simplifications and global sorting. (Bottom row) Our method addresses these issues with: (1) 3D Gaussian evaluation, (2) a correct aliasing filter, adapted specifically to Gaussian evaluation in 3D, (3) accurate and robust bounding, and (4) efficient 3D culling integrated into hierarchical sorting.

Abstract

Although 3D Gaussian Splatting (3DGS) has revolutionized 3D reconstruction, it still faces challenges such as aliasing, projection artifacts, and view inconsistencies, primarily due to the simplification of treating splats as 2D entities. We argue that incorporating full 3D evaluation of Gaussians throughout the 3DGS pipeline can effectively address these issues while preserving rasterization efficiency. Specifically, we introduce an adaptive 3D smoothing filter to mitigate aliasing and present a stable view-space bounding method that eliminates popping artifacts when Gaussians extend beyond the view frustum. Furthermore, we promote tile-based culling to 3D with screen-space planes, accelerating rendering and reducing sorting costs for hierarchical rasterization. Our method achieves state-of-the-art quality on in-distribution evaluation sets and significantly outperforms other approaches for out-of-distribution views. Our qualitative evaluations further demonstrate the effective removal of aliasing, distortions, and popping artifacts, ensuring real-time, artifact-free rendering.

*Both authors contributed equally to this work

1. Introduction

3D Gaussian Splatting (3DGS) [14] has recently revolutionized inverse rendering by enabling fast, differentiable rasterization of 3D Gaussian point clouds. While 2D splat evaluation is highly efficient, the projection from 3D to 2D Gaussians remains an approximation, introducing artifacts that become particularly noticeable under non-standard camera settings, such as a wide field-of-view in virtual reality rendering. Additionally, 3DGS further approximates the rendering of 2D splats by assuming them to be parallel to the current view-plane, leading to blend order inconsistencies and popping artifacts under simple camera rotations [28]. A natural solution to overcome these limitations is to render 3D Gaussians via ray tracing. However, this approach introduces significant computational overhead, requires additional acceleration structures, and is generally impractical for training due to the cost of frequent data structure updates [23].

Several works attempt to bridge the gap between 2D splatting and 3D ray tracing by first bounding 3D Gaussians in screen space and then computing their contributions per ray in 3D [11, 28, 32, 34, 37]. While these hybrid 2D/3D approaches address certain limitations of 3DGS, their re-

liance on screen space computations still makes them prone to artifacts, particularly when rendering out-of-distribution camera poses—*i.e*., viewpoints or parameters significantly different from the training data (*cf*. Fig. 1): (1) Even when considering the highest contribution point in 3D [28, 37] or adjusting transformations per Gaussian [34], distortions may still occur. (2) Zooming in or out beyond the typical training views introduces artifacts, especially when evaluating Gaussians in 3D, as 2D anti-aliasing techniques can no longer be applied [36]. (3) Despite the use of 3D bounding planes, screen space computations become unstable when Gaussians extend behind the image plane, leading to artifacts at image boundaries [11]. (4) Simplified per-pixel sorting strategies that focus on high-opacity Gaussians, can result in inaccurate renderings, particularly for viewpoints far from the training distribution [11].

We address these shortcomings in our 3D Gaussian rasterizer, which considers the 3D nature of Gaussians through all steps of the 3DGS rendering pipeline, making the following contributions:

- We start by analyzing previous anti-aliasing approaches and introduce an adaptive 3D smoothing filter that accurately dilates 3D Gaussians and removes aliasing artifacts, especially for out-of-distribution views.
- We show how bounding of 3D Gaussians can be moved from screen space to view space for stable bounding of Gaussians that reach outside the view frustum, removing disturbing popping artifacts on image boundaries.
- Elevating previous 2D tile-based culling algorithms to 3D by performing frustum-based culling with screen space planes, thereby accelerating rendering and reduce sorting costs for depth-sorted hierarchical rasterization [28].
- A detailed analysis of common 3D Gaussian rendering artifacts, and ablation of our employed components.

Overall, our method achieves state-of-the-art reconstruction quality, while enabling artifact-free rendering in real-time. Our training and rendering source code is publicly available at https://github.com/DerThomy/AAA-Gaussians.

2. Related Work

In this section, we cover recent radiance field and 3D Gaussian Splatting methods, with a focus on artifact-free rendering of 3D Gaussian representations.

2.1. Radiance Fields & 3D Gaussian Splatting

Radiance fields have attracted widespread interest in novel view synthesis since the publication of Neural Radiance Fields (NeRF) [22], which use large coordinate-based MLPs to model view-dependent color and density. Follow-up work includes improvements in terms of anti-aliasing [1, 3], extending NeRFs to unbounded scenes [2, 26], more efficient encodings [25], or fast rendering [7, 10, 31]. Despite the aforementioned improvements, most NeRF methods still require multiple costly MLP evaluations for each pixel, manifesting in long training and rendering times.

More recently, 3D Gaussian Splatting (3DGS) [14] exploded in popularity, replacing the slower implicit representations of NeRFs with an explicit 3D Gaussian point cloud representation that can be efficiently rasterized through elliptical weighted average (EWA) splatting [39]. The initial sparse point cloud can either be initialized randomly, from Structure-from-Motion [29], or from other guidances (*e.g*., a pretrained NeRF [27]). To achieve good scene coverage, this point cloud has to be densified by adding, cloning, or pruning points based on screen space gradients [14], per-view saliency maps [20], depth supervision [15], or by relocating low-opacity Gaussians [16]. Additionally, strategies such as opacity decay [28] and iterative pruning [8, 9] are employed to reduce the size of the resulting point cloud.

2.2. Artifacts in 3D Gaussian Splatting

The original 3DGS suffers from a number of artifacts (underlined), which have been addressed in recent related work. Aliasing artifacts during rasterization can be solved by approximating the integral over each pixel [18], using multi-scale 3D Gaussians [35], or by applying a smoothing filter to both the 3D Gaussian and its projected 2D splat [36]. Popping artifacts occur due to the global sort of primitives before rasterization, leading to sudden changes in the blending order. StopThePop [28] resolves this by computing Gaussian depth values along each view ray and using hierarchical k-buffers to improve the sort order. Other recent work relies on hybrid transparency [21], where only important contributors are sorted, while low-opacity Gaussians are blended in a "tail" [11]. Even perfect per-pixel sorting exhibits blending artifacts, due to the non-overlapping assumption of primitives, however, this cannot be solved analytically for Gaussians [19] and requires expensive volumetric integration [5, 6]. Furthermore, projection artifacts occur due to the affine approximation when projecting 3D Gaussians to 2D splats, resulting in disturbing cloud-like artifacts and elongated Gaussians, especially at the image border and for large field of view settings (*e.g*. in virtual reality [33]). Recent methods solve this by projecting Gaussians onto the unit sphere tangent plane [13] or using Unscented Transform [34]. Another way to avoid projection artifacts is to evaluate Gaussians in 3D. Several methods use ray tracing for this purpose [4, 19, 24], however, these methods are computationally expensive and require additional acceleration data structures, as well as specialized ray tracing hardware to achieve competitive performance. To circumvent this, several methods propose to bound the Gaussian on screen, and then evaluate it in 3D by finding its point of maximum contribution along the ray. However, these methods rely on inaccurate 2D bounds from the approximate affine projection [28, 32, 37] or Unscented Transform [34]. Only recently,

Hahlbohm *et al*. [11] computed accurate 2D bounds through plane fitting to the 3D Gaussian ellipsoid [30].

Despite the large body of work addressing artifact-free 3D Gaussians rendering, many of the aforementioned works imply significant performance penalties, still exhibit errors in extreme configurations or focus only on a single artifact. To the best of our knowledge, our method is the first to tackle all artifacts in a single, unified framework.

3. Method

This section covers necessary preliminary knowledge, and the components of our artifact-free 3D Gaussian renderer. Our rasterizer utilizes hierarchical sorting [28] and 3D Gaussian evaluation through screen space planes [11] to remove popping and projection artifacts. Sec. 3.2 introduces our novel adaptive 3D filter that removes aliasing artifacts; Sec. 3.3 contains our perspective correct bounding approach that prevents pop-in of Gaussians from outside the view-frustum; Finally, Sec. 3.4 proposes our novel frustum-based culling, a crucial component that improves both performance and sort-order through hierarchical culling.

3.1. Preliminaries

Gaussian Splatting. Following Kerbl *et al*. [14], we model the scene as a collection of anisotropic 3D Gaussians that serves as an approximation of the scene's geometric structure. Each Gaussian is parameterized by its 3D mean $\mathbf{\mu} \in \mathbb{R}^3$, scaling factor $\mathbf{s} \in \mathbb{R}^3_+$, and rotation quaternion $\mathbf{q} \in \mathbb{R}^4$. The probability density function of a Gaussian at a position $\mathbf{x} \in \mathbb{R}^3$ is defined as

$$\mathcal{G}(\mathbf{x}) = \exp\left(-\frac{1}{2}\rho(\mathbf{x})^2\right), \text{using} \tag{1}$$

$$\rho(\mathbf{x}) = \sqrt{(\mathbf{x}-\mathbf{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{\mu})}\,. \tag{2}$$

The covariance matrix $\mathbf{\Sigma}$ is given by

$$\mathbf{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^\top\mathbf{R}^\top, \tag{3}$$

where $\mathbf{S} = \mathrm{diag}(\mathbf{s})$ is a diagonal scaling matrix and $\mathbf{R}$ is the rotation matrix obtained from $\mathbf{q}$. For rendering, the 3D Gaussians are projected onto the 2D image plane. Since Gaussian distributions are not inherently preserved under nonlinear transformations, Kerbl *et al*. [14] employ a local affine approximation [39].

Evaluation in 3D. This approximation introduces a projection error that leads to rendering artifacts, especially at image borders [13]. To circumvent these issues, Hahlbohm *et al*. [11] evaluate Gaussians directly in 3D: They represent the ray through each pixel (x, y) as the intersection of the two screen space planes $\mathbf{\pi}_x = (1, 0, 0, -x)^\top$ and $\mathbf{\pi}_y = (0, 1, 0, -y)^\top$, which are then transformed into each Gaussian's normalized space, where the distance to the origin is equal to $\rho(\mathbf{x})$:

$$\mathbf{T}' = \mathbf{M}_{\text{vp}}\mathbf{P}\mathbf{V}\mathbf{T} \quad \text{using} \quad \mathbf{T} = \begin{pmatrix} \mathbf{R}\mathbf{S} & \mathbf{\mu} \\ \mathbf{0}^\top & 1 \end{pmatrix}, \tag{4}$$

where $\mathbf{T}$ is the transformation from Gaussian space to world space, while $\mathbf{M}_{\text{vp}}$, $\mathbf{P}$ and $\mathbf{V}$ correspond to the viewport, projection and view matrices, respectively. The transformation of the planes is therefore given by

$$\mathbf{\pi}'_{x/y} = \left(\mathbf{T}'^{-1}\right)^{-\top} \mathbf{\pi}_{x/y} = \mathbf{T}'^{\top}\mathbf{\pi}_{x/y}\,. \tag{5}$$

By computing the distance of the intersection line formed by the two planes to the origin, the Gaussian's contribution to the pixel ray is determined. In this formulation, the integral is no longer evaluated to determine the Gaussian contribution; instead, only the maximum contribution along the ray is considered. This formulation avoids the use of the inverse covariance $\mathbf{\Sigma}^{-1}$, which can become numerically instable for degenerate Gaussians (any $\mathbf{s}_i \approx 0$).

Anti Aliasing. Aliasing artifacts arise when the sampling rate during rendering deviates from the one used during training, *e.g*. changes in image resolution, focal length, or viewing distance (*cf*. Fig. 2). To address this, Yu *et al*. [36] combine a 3D smoothing filter with a 2D screen space Mip filter. The 3D smoothing filter prevents high-frequency sampling artifacts by applying a low-pass Gaussian filter to each Gaussian, where the filter size is determined by the maximum sampling frequency observed during training across all N_c training cameras, given by

$$\hat{\nu}_{\text{train}} = \max\left(\{\hat{\nu}_n\}_{n=1}^{N_c}\right), \quad \text{with} \quad \hat{\nu} = \frac{f}{d}, \tag{6}$$

where f denotes the camera's focal length in pixels and d represents the z-component of the Gaussian's mean $\mathbf{\mu}$ in view space. The smoothing filter is defined by covariance $\hat{\mathbf{\Sigma}} = \mathbf{\Sigma} + {}^{k}/_{\hat{\nu}^2}\,\mathbf{I}$, where k controls the size of the filter. The smoothed Gaussian is then given by

$$\hat{\mathcal{G}}(\mathbf{x}) = \sqrt{\frac{|\mathbf{\Sigma}|}{|\hat{\mathbf{\Sigma}}|}}\, \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{\mu})^\top \hat{\mathbf{\Sigma}}^{-1}(\mathbf{x}-\mathbf{\mu})\right). \tag{7}$$

To prevent low-frequency sampling artifacts—*e.g*. when increasing the camera distance—they replace the screen space dilation filter of Kerbl *et al*. [14] with a 2D Mip filter, which approximates a 2D box filter in image space.

Tile-Based Rendering. To optimize rendering efficiency, 3DGS partitions the image into tiles. Each Gaussian is evaluated for overlap with these tiles and assigned accordingly, ensuring that each tile processes only the Gaussians that intersect with it during rendering. To determine overlap, Kerbl *et al*. [14] compute a screen space bounding box based on the eigenvalues of the 2D covariance matrix.

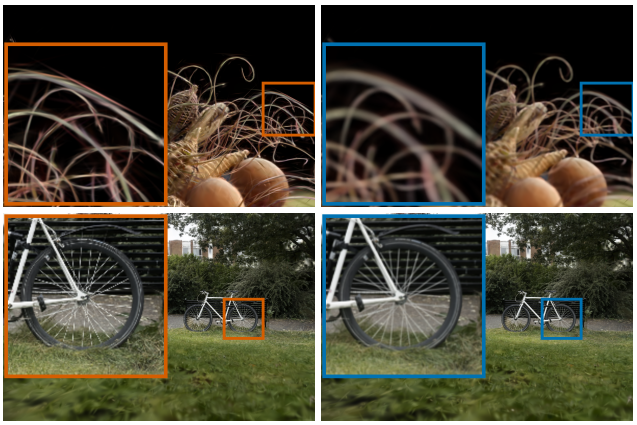

Figure 2. (Left) Aliasing artifacts manifest when camera positions deviate significantly from training distances: (1) Gaussians become too thin due to over-sampling when moving close. (2) Gaussians become too small due to under-sampling when moving farther away. (Right) By dynamically adjusting to varying view conditions, our adaptive 3D filter effectively removes these artifacts, preserving fine details and ensuring consistent image quality.

Blending Sort Order. Kerbl *et al*. [14] approximate the blending order based on the global depth of each Gaussian's view space mean. This strategy eliminates the need for a computationally expensive per-pixel sorting of Gaussians, but introduces popping artifacts when the camera rotates. To address this issue, Radl *et al*. [28] introduce an efficient hierarchical rasterization approach that approximates a pixel-perfect sorting, significantly reducing visual artifacts while maintaining computational efficiency by interleaving hierarchical sorting with repeated culling.

3.2. 3D Gaussian Anti-Aliasing

Aliasing is one of the most pressing challenges in 3D Gaussian evaluation, particularly when rendering scenes at varying distances. While the 3D smoothing filter introduced by Yu *et al*. [36] is inherently compatible with 3D evaluation, the 2D screen space Mip filter cannot be directly applied. As a result, recent approaches that rely on 3D evaluation omit the screen space filter and depend solely on the 3D smoothing filter [11, 37]. While this effectively mitigates artifacts when moving the camera closer, it remains susceptible to aliasing when increasing the viewing distance, as fine details are not adequately filtered, leading to flickering and loss of visual stability.

To address this limitation, we replace the 2D screen space Mip filter by a full 3D filtering approach that seamlessly integrates with 3D evaluation methods, effectively preventing low-frequency aliasing. A naïve approach would be to recompute the 3D smoothing filter for each rendering view, but this proves insufficient, as it causes Gaussians to become overly transparent (*cf*. Fig. 3). This issue arises because the

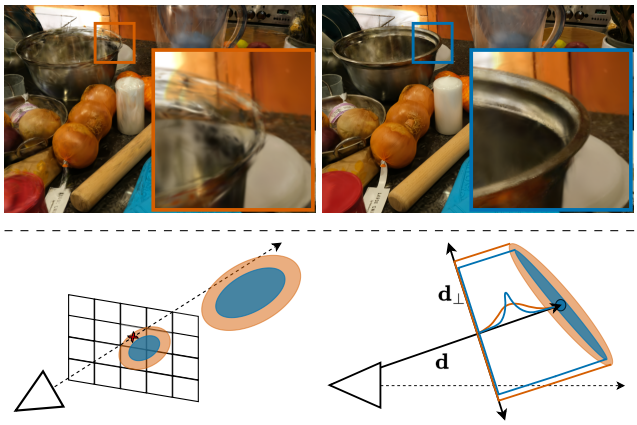

Figure 3. Improvement due to our adaptive 3D dilation filter: Whereas previous methods adjust amplitudes based solely on the change of volume of the 3D Gaussian, often leading to excessive transparency (top left), our method adapts based on the area perpendicular to the viewing ray.

amplitude in Eq. (7) decreases according to the change in volume, while Gaussians are evaluated only at their point of maximum contribution along a ray $\mathbf{d}$, rather than being fully integrated along the ray. Consequently, incorporating the scaling change along the ray overestimates amplitude scaling for highly anisotropic Gaussians, which leads to an excessively small normalization factor

$$\sqrt{\frac{|\boldsymbol{\Sigma}|}{|\hat{\boldsymbol{\Sigma}}|}} = \sqrt{\frac{\prod_{i=1}^{3} \mathbf{s}_i^2}{\prod_{i=1}^{3} \left(\mathbf{s}_i^2 + \frac{k}{\hat{\nu}^2}\right)}}\,, \tag{8}$$

in Eq. (7). Notably, this reduction occurs even when the scale change perpendicular to $\mathbf{d}$ is minimal, highlighting the need for a more robust filtering approach. Therefore, we reformulate the normalization to only factor in the change of scale perpendicular to $\mathbf{d}$ (*cf*. Fig. 3), that is

$$\hat{\mathcal{G}}_\perp(\mathbf{x}) = \sqrt{\frac{|\boldsymbol{\Sigma}_\perp|}{|\hat{\boldsymbol{\Sigma}}_\perp|}}\, \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\,, \tag{9}$$

where $\mathbf{d}$ is the normalized vector between $\boldsymbol{\mu}$ and the camera origin $\mathbf{o}$, and $\boldsymbol{\Sigma}_\perp$ denotes the 2×2 covariance matrix projected onto the subspace orthogonal to $\mathbf{d}$. It can be shown (*cf*. the supplementary material for a derivation) that the perpendicular scaling factor is given by

$$\sqrt{\frac{|\boldsymbol{\Sigma}_\perp|}{|\hat{\boldsymbol{\Sigma}}_\perp|}} = \sqrt{\frac{|\boldsymbol{\Sigma}|\ \mathbf{d}^\top \boldsymbol{\Sigma}^{-1}\mathbf{d}}{|\hat{\boldsymbol{\Sigma}}|\ \mathbf{d}^\top \hat{\boldsymbol{\Sigma}}^{-1}\mathbf{d}}}\,. \tag{10}$$

Since the inverse covariance matrix is given by $\boldsymbol{\Sigma}^{-1} = \mathbf{R}\mathbf{S}^{-2}\mathbf{R}^\top$, we can express the directional quadratic form as

$$\mathbf{d}^\top \boldsymbol{\Sigma}^{-1}\mathbf{d} = \mathbf{d}^\top \mathbf{R}\, \mathbf{S}^{-2}\mathbf{R}^\top \mathbf{d} = \sum_{i=1}^{3} \frac{\mathbf{d}_i'^2}{\mathbf{s}_i^2}\,, \tag{11}$$

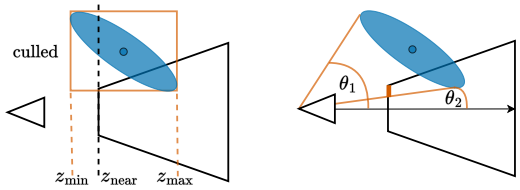

Figure 5. (Left) Hahlbohm *et al*. [11] compute the screen bounds of a Gaussian by fitting planes in screen space. However, they discard Gaussians whose z-bounds ($z_{\text{min,max}}$) are outside the near/far-planes, which can lead to popping. (Right) We instead compute view space angles $\theta_{1,2}$, leading to a more robust computation and bounding.

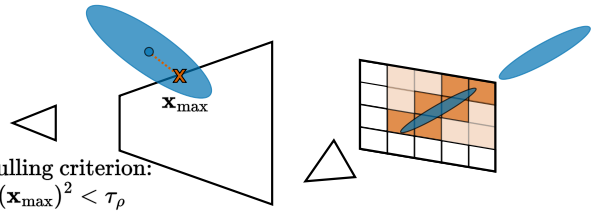

Figure 6. Our frustum culling algorithm finds the maximum contribution point inside a 3D frustum, via projection of the origin onto the transformed planes and edges in Gaussian space. Comparing $\rho(\mathbf{x})$ at this maximum point against the threshold τ_ρ, we can cull away a whole Gaussian against the view frustum (Left), as well as individual tiles for a single Gaussian (Right).

where $\mathbf{d}' = \mathbf{R}^\top \mathbf{d}$. Using the determinant factorization from Eq. (8), we can simplify Eq. (10) to

$$\sqrt{\frac{\mathbf{d}_1'^2\,\mathbf{s}_2^2\,\mathbf{s}_3^2 + \mathbf{d}_2'^2\,\mathbf{s}_1^2\,\mathbf{s}_3^2 + \mathbf{d}_3'^2\,\mathbf{s}_1^2\,\mathbf{s}_2^2}{\mathbf{d}_1'^2\,\hat{\mathbf{s}}_2\,\hat{\mathbf{s}}_3 + \mathbf{d}_2'^2\,\hat{\mathbf{s}}_1\,\hat{\mathbf{s}}_3 + \mathbf{d}_3'^2\,\hat{\mathbf{s}}_1\,\hat{\mathbf{s}}_2}}\,, \tag{12}$$

where the updated scaling factors are given by $\hat{\mathbf{s}}_i = \mathbf{s}_i^2 + {}^k\!/_{\hat{v}^2}$. This formulation enables efficient computation while avoiding explicit matrix inversion, ensuring numerical stability. To address artifacts when moving the camera close to a Gaussian, we integrate the 3D smoothing filter of Yu *et al*. [36] with our proposed 3D kernel: We store the maximum sampling frequency observed across all training cameras $\hat{v}_{\text{train}}$. During rendering, the effective sampling frequency is defined as

$$\hat{v}' = \min(\hat{v}_{\text{train}}, \hat{v})\,, \tag{13}$$

where $\hat{v}'$ corresponds to the sampling frequency used for our filter. This formulation ensures that Gaussians do not shrink excessively when the camera moves closer, while still providing effective anti-aliasing when the camera moves farther away.

3.3. Perspective Correct Bounding

Efficient evaluation of 3D Gaussians in a software rasterizer requires accurate bounding in screen space to avoid unnecessary evaluations. Following Sigg *et al*. [30], Hahlbohm *et al*. [11] perform exact plane fitting to the ellipsoid, defined by the 3D Gaussian's level set at τ_ρ in projective space, which fails when a Gaussian's extent reaches behind the image plane. To mitigate this, they discard these Gaussians, leading to noticeable popping (*cf*. Fig. 5). Instead, we perform the plane fitting in view space with planes $\boldsymbol{\pi}_\theta = (\cos(\theta), 0, -\sin(\theta), 0)^\top$, and $\boldsymbol{\pi}_\phi = (0, \cos(\phi), -\sin(\phi), 0)^\top$, and solving for θ and ϕ:

$$\theta_{1,2} = \tan^{-1}\left(\frac{s_{1,3} \pm \sqrt{s_{1,3}^2 - s_{1,1}s_{3,3}}}{s_{3,3}}\right), \tag{14}$$

$$\phi_{1,2} = \tan^{-1}\left(\frac{s_{2,3} \pm \sqrt{s_{2,3}^2 - s_{2,2}s_{3,3}}}{s_{3,3}}\right), \tag{15}$$

with $s_{i,j} = \langle \mathbf{t}, \mathbf{T}_{(\text{view},i)} \odot \mathbf{T}_{(\text{view},j)} \rangle$, $\boldsymbol{t} = (\tau_\rho, \tau_\rho, \tau_\rho, -1)^\top$, and $\mathbf{T}_{(\text{view},i)}$ denoting the i-th row of transformation matrix from Gaussian space to view space $\mathbf{T}_{\text{view}} = \mathbf{VT}$ (*cf*. the supplementary material for a derivation). We additionally compute the angles θ_μ, ϕ_μ of the view space Gaussian mean in x/y-direction. This is followed by a rotation step, where we ensure that

$$(\theta_\mu - \pi) < \theta_1 < \theta_\mu < \theta_2 < (\theta_\mu + \pi), \tag{16}$$

and a bounding step to the range $[-(\frac{\pi}{2} - \epsilon), \frac{\pi}{2} - \epsilon]$ (with a small $\epsilon \in \mathbb{R}_+$) to translate those bounds to the screen

$$\theta_1 = \max\left(-\frac{\pi}{2} + \epsilon, \theta_1\right), \theta_2 = \min\left(\frac{\pi}{2} - \epsilon, \theta_2\right). \tag{17}$$

If the camera center is inside the Gaussian's ellipsoid, no valid solution exists, in which case we discard this Gaussian. Additionally, if the ellipsoid intersects the x-axis, it cannot be bounded in the screen space y-axis, and vice-versa. In these cases, the term inside the square root becomes negative, and we conservatively set the bounds to the entire screen for the affected axis. In the next step, tile-based culling removes all tiles that receive negligible contribution from the Gaussian or where the Gaussian's depth remains closer than the near plane across the entire tile.

3.4. Frustum-Based Culling

Accurate screen space bounding with axis-aligned bounding boxes (AABBs) of the ellipsoids helps to reduce the per-pixel workload, however, it delivers bad bounds for highly

non-axis-aligned ellipsoids. Additionally, Gaussians could receive valid screen bounds, but never contribute to any pixel. To mitigate this, Radl *et al*. [28] perform per-tile culling of the projected 2D ellipse, which drastically reduces the number of Gaussian/tile combinations, and also prevents them from performing unnecessary sorting operations in their hierarchical sort. We translate this tile-based culling approach to 3D by constructing a per-tile frustum $\mathcal{F}$ from 4 planes $\boldsymbol{\pi}_{x_{1,2}} = (1, 0, 0, -x_{(\min,\max)}), \boldsymbol{\pi}_{y_{1,2}} = (0, 1, 0, -y_{(\min,\max)})$, where $x_{(\min,\max)}, y_{(\min,\max)}$ define the tile boundaries in pixel coordinates. We then compute the point of maximum contribution of the Gaussian inside this 3D frustum and discard tiles where $\rho(\mathbf{x})^2$ is above the threshold τ_ρ, *i.e*.

$$\min_{\mathbf{x}\in\mathcal{F}} \rho(\mathbf{x})^2 < \tau_\rho. \tag{18}$$

In the trivial case where the Gaussian's mean is already inside this frustum, it is consequently the point of maximum contribution. Otherwise, this point has to lie on the planes and edges of the frustum. We find the maximum contribution point on the planes by transforming them into the normalized Gaussian space, and finding the point closest to the origin. A naïve solution is to do this for each plane and edge, and ensuring that the projected point lies within the bounds of the frustum and in front of the camera. Instead, we only project onto the x/y-planes (and their corresponding edges) that are closest to the Gaussian's mean in screen space, limiting the evaluations to 2 planes and 3 edges (instead of 4 planes and 4 edges for the naïve approach). This can be implemented efficiently, which is critical as this routine has to be executed for many tiles per Gaussian.

Additionally, we cull Gaussians during pre-processing against the entire view frustum to discard all non-contributing Gaussians (*cf*. Fig. 6). In contrast to our exact frustum culling, other methods discard Gaussians only based on their mean (*e.g*., if it lies behind the image plane [14]) or based on the screen space z-bounds [11], which results in incorrect culling of contributing Gaussians and popping artifacts at the image borders.

4. Evaluation

Following prior work, we evaluate our method on 13 outdoor and indoor scenes from three different datasets: Mip-NeRF 360 [2], Tanks & Temples [17], and Deep Blending [12].

Implementation Details. We use the pre-downscaled images of Mip-NeRF 360 [2] for training and evaluation, following the setup of 3DGS [14]. For densification, we adopt Markov Chain Monte Carlo (MCMC) [16] with identical parameter settings. Our approach builds on the hierarchical rasterizer of Radl *et al*. [28], retaining their per-ray sorting and queuing strategies while replacing the bounding, culling, depth evaluation, contribution estimation, and anti-aliasing with our 3D-aware implementations. All compared methods optimize to the same number of primitives, with the exception of Hybrid Transparency [11], where we re-evaluate image metrics from their original results, as their training code was unavailable at the time of writing. Following previous works [14, 36, 39] we use a kernel size of $k = 0.3$ for our adaptive 3D filter.

4.1. Image Metrics

We compare our method against 3DGS [14], MCMC [16], and Taming 3DGS [20], which all use different densification approaches. Additionally, we compare against StopThePop [28] and Mip-Splatting [36], both of which build on the original 3DGS densification but specifically target artifact removal. For Hybrid Transparency [11], we re-evaluate image metrics on their provided results for the Mip-NeRF360 dataset. Our evaluation considers PSNR, SSIM, and LPIPS [38], with LPIPS computed on unnormalized images to maintain consistency with prior work.

Standard Datasets. We begin by evaluating standard datasets, which represent in-distribution camera and view parameters. As shown in Tab. 1, our method outperforms others in nearly all metrics and matches MCMC in overall quality—while suffering from none of the artifacts. Notably, our approach prevents the optimizer from "cheating per-view" inconsistencies via popping which explains the slightly lower PSNR in Tanks & Temples. In this dataset, MCMC relies on distorted Gaussians and popping to compensate for large exposure changes, effectively "faking" full-screen adjustments. However, as shown in Fig. 7, this strategy breaks down for out-of-distribution viewing configurations.

Ablation. We analyze the impact of our components in Tab. 2. While standard image metrics remain largely unaffected for in-distribution test views, disabling individual components leads to distinct artifacts. Our anti-aliasing ensures stability across resolution changes and distance to the observed scene content (Tab. 4). Removing hierarchical per-pixel sorting introduces popping artifacts and allows the method to cheat with a view-inconsistent representation (again see Radl *et al*. [28]). Finally, disabling 3D evaluation results in projection artifacts, particularly noticeable when increasing the field of view (Fig. 7). Frustum-based culling has no impact on reconstruction quality, but it plays a significant role in improving performance (*cf*. Tab. 5).

4.2. View-Consistent Rendering

Densification primarily impacts image metrics on in-distribution test views, as inconsistencies are implicitly learned during training. However, for out-of-distribution

	popping	aliasing	distortion	Mip-NeRF 360			Tanks & Temples			Deep Blending		
				PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS [14]	✗	✗	✗	27.443	0.814	0.215	23.734	0.847	0.175	29.510	0.902	0.237
StopThePop [28]		✗	✗	27.304	0.815	0.211	23.226	0.846	0.171	29.929	0.908	0.231
Mip-Splatting [36]	✗		✗	27.540	0.817	0.216	23.821	0.852	0.176	29.660	0.905	0.243
MCMC [16]	✗	✗	✗	28.027	0.836	0.187	24.642	0.872	0.147	29.727	0.906	0.233
Taming 3DGS [20]	✗	✗	✗	27.826	0.823	0.207	24.067	0.855	0.168	29.878	0.910	0.235
Hybrid Transparency† [11]	*	*		27.169	0.822	0.195	-	-	-	-	-	-
Ours				27.835	0.836	0.188	23.582	0.867	0.145	30.485	0.913	0.222

Table 1. Standard image metrics for our method and related work for in-distribution views. Even though we focus on out-of-distribution effects, our approach matches the state-of-the-art for in-distribution views. We also include an overview of the artifacts each method exhibits. *While Hybrid Transparency [11] does not suffer from classical popping, they still experience "pop-in" at image borders due to incorrect culling, as well as aliasing due to undersampling of their fixed 3D filter. †Numbers were re-evaluated on the original evaluation images.

	popping	aliasing	distortion	Mip-NeRF 360			Tanks & Temples			Deep Blending		
				PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Ours				27.835	0.836	0.188	23.582	0.867	0.145	30.485	0.913	0.222
Ours w/o hier. sort	✗			27.898	0.836	0.189	23.561	0.865	0.148	30.325	0.912	0.226
Ours w/o AA		✗		27.811	0.836	0.185	23.648	0.867	0.141	30.344	0.910	0.224
Ours w/o 3D eval.			✗	27.874	0.836	0.189	24.119	0.869	0.150	30.444	0.912	0.223

Table 2. Ablation study on the effect of our components for in-distribution views. Removing individual features may actually increase image metrics, as method can better overfit to the data set views. See the out-of-distribution evaluation for the benefits of our contributions.

views—*e.g.* with a larger field of view (FOV) or changed resolution—these inconsistencies become more apparent in evaluation metrics.

Larger Field of View. We assess FOV robustness by artificially increasing the FOV and resolution of test views while extracting a pixel-perfect cutout from the original image for ground truth comparison. Our setup follows Huang *et al*. [13] but decreases focal length by 3× and increases resolution by 3×. As shown in Tab. 3, our method remains unaffected by these changes, whereas all other approaches suffer significant quality degradation due to distortion artifacts. A visual comparison is provided in Fig. 7.

Changing Resolution. To assess our method's anti-aliasing capability, we perform a multi-resolution evaluation using the original training resolution (1×), half resolution ($\frac{1}{2}$×), and double resolution (2×). In Tab. 4, we compare results for two Mip-NeRF 360 scenes against the provided pre-downscaled (or original size) images. While all methods perform similarly at 1× resolution, our anti-aliasing 3D filter preserves quality across lower and higher resolutions (*cf*. supplementary material for visual comparisons).

Dataset	Mip-NeRF 360			Tanks & Temples			Deep Blending		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Mip-Splatting	26.053	0.784	0.245	17.314	0.737	0.256	26.192	0.875	0.247
3DGS	26.820	0.804	0.219	17.112	0.741	0.229	27.553	0.889	0.243
StopThePop	27.040	0.812	0.213	20.241	0.809	0.192	25.592	0.854	0.296
Mip-Splatting	26.053	0.784	0.245	17.314	0.737	0.256	18.315	0.782	0.355
MCMC	23.347	0.779	0.213	14.369	0.668	0.296	20.328	0.823	0.282
Taming 3DGS	23.296	0.763	0.231	11.545	0.534	0.448	30.488	0.913	0.222
Ours	27.836	0.836	0.188	23.583	0.867	0.145			

Table 3. The large FOV evaluation shows that 3D Gaussian evaluation leads to more faithful reconstruction and rendering. Compared to related work which relies on evaluating 2D splats on the image plane, our method gracefully retains its image quality in this challenging out-of-distribution rendering scenario.

Close to Scene Camera Location. Popping artifacts caused by incorrect culling when moving close to scene

Res.		Bonsai			Bicycle		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
$\frac{1}{2}\times$	MCMC	28.98	0.941	0.099	21.15	0.774	0.171
	Ours	32.12	0.953	0.095	26.73	0.854	0.123
	Ours w/o AA	28.54	0.939	0.097	20.99	0.771	0.174
$1\times$	MCMC	32.65	0.948	0.191	25.69	0.799	0.168
	Ours	32.32	0.948	0.189	25.74	0.801	0.171
	Ours w/o AA	32.13	0.947	0.189	25.65	0.801	0.165
$2\times$	MCMC	31.49	0.936	0.279	22.13	0.689	0.288
	Ours	32.09	0.940	0.276	24.52	0.728	0.264
	Ours w/o AA	30.99	0.935	0.278	21.90	0.687	0.288

Table 4. Multi-resolution evaluation ablation of MCMC and our method, with and without anti-aliasing. Ours gives clearly better results when changing the resolution during test time, highlighting the benefits of the 3D anti-aliasing filter.

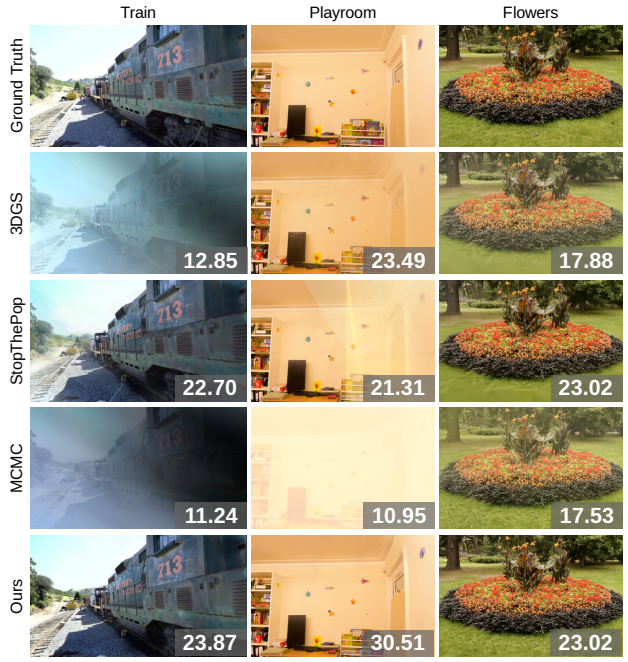

Figure 7. Example results with PSNR when rendering with a larger FOV, comprising out-of-distribution view settings. Clearly, other methods suffer from severe distortion artifacts.

content—which our method resolves—are difficult to evaluate quantitatively, as ground truth data cannot be generated from existing evaluation views. An example for this kind of popping is shown in Fig. 1. For more examples, please see the supplemental video.

4.3. Performance Timings

We evaluate the runtime performance of our components and MCMC in Tab. 5, using an NVIDIA RTX 4090 with timings averaged over an interpolated camera path across all available camera poses. Our method is only slightly slower than MCMC (with standard 3DGS bounding) and even outperforms it on the Tanks and Temples dataset. When removing culling, performance drops significantly, as the hierarchical sort heavily relies on culling to reduce its sorting overhead. Disabling hierarchical sorting improves speed beyond MCMC but introduces noticeable popping artifacts. Lastly, disabling our 3D evaluation results in similar or worse performance, indicating that our 3D evaluation is as fast or even faster than 2D splat variants.

Timings in ms	M360 Indoor	M360 Outdoor	T&T	DB
Ours	7.72	10.66	7.03	5.81
Ours w/o culling	14.40	22.98	12.78	8.88
Ours w/o hier. sort	4.11	6.29	3.69	3.47
Ours w/o 3D	7.64	10.30	7.52	6.32
MCMC	6.79	8.81	8.28	4.43

Table 5. Average performance timings for different configuration of our method and MCMC. As expected, hierarchical sorting introduces the largest performance cost. However, accurate culling compensates for a significant portion of the cost, demonstrating the high efficiency of our 3D culling. Notably, our 3D evaluation is as fast as or even faster than 2D splat approximations.

5. Conclusion, Limitations, And Future Work

In this work, we addressed the limitations of current 3D Gaussian Splatting methods and made several key contributions to enable fast, artifact-free rendering of 3D Gaussians. Our method introduces a novel 3D smoothing filter that effectively removes aliasing artifacts when evaluating Gaussians in 3D, along with stable 3D bounding and culling that performs consistently across various viewing scenarios. We thoroughly evaluated the effectiveness of our components, showing that our method is robust to out-of-distribution camera views while maintaining standard image metrics on par with the current 3DGS state-of-the-art. To our knowledge, we are the only rasterization-based approach capable of delivering artifact-free rendering of 3D Gaussians, with framerates exceeding 100 FPS on consumer-grade hardware.

While achieving more view-consistent results, we observe that standard image metrics do not show significant improvement when evaluation views stay within the training distribution. Although our view-space bounding approach is less tied to the perspective projection, it remains closely tied to the pinhole camera model, which limits its adaptability to other camera models. Furthermore, as our method exhibits stronger view consistency and less room for exploiting view-dependent effects, it would benefit disproportionately from a more expressive view-dependent encoding.

Acknowledgments

This research was supported by the Austrian Science Fund *FWF* [10.55776/I6663], the German Science Foundation *DFG* [contract 528364066] and the *Alexander von Humboldt Foundation* funded by the German Federal Ministry of Research, Technology and Space. For open access purposes, the author has applied a CC BY public copyright license to any author-accepted manuscript version arising from this submission.

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. In *IEEE/CVF International Conference on Computer Vision*, 2021. 2

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2, 6

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *IEEE/CVF International Conference on Computer Vision*, 2023. 2

[4] Krzysztof Byrski, Marcin Mazur, Jacek Tabor, Tadeusz Dziarmaga, Marcin Kądziołka, Dawid Baran, and Przemysław Spurek. RaySplats: Ray Tracing based Gaussian Splatting. *arXiv preprint arXiv:2501.19196*, 2025. 2

[5] Adam Celarek, George Kopanas, George Drettakis, Michael Wimmer, and Bernhard Kerbl. Does 3D Gaussian Splatting Need Accurate Volumetric Rendering? In *Computer Graphics Forum*, page e70032, 2025. 2

[6] Jorge Condor, Sebastien Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media. *ACM TOG*, 2025. 2

[7] Daniel Duckworth, Peter Hedman, Christian Reiser, Peter Zhizhin, Jean-François Thibert, Mario Lučić, Richard Szeliski, and Jonathan T. Barron. SMERF: Streamable Memory Efficient Radiance Fields for Real-Time Large-Scene Exploration. *ACM TOG*, 43(4), 2024. 2

[8] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. In *Advances in Neural Information Processing Systems*, 2024. 2

[9] Guangchi Fang and Bing Wang. Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians. In *European Conference on Computer Vision*, 2024. 2

[10] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields without Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[11] Florian Hahlbohm, Fabian Friederichs, Tim Weyrich, Linus Franke, Moritz Kappel, Susana Castillo, Marc Stamminger, Martin Eisemann, and Marcus Magnor. Efficient Perspective-Correct 3D Gaussian Splatting Using Hybrid Transparency, 2024. 1, 2, 3, 4, 5, 6, 7

[12] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep Blending for Free-viewpoint Image-based Rendering. *ACM TOG*, 37(6), 2018. 6

[13] Letian Huang, Jiayang Bai, Jie Guo, Yuanqi Li, and Yanwen Guo. On the Error Analysis of 3D Gaussian Splatting and an Optimal Projection Strategy. In *European Conference on Computer Vision*, 2024. 2, 3, 7

[14] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 42(4), 2023. 1, 2, 3, 4, 6, 7

[15] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM TOG*, 43(4), 2024. 2

[16] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian Splatting as Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*, 2024. 2, 6, 7

[17] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM TOG*, 36(4), 2017. 6

[18] Zhihao Liang, Qi Zhang, Wenbo Hu, Ying Feng, Lei Zhu, and Kui Jia. Analytic-Splatting: Anti-Aliased 3D Gaussian Splatting via Analytic Integration. *arXiv preprint arXiv:2403.11056*, 2024. 2

[19] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T. Barron, and Yinda Zhang. EVER: Exact Volumetric Ellipsoid Rendering for Real-time View Synthesis, 2024. 2

[20] Saswat Mallick, Rahul Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. Taming 3DGS: High-Quality Radiance Fields with Limited Resources. In *SIGGRAPH Asia Conference Papers*, 2024. 2, 6, 7

[21] Marilena Maule, João Comba, Rafael Torchelsen, and Rui Bastos. Hybrid transparency. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 103–118, 2013. 2

[22] 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 *European Conference on Computer Vision*, 2020. 2

[23] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM TOG*, 2024. 1

[24] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Riccardo Perel, or de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler,

Nicholas Sharo, and Zan Gojcic. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM TOG*, 43(6), 2024. 2

[25] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG*, 41(4), 2022. 2

[26] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. In *Computer Graphics Forum*, pages 45–59. Wiley Online Library, 2021. 2

[27] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. RadSplat: Radiance Field-Informed Gaussian Splatting for Robust Real-Time Rendering with 900+ FPS. *arXiv preprint 2403.13806*, 2024. 2

[28] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM TOG*, 43(4), 2024. 1, 2, 3, 4, 6, 7

[29] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. 2

[30] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus Gross. GPU-Based Ray-Casting of Quadratic Surfaces. In *Symposium on Point-Based Graphics*. The Eurographics Association, 2006. 3, 5

[31] Michael Steiner, Thomas Köhler, Lukas Radl, and Markus Steinberger. Frustum Volume Caching for Accelerated NeRF Rendering. *Proceedings of the ACM on Computer Graphics and Interactive Technologies*, 7(3), 2024. 2

[32] Chinmay Talegaonkar, Yash Belhe, Ravi Ramamoorthi, and Nicholas Antipa. Volumetrically Consistent 3D Gaussian Rasterization, 2025. 1, 2

[33] Xuechang Tu, Lukas Radl, Michael Steiner, Markus Steinberger, Bernhard Kerbl, and Fernando de la Torre. VRSplat: Fast and Robust Gaussian Splatting for Virtual Reality. *Proceedings of the ACM on Computer Graphics and Interactive Technologies*, 8(1), 2025. 2

[34] Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moenne-Loccoz, and Zan Gojcic. 3DGUT: Enabling Distorted Cameras and Secondary Rays in Gaussian Splatting, 2024. 1, 2

[35] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20923–20931, 2024. 2

[36] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian Splatting. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2, 3, 4, 5, 6, 7

[37] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM TOG*, 2024. 1, 2, 4

[38] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 6

[39] Mathias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. EWA Volume Splatting. In *IEEE Visualization*, 2001. 2, 3, 6