DOI: 10.1111/cgf.70014
EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)
COMPUTER GRAPHICS *forum*

Efficient Perspective-Correct 3D Gaussian Splatting Using Hybrid Transparency

Florian Hahlbohm[1] Fabian Friederichs[1] Tim Weyrich[2,3] Linus Franke[2] Moritz Kappel[1]
Susana Castillo[1] Marc Stamminger[2] Martin Eisemann[1] Marcus Magnor[1,4]

[1]Computer Graphics Lab, TU Braunschweig, Germany {lastname}@cg.cs.tu-bs.de
[2] Visual Computing Erlangen, FAU Erlangen-Nürnberg, Germany {firstname.lastname}@fau.de
[3]University College London, United Kingdom [4]University of New Mexico, USA
https://fhahlbohm.github.io/htgs/

Figure 1: *We present the first perspectively and simultaneously geometrically accurate approach for real-time rendering of 3D Gaussian splats. Our temporally stable blending formulation based on hybrid transparency effectively removes the popping artifacts caused by the approximate, global sorting scheme of 3D Gaussian Splatting (3DGS) [KKLD23]. Additionally, we replace the approximate projection used in 3DGS with a novel, numerically stable formulation for evaluation of general 3D Gaussians along per-pixel rays to improve rendering quality. We also achieve a significant speedup compared to state-of-the-art approaches addressing these challenges [HYC*24, RSP*24], as our blending formulation only requires partial depth-ordering and our splat evaluation is well-suited for rasterization.*

Abstract

3D Gaussian Splats (3DGS) have proven a versatile rendering primitive, both for inverse rendering as well as real-time exploration of scenes. In these applications, coherence across camera frames and multiple views is crucial, be it for robust convergence of a scene reconstruction or for artifact-free fly-throughs. Recent work started mitigating artifacts that break multi-view coherence, including popping artifacts due to inconsistent transparency sorting and perspective-correct outlines of (2D) splats. At the same time, real-time requirements forced such implementations to accept compromises in how transparency of large assemblies of 3D Gaussians is resolved, in turn breaking coherence in other ways. In our work, we aim at achieving maximum coherence, by rendering fully perspective-correct 3D Gaussians while using a high-quality approximation of accurate blending, hybrid transparency, on a per-pixel level, in order to retain real-time frame rates. Our fast and perspectively accurate approach for evaluation of 3D Gaussians does not require matrix inversions, thereby ensuring numerical stability and eliminating the need for special handling of degenerate splats, and the hybrid transparency formulation for blending maintains similar quality as fully resolved per-pixel transparencies at a fraction of the rendering costs. We further show that each of these two components can be independently integrated into Gaussian splatting systems. In combination, they achieve up to 2× higher frame rates, 2× faster optimization, and equal or better image quality with fewer rendering artifacts compared to traditional 3DGS on common benchmarks.

CCS Concepts

• ***Computing methodologies*** → ***Rendering; Point-based models; Rasterization; Machine learning approaches;***

1. Introduction

Novel view synthesis has undergone a significant transformation with the advent of Neural Radiance Fields (NeRF) [MST*20] and 3D Gaussian Splatting (3DGS) [KKLD23], both of which have established themselves as powerful techniques for rendering complex 3D scenes. Among these, 3DGS has emerged as the de-facto representation for user-facing radiance field applications due to its fast optimization and real-time rendering capabilities. Its speed and efficiency stem from an explicit point-based representation, where each point's extent is modeled by an anisotropic 3D Gaussian. Using fast, tile-based rasterization, this can be efficiently implemented on modern GPUs.

However, the remarkable performance of 3DGS is achieved through a series of approximations that, while effective in many scenarios, introduce limitations in multi-view consistency. One significant issue arises from the affine approximation used in projecting 3D Gaussian splats onto the image plane, causing differing pixel contributions depending on camera placement. Another one is the per-primitive sorting, which causes elliptical color patches to suddenly appear or disappear during movement (called popping [RSP*24]). In order to solve the issues, it is necessary to accurately sort and evaluate Gaussians per pixel, introducing a strong bottleneck on the sorting stage. In this paper, we introduce an pipeline which improves the original 3DGS framework in eliminating these artifacts while offering superior performance through a hybrid transparency approximation.

For projection artifacts, although the affine approximation performs well on benchmark datasets, it fails to model perspective distortion correctly, especially when parts of the scene are viewed at close distances. The result are visually disturbing artifacts, where the projected Gaussians take on extreme, distorted shapes, severely affecting the rendering quality (see Fig. 2).

Recent work on 2D Gaussian Surfels [HYC*24] has made strides in achieving perspective-accurate rendering by leveraging established techniques from Sigg et al. [SWBG06] and Weyrich et al. [WHA*07]. For 3D Gaussians, concurrent research explores evaluating splats by calculating intersection points with per-pixel viewing rays [YSG24]. This approach can be implemented in ray-tracing frameworks [MLMP*24], whose rendering speeds are significantly lower. Alternatively, the rasterization-based approach faces numerical instability due to the matrix inversions required, making optimization via gradient descent challenging and prone to catastrophic failure if handled incorrectly.

Figure 2: *In 3DGS [KKLD23], Kerbl et al. use an affine approximation for projecting 3D Gaussians. Common benchmarks do not include viewpoints where this approximation matter is highlighted. Here, we demonstrate the benefits of our perspective-correct projection by comparing renderings from 3DGS and our method.*

In this paper, we propose a fast, differentiable method for perspective-accurate 3D Gaussian splat evaluation at the point of maximum contribution along per-pixel viewing rays that avoids matrix inversion entirely.

The second issue with 3DGS lies in depth ordering during rendering, where only the view-space depth of the mean is considered. This causes incorrect per-pixel blending order of fragments, leading to the so-called "popping" artifacts that disrupt multi-view consistency and especially the viewing experience in motion. Recent work, such as *StopThePop* [RSP*24], addresses this through hierarchical sorting, which is comprised of global presorting and then locally sorting within a sliding window, with progressively lower tile sizes.

We propose a similar idea, based on the established rendering paradigm of *Hybrid Transparency* [Wym16], which lets us skip the global presorting and provides high performance. By alpha-blending the first K fragments (called the *core*) in correct depth-order per pixel and accumulating remaining contributions (the *tail*) using an order-independent residual, our method mitigates the popping artifacts (see Fig. 3) while maintaining superior performance.

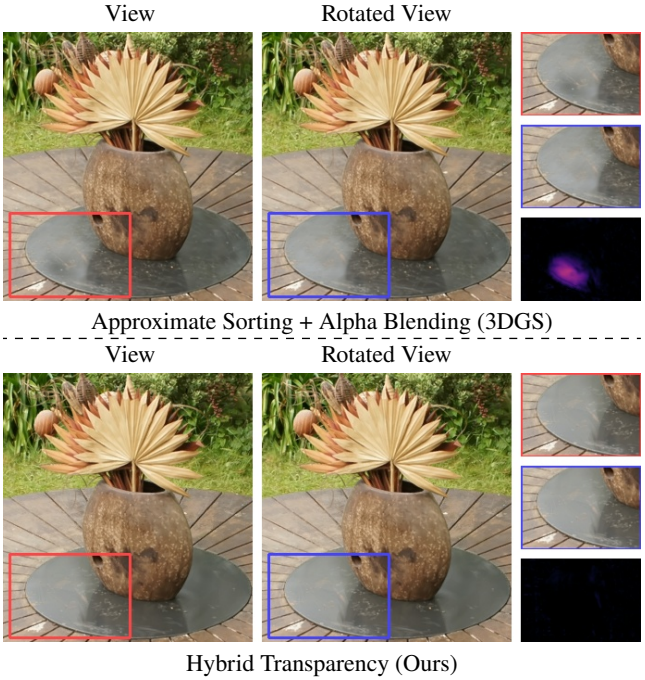

Figure 3: *Radl et al. [RSP*24] discuss how 3DGS approximates depth-ordering for alpha blending, leading to popping artifacts during view rotation. To solve this without sacrificing performance, we propose a hybrid transparency approach, combining alpha blending with order-independent transparency, resulting in temporally stable rendering and an improved viewing experience.*

These two enhancements – perspective-accurate splat evaluation and improved depth-ordering – can be integrated into existing 3DGS systems independently. In our implementation, we combine both improvements, demonstrating that they not only eliminate the respective artifacts but also enhance rendering speed, crucial for real-time applications that rely on fast scene inspection. In summary, we make the following contributions:

- We present a novel, matrix inversion-free ray-splat intersection method within a differentiable renderer for 3D Gaussian splats, resulting in numerically stable optimization and fully perspective-correct rendering.
- We introduce hybrid transparency to 3D Gaussian splat rendering, resulting in both very stable results and improved performance.
- We evaluate our efficient implementation, which outperforms established methods significantly, while matching or exceeding their visual fidelity.

2. Related Work

This work is at the intersection of novel view synthesis, Gaussian splatting, and order-independent transparency.

2.1. Novel View Synthesis

Novel view synthesis aims to render images of a scene from arbitrary viewpoints given a fixed number of input images.

The field has recently been revolutionized by Neural Radiance Fields (NeRF) introduced by Mildenhall et al. [MST*20]. NeRF represents a scene by optimizing a large MLP that maps positions and viewing directions to volumetric density and color. The use of differentiable volume rendering enables optimization using gradient descent, which leads to high-quality results; however, NeRF's slow training and rendering process hinders its use in interactive applications. To accelerate NeRF, voxel grids with trilinear interpolation were introduced, offering faster, continuous representations [FKYT*22, SSC22]. Memory-efficient methods, such as multi-resolution hash tables [MESK22] and tensor factorization [CXG*22, RSV*23], further reduce computational overhead. The current state-of-the-art method in terms of image quality, Zip-NeRF [BMV*23], combines these grid-based methods with solutions addressing aliasing issues [BMT*21, BMV*22].

Alternatively, point-based models use explicit point clouds for geometry, rendering images through fast rasterization. Due to the discrete nature of point clouds, recent point-based radiance field methods employ large convolutional neural networks (CNN) for hole-filling in image-space [ASK*20, RFS22, FRF*23, KPLD21, HFF*23] or MLPs for feature decoding [KLR*22], but these NNs are computationally expensive. Addressing this limitation, Franke et al. showed that assigning a radius to each point in combination with trilinear interpolation into an image pyramid allows using a smaller CNN [FRFS24]. While enabling fast rendering, image quality of point-based models is often lacking behind implicit NeRF- and grid-based models. Very recently, Hahlbohm et al. showed that point-based models and grid-based models can be combined to improve image quality and robustness [HFK*25]. Remaining issues with point-based models are temporal instabilities and the reliance on dedicated GPUs, which are both due to the used CNN. Another approach involves converting a trained NeRF model into a faster representation for inference [HSM*21, YLT*21]. Recent methods achieve impressive frame rates with only a small quality loss via triangle mesh baking [YHR*23, RGS*24] or distilling of state-of-the-art NeRF models into memory-efficient, hybrid representations [DHR*24]. While they render exceptionally fast even on low-end mobile devices, a major disadvantage is their reliance on the NeRF model that needs to be trained before the – often equally expensive – baking process. Other approaches that proved effective directly optimize a triangle mesh [CFHT23] or a set of tetrahedra [KS23].

2.2. Gaussian Splatting

In early work on point-based rendering, points were rasterized as opaque splats [GD98, RL00, PZvBG00] and thus suffered from strong aliasing. It was shown that quality can be significantly improved using semitransparent splats [ZPvBG01b], however this requires costly sorting and blending operations. To address performance issues, fast algorithms for then GPUs [BHZK05] or dedicated hardware implementations [WHA*07] were proposed.

More recently, point-based rendering regained momentum as a powerful primitive for differentiable rendering in novel view synthesis [KPLD21, ASK*20, RFS22]. In 2023, Kerbl et al. [KKLD23] re-introduced rasterization-based rendering of point primitives with anisotropic Gaussian extent in their seminal work *3D Gaussian Splatting (3DGS)*. Utilizing fast, tile-based rasterization [LZ21], 3DGS optimizes a set of explicit 3D Gaussians via gradient descent and an adaptive density control mechanism. Due exceptionally fast rendering during inference, 3DGS has inspired a plethora of follow-up research addressing, e.g., anti-aliasing [YCH*24], compression [BKL*24], dynamic scenes [LKLR24, WYF*24], large scenes [KMK*24], and the reliance on an initial point cloud [KRS*24, NMR*25]. Especially relevant to this work is *StopThePop* by Radl et al. [RSP*24], which employs a hierarchical sorting approach to reduce popping artifacts originating from approximate sorting used in 3DGS. Notably, the per-pixel ordering resulting from this hierarchical sorting approach is not guaranteed to be fully correct. We also present a method for avoiding these popping artifacts, but achieve this by changing the blending procedure to a hybrid transparency-based approach so that our method only requires partial depth-ordering of the initial K splats contributing to each pixel color. Thus, we avoid the need for a sophisticated sorting implementation which, even when compared to the approximate solution used by 3DGS, improves performance. Equally important in the context of this work is the perspectively accurate *2D Gaussian Splatting (2DGS)* by Huang et al. [HYC*24] as well as concurrent work that introduces ray tracing into the 3DGS framework [YSG24, MLMP*24]. While the latter also provide a perspectively accurate rendering formulation by using ray tracing, they introduce a significant slowdown during both the optimization and subsequent inference. Furthermore, their evaluation of the 3D Gaussian primitives relies on matrix inversion, which is prone to numerical instabilities when Gaussians fall flat along one or more principal axes. The perspectively correct approach by Huang et al. [HYC*24] circumvents these numerical instabilities by utilizing an optimized approach for 2D, i.e., degenerate 3D Gaussians [SWBG06, WHA*07]. Regarding the perspectively correct rendering of non-degenerate 3D Gaussians, projecting each

Gaussian onto a tangential plane of the unit sphere around the camera [HBG*24] works well but significantly slows down optimization and subsequent inference. In this work, we present an approach that enables perspectively correct rendering of 3D Gaussians through ray-splat intersection: we extend the perspectively correct approach of Weyrich et al. [WHA*07] to work with non-degenerate 3D Gaussians, which enables the use of ray-splat intersection for rasterization-based rendering with negligible overhead and no issues with respect to numerical instabilities.

2.3. Order-Independent Transparency

The seminal Porter-Duff algorithm [PD84] blends two semi-transparent surfaces correctly, but extending it to more than two requires costly sorting of the input primitives, and even produces wrong results in case of intersecting primitives. Several techniques have been developed relying on even more costly per-pixel sorting to circumvent this problem, either by sorting explicitly [Car84] or implicitly using multiple render passes [Eve01, BM08]. These algorithms are in general invariant to the order of input primitives and produce correct results, and are therefore called *exact order-independent transparency (OIT)* methods. A good overview can be found in [Wym16]. A plethora of approaches try to approximate exact OIT by avoiding the costly sorting step. Some of them are not truly order-independent: the outcome still varies slightly depending on the order of input primitives [BCL*07, ESSL10, SML11, SV14]. True, *approximate OIT* algorithms generally modify the blending operations to achieve order-invariance. Ignoring the order-dependent parts of the alpha blending formula results in the "weighted sum" operator [Mes07], which works well with surfaces of small opacity but becomes increasingly inaccurate for more opaque surfaces, causing over-darkening or over-brightening. The weighted blended order-independent transparency (WBOIT) operator avoids the over-darkening by replacing the opacity and surface colors with opacity-weighted averages and introducing a monotonic decreasing depth weighting factor, heuristically reducing the influence of distant surfaces [MB13]. This approach works well for surfaces of low opacity and similar color, like smoke, but becomes problematic for almost opaque surfaces, and the weighting function needs to be tuned for every scene to achieve optimal results. Extending this approach to rendering the surfaces into layers and applying the WBOIT operator to each before blending the resulting colors in order provides an interesting trade-off between quality and efficiency [FEE21]. Moment-based transparency techniques remedy some of the shortcomings of WBOIT by computing moments of the transmittance which are used in a second pass to reconstruct an approximate transmittance function, that weighs the influence of each surface on the final color [MKKP18]. Sharp changes in the transmittance function are not well captured at a lower number of power moments, whereas higher numbers decrease the computational efficiency. As the influence of later surfaces is generally less than the first surfaces due to decreasing transmittance, another, truly order-independent approach is *hybrid transparency* that blends the first K surfaces correctly (including sorting), whereas the remaining surfaces are blended without sorting [MCTB13]. In this work, we integrate this hybrid transparency approach into the 3DGS framework and extend the formulation proposed by Maule et al. to improve its behavior within a gradient descent-based optimization scheme.

3. Method

We introduce the employed scene representation and then describe the two parts of our contribution.

3.1. Preliminaries

Following Kerbl et al. [KKLD23], we represent the scene using a set of 3D points with anisotropic Gaussian extent as a proxy for geometry. Cutting each Gaussian at 3σ results in ellipsoid-shaped splats. Each splat is defined by its 3D mean $\boldsymbol{\mu} \in \mathbb{R}^3$, a scalar opacity value $o \in \mathbb{R}$, three principal tangential vectors $\boldsymbol{t}_u, \boldsymbol{t}_v, \boldsymbol{t}_w \in \mathbb{R}^3$ modeling its orientation, and three scalars $s_u, s_v, s_w \in \mathbb{R}$ modeling its scale. Note that $\boldsymbol{t}_u, \boldsymbol{t}_v, \boldsymbol{t}_w$ represent a 3D rotation, which allows modeling these parameters using a quaternion $q \in \mathbb{R}^4$ that is easier to optimize via gradient descent. For rendering, we obtain a valid opacity value in $[0, 1]$ using a Sigmoid activation, represent s_u, s_v, s_w in logarithmic space, and ensure q is normalized. The view-dependent color representation based on spherical harmonics (SH) is identical to 3DGS, i.e., each point uses SH up to degree 3 resulting in 48 coefficients total.

For any point $\boldsymbol{x} \in \mathbb{R}^3$, we can obtain an α value for a given 3D Gaussian by multiplying its opacity with the value of the Gaussian at that point:

$$\alpha = o \cdot \mathrm{e}^{-\frac{1}{2}\rho(\boldsymbol{x})^2} \quad \text{using} \quad \rho(\boldsymbol{x})^2 = (\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu}), \tag{1}$$

where Σ^{-1} can easily be computed from $(\boldsymbol{t}_u, \boldsymbol{t}_v, \boldsymbol{t}_w)$ and (s_u, s_v, s_w) (cf. Kerbl et al. [KKLD23]). These α values are then used in conjunction with the view-dependent RGB color $\boldsymbol{c} \in \mathbb{R}^3$ when computing per-pixel colors through standard depth-sorted alpha blending of all N contributing splats:

$$C = \sum_{i=1}^{N} \alpha_i T_i \boldsymbol{c}_i, \quad \text{with} \quad T_i = \prod_{j=1}^{i-1} (1-\alpha_j). \tag{2}$$

3.2. Accurate Splat Bounding and Evaluation

In 3DGS, Kerbl et al. [KKLD23] project a 3D Gaussian onto the image plane using the Jacobian of the affine approximation of the projective transformation [ZPvBG01a]. The result is a 2D Gaussian that can easily be evaluated for each pixel and also allows for easy construction of an axis-aligned bounding box around the center (e.g., at 3σ) in screen space for accelerated computation. However, this approach is not fully perspective-accurate, meaning it introduces artifacts when splats are viewed from certain angles (see Fig. 2).

In this work, we propose a solution for this problem that builds upon an established approach for perspectively accurate rendering of ellipsoidal surfaces by Sigg et al. [SWBG06] and the respective optimizations by Weyrich et al. [WHA*07] for 2D Gaussian splats. It should be noted that these approaches elegantly avoid the matrix inversions required by similar approaches (e.g., Eq. (1)), which makes them numerically stable even for degenerate splats, i.e., those where one or more main axes vanish. Notably, Huang et al. [HYC*24] also base their perspectively accurate ray-splat intersection and evaluation on the aforementioned work. However, their approach only works for 2D Gaussians, i.e., degenerate splats. In contrast, we now introduce an approach that enables perspectively accurate 3D splat evaluation

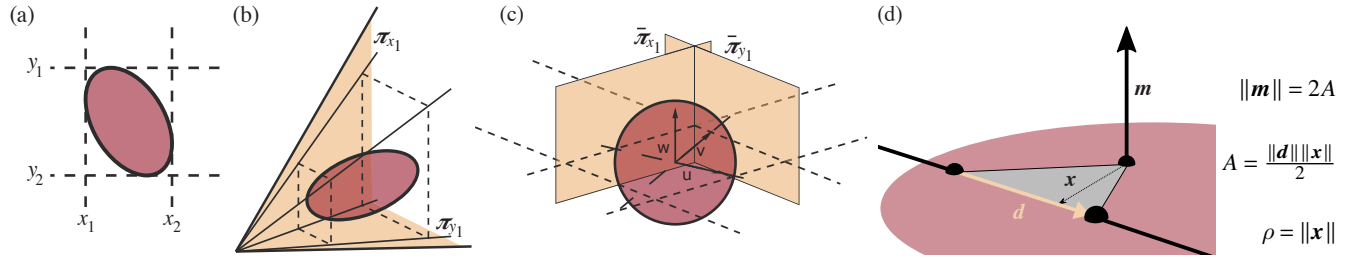

Figure 4: *The perspectively correct screen-space bounding box of a splat (a) is given by the projection of its bounding frustum in view space (b). When transformed into local splat coordinates, the frustum planes align with tangential planes of the unit sphere (c). Our approach for splat evaluation along viewing rays makes use of the Plücker coordinate representation ($\boldsymbol{d} : \boldsymbol{m}$). In local splat coordinates, the point along the ray that maximizes the Gaussian's value corresponds to the point $\boldsymbol{x}$ that minimizes the perpendicular distance $\|\boldsymbol{x}\|$ to the origin (d). Parts (a-c) courtesy of Weyrich et al. [WHA*07]; used with permission.*

along per-pixel viewing rays. Thus, our approach is more general compared to Huang et al., as the 3D Gaussians we use may still degenerate to 2D Gaussians without impacting the optimization or rendering due to the numerical stability of our approach.

We start by describing the approach for computing tight, axis-aligned screen-space bounding boxes for each 3D Gaussian. We define the model-view matrix $M \in \mathbb{R}^{4\times4}$, the projection to clip-space $P \in \mathbb{R}^{4\times4}$, and the viewport transform $V \in \mathbb{R}^{4\times4}$ mapping to window coordinates. Using the per-splat attributes described in Sec. 3.1, the transformation $T' \in \mathbb{R}^{4\times4}$ from the normalized splat space (where the Gaussian's ellipsoid becomes the unit sphere) to screen space is:

$$T' = VPMT, \quad \text{where} \quad T = \begin{pmatrix} | & | & | & | \\ s_u \boldsymbol{t}_u & s_v \boldsymbol{t}_v & s_w \boldsymbol{t}_w & \boldsymbol{\mu} \\ | & | & | & | \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

is the transformation from normalized splat space to world space. Using Weyrich et al.'s optimization of Sigg et al.'s bounding box computation [SWBG06], extended to non-degenerate (full 3D) Gaussians, the desired tight 3D bounding box in screen space is $[b_1,t_1]\times[b_2,t_2]\times[b_3,t_3]$ (bottom and top values, b_i and t_i, for $i \in \{1,2,3\}$ the x, y, and z coordinate, respectively) is:

$$b_i = p_i - h_i \tag{4}$$

$$t_i = p_i + h_i\,, \tag{5}$$

with

$$s = \langle(\rho_c,\rho_c,\rho_c,-1), T'_4 \odot T'_4\rangle\,, \quad \rho_c = 2\ln\frac{o}{\tau_\alpha}\,, \tag{6}$$

$$\boldsymbol{f} = \frac{1}{s}(\rho_c,\rho_c,\rho_c,-1)\,, \tag{7}$$

$$p_i = \langle \boldsymbol{f}, T'_i \odot T'_4\rangle\,, \tag{8}$$

$$h_i = \sqrt{p_i^2 - \langle \boldsymbol{f}, T'_i \odot T'_i\rangle}\,, \tag{9}$$

and T'_i the i-th row of T', $\odot$ denoting component-wise multiplication, and the dot product denoted by $\langle\cdot,\cdot\rangle$. A key advantage of this calculation, for which we provide a visual explanation in Fig. 4, is that it offers a closed-form solution without the need for matrix inversion and regardless of whether the ellipsoid is degenerate in any direction. Following Radl et al. [RSP*24], we also compute an individual cutoff value for each splat to obtain smaller bounding boxes when their opacity value is less than one based on τ_α, a hyperparameter defining the minimum α value of a fragment for it not to be skipped during blending (see Eq. (1)).

Next, we seek to compute the value of the 3D Gaussian for a given viewing ray. In computer graphics, ray intersections with deformed primitives, such as our 3D splats, are an established practice since the dawn of ray tracing, generally following the approach of inversely transforming a viewing ray into the undeformed space. In our case, re-using quantities of the bounding-box setup above, this would mean transforming the ray by $(VPMT)^{-1}$ into splat space, where the distance of the transformed ray to the origin yields ρ, Eq. (1). In order to avoid the matrix inversion that we successfully sidestepped during bounding-box calculation, we once again follow Weyrich et al. and represent a viewing ray through a pixel at (x_s, y_s) as two perpendicular planes $\boldsymbol{\pi}_x = (1,0,0,-x_s)^\top$ and $\boldsymbol{\pi}_y = (0,1,0,-y_s)^\top$ in screen space. Note that we follow the common convention of representing planes as homogeneous vectors $\boldsymbol{p} = (a,b,c,d)^\top$, so that a point $\boldsymbol{x}$ lies on the plane if and only if $\boldsymbol{p}^\top\boldsymbol{x} = 0$. In contrast to 3D points, transforming those planes by $(VPMT)^{-1}$ into planes $\boldsymbol{\pi}^{\mathrm{s}}_{x/y}$ in splat space requires the inverse-transposed mapping, that is:

$$\boldsymbol{\pi}^{\mathrm{s}}_{x/y} = \left((VPMT)^{-1}\right)^{-\top}\boldsymbol{\pi}_{x/y} = (VPMT)^\top\boldsymbol{\pi}_{x/y}\,, \tag{10}$$

once again sidestepping inversion. Next up, intersection of $\boldsymbol{\pi}^{\mathrm{s}}_x$ and $\boldsymbol{\pi}^{\mathrm{s}}_y$ yields the viewing ray in splat space, and its distance to the origin is ρ. In contrast to Weyrich et al., however, ρ cannot quite as easily be computed, due to the non-degeneracy of our 3D Gaussians. Instead, we employ the dual Plücker coordinate representation that derives the mapped viewing ray $\mathcal{L}^*_{\mathrm{s}}$ from the intersection of $\boldsymbol{\pi}^{\mathrm{s}}_{x/y}$ from their coefficients $\boldsymbol{\pi}^{\mathrm{s}}_x =: (a^1 : a^2 : a^3 : a^0)$, and $\boldsymbol{\pi}^{\mathrm{s}}_y =: (b^1 : b^2 : b^3 : b^0)$, respectively:

$$\mathcal{L}^*_{\mathrm{s}} = (p^{23} : p^{31} : p^{12} : p^{01} : p^{02} : p^{03})\,, \tag{11}$$

$$\text{with} \quad p^{ij} = \begin{vmatrix} a^i & a^j \\ b^i & b^j \end{vmatrix} = a^i b^j - a^j b^i\,. \tag{12}$$

The line's dual Plücker coordinates $\mathcal{L}^*_{\mathrm{s}}$ are numerically equivalent to its primal Plücker coordinates $\mathcal{L}_{\mathrm{s}}$ up to some common scaling

factor λ:

$$\mathcal{L}_s =: (\boldsymbol{d} : \boldsymbol{m}) = (p_{01} : p_{02} : p_{03} : p_{23} : p_{31} : p_{12}) \quad (13)$$

$$= (\lambda p^{23} : \lambda p^{31} : \lambda p^{12} : \lambda p^{01} : \lambda p^{02} : \lambda p^{03}) \quad (14)$$

where $\boldsymbol{d}$ is the ray direction in splat space and $\boldsymbol{m}$ is its moment around the origin (see Fig. 4 for a visualization). From the known equality $distance(\mathcal{L}_s, origin) = \frac{\|\boldsymbol{m}\|}{\|\boldsymbol{d}\|}$, it follows that:

$$\rho(\boldsymbol{x})^2 = \frac{\|\lambda(p^{01}, p^{02}, p^{03})^\top\|^2}{\|\lambda(p^{23}, p^{31}, p^{12})^\top\|^2} = \frac{\left\| \begin{matrix} a^0 b^1 - a^1 b^0 \\ a^0 b^2 - a^2 b^0 \\ a^0 b^3 - a^3 b^0 \end{matrix} \right\|^2}{\left\| \begin{matrix} a^2 b^3 - a^3 b^2 \\ a^3 b^1 - a^1 b^3 \\ a^1 b^2 - a^2 b^1 \end{matrix} \right\|^2} . \quad (15)$$

Note that we directly compute $\rho(\boldsymbol{x})^2$ for Eq. (1). This formulation is numerically stable, as the only potential issue – a vanishing denominator in Eq. (15) – can easily be detected and corresponds to the ray missing the splat. Conveniently, the obtained value corresponds to the point along the pixel's viewing ray where the Gaussian has the highest value, essentially making it equivalent to the numerically unstable and slower approaches used in concurrent works [YSG24, MLMP*24]. Compared to Kerbl et al. [KKLD23], our calculations incur reduced setup costs for each Gaussian but slightly higher costs per pixel, which is in fact a desirable tradeoff whenever graphics primitives cover only few pixels [MBDM97]. Moreover, the added computations are exclusively additions and multiplications, except for the division in Eq. (15), meaning they are both fast and trivial to differentiate.

3.3. Temporally-Stable Rendering via Hybrid Transparency

In 3DGS, Kerbl et al. [KKLD23] use a tile-based rasterization approach to efficiently render images from a set of 3D Gaussian primitives. To achieve robust optimization via gradient descent, they compute per-pixel color by applying standard α-blending, Eq. (2), in depth-sorted order. In this work, we propose to integrate a hybrid-transparency approach for blending into the 3DGS framework. This is motivated by three observations: (1) The global sorting scheme proposed by Kerbl et al. [KKLD23] sorts splats as a whole, leading to incorrectly resolved splat-splat intersections and temporal incoherence ("popping") during rendering and optimization; resolving these artifacts through a full sort of all 3D Gaussian pixel fragments, however, is prohibitively slow and thus approximate sorting is required (cf. Radl et al. [RSP*24]). (2) As we will demonstrate in our experiments, using "full" OIT, which forgoes sorting completely, works well during optimization and even improves background reconstruction but results in semi-transparent foreground rendering, which is clearly undesirable for novel-view synthesis and reconstruction methods. (3) Blending only the first K contributions for each pixel has been shown to work well in differentiable rendering frameworks while also much faster than approaches that require a complete sort [LZ21, FRFS24].

The hybrid-transparency approach [MCTB13] splits up the per-pixel color computation into two parts: The *core* that uses standard α-blending to blend the first K contributions for each pixel, including sorting. All remaining contributions are combined into a *tail*, that is blended without sorting. To decide whether a splat is one of the first K, we need to compute a pixel-specific depth value for each of them, i.e., the z-value of the point of evaluation $\boldsymbol{x}$ (see Fig. 4) in view space. Re-using the intermediate values of our splat evaluation (see Sec. 3.2), $\boldsymbol{x}_{\text{view}}$ can be computed efficiently by

$$\boldsymbol{x}_{\text{view}} = MT \frac{\boldsymbol{d} \times \boldsymbol{m}}{\|\boldsymbol{d}\|^2} . \quad (16)$$

As in Sec. 3.2, $\boldsymbol{x}_{\text{view}}$ is the splat's point of maximum contribution along the pixel ray, which again makes our computation compatible with the numerically unstable and slower approaches used in recent work [RSP*24]. In our implementation, we keep track of the first K contributions and accumulate all of the $(N-K)$ contributions for each pixel to then compute its color as

$$C = \sum_{i=1}^{K} \alpha_i T_i \boldsymbol{c}_i + T_{K+1} \Big((1 - T_{\text{tail}}) \boldsymbol{c}_{\text{tail}} + T_{\text{tail}} \boldsymbol{c}_{\text{bg}} \Big) , \quad (17)$$

with

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \quad \boldsymbol{c}_{\text{tail}} = \frac{\sum_{i=K+1}^{N} \alpha_i \boldsymbol{c}_i}{\sum_{i=K+1}^{N} \alpha_i}, \quad T_{\text{tail}} = \prod_{i=K+1}^{N} (1 - \alpha_i) . \quad (18)$$

Notably, our computation of T_{tail} is more accurate compared to the average-based formulation of Maule et al. [MCTB13]. As the partial derivative of the blending function is the same for all splats inside the tail, our approach allows pre-computing all required partial derivatives for each pixel during the forward pass. Thus, our backward pass can be implemented very efficiently.

4. Experiments

We conduct multiple experiments on established datasets to validate our approach.

4.1. Setup

Following recent work, we evaluate on the established Mip-NeRF360 [BMV*22] and Tanks and Temples [KPZK17] datasets, which provide a diverse set of 17 real scenes with varying challenges. We use all nine scenes from the Mip-NeRF360 dataset as well as all eight scenes from the *intermediate* set of the Tanks and Temples dataset resulting in a total of 17 scenes. This allows us to evaluate our method on a broad spectrum of challenges regarding both geometric and photometric aspects. As is common practice, we use the $4\times$ / $2\times$ downscaled images for the outdoor/indoor scenes of the Mip-NeRF360 dataset. For the Tanks and Temples scenes, we use the full-resolution images to evaluate performance at full HD resolution. To compute meaningful quality metrics, we use the established 7:1 train/test split [BMV*22] for all scenes. Central to our evaluation is the comparison against the original 3D Gaussian Splatting [KKLD23]. Furthermore, we compare against the recent StopThePop [RSP*24], 2D Gaussian Splatting [HYC*24], and Huang et al.'s approach [HBG*24], as they also provide solutions for the problems addressed in this work. For StopThePop, we report results of both model variants the authors propose in their paper [RSP*24]. We also compare against Gaussian Opacity Fields

Method	Mip-NeRF360 [BMV*22] (Img. Res. ≈ 1MP–1.5MP)						Tanks&Temples [KPZK17] (Img. Res. ≈ 2MP)					
	SSIM↑	PSNR↑	LPIPS↓	Training	Render	#Splats	SSIM↑	PSNR↑	LPIPS↓	Training	Render	#Splats
Zip-NeRF [BMV*23]	0.828	28.56	0.219	5h	5s	N/a	0.878	26.75	0.233	5h	10s	N/a
3DGS [KKLD23]	0.814	27.20	0.254	18m40s	3.2ms	3.31M	0.866	25.27	0.276	20m18s	4.6ms	1.50M
StopThePop [RSP*24]	0.814	27.30	0.252	20m11s	4.7ms	3.28M	0.866	24.99	0.270	22m03s	4.8ms	1.49M
↪ w/ opacity decay	0.809	27.08	0.267	12m02s	2.5ms	1.71M	0.862	24.98	0.284	14m22s	3.0ms	0.81M
2DGS [HYC*24]	0.795	26.81	0.297	21m08s	7.4ms	2.01M	0.851	24.55	0.314	28m45s	10.4ms	0.95M
Huang et al. [HBG*24]	0.814	27.41	0.257	1h01m	10.7ms	3.30M	0.866	25.19	0.276	1h13m	15.7ms	1.49M
GOF [YSG24]	0.821	27.27	0.238	1h21m	60.0ms	2.91M	0.866	25.01	0.274	1h35m	81.3ms	1.26M
Ours	0.822	27.17	0.234	10m07s	2.2ms	2.19M	0.866	24.62	0.272	8m33s	2.6ms	0.81M

Table 1: *Quantitative comparisons on the Mip-NeRF360 and Tanks and Temples datasets. Our approach of using perspectively correct splat evaluation in combination with hybrid transparency significantly reduces training and rendering times with image quality being similar to the baselines'. Excluding Zip-NeRF, the three best results are highlighted in* ***green*** *in descending order of saturation.*

(GOF) [YSG24], a very recent work that uses ray tracing-based volume rendering. For all baselines, we use the official implementation to compute the results for all 17 scenes and use the same script for all quality metric computations. We use a single RTX 4090 for all experiments and measure frame rate by rendering the test-set images of each scene 100 times and averaging over the results [DHR*24].

4.2. Implementation and Optimization

We implement our method from scratch in PyTorch and CUDA using the implementation of Kerbl et al. [KKLD23] as a reference. While the optimization pipeline and hyperparameters are mostly the same as in 3DGS, we make multiple modifications to solve technical challenges arising from our changes to the rendering pipeline. First, we tackle aliasing problems by employing our own fast CUDA implementation of the 3D filter proposed by Yu et al. [YCH*24] as our inversion-free splat evaluation prevents use of the screen-space low-pass filter in EWA splatting [ZPvBG01a]. As the screen-space gradients 3DGS uses for densification are not available with our splat evaluation scheme, we follow Moenne-Loccoz et al. and instead directly use the gradient of the splat means scaled by half the respective distance to the camera in view space [MLMP*24]. To prevent over-densification, we replace the opacity reset with a decay strategy [RSP*24] that multiplies splat opacity by $\lambda_o = 0.9995$ every 50 iterations during densification. Furthermore, we adapt the densification changes from GOF to our approach [YSG24] and apply visibility score-based pruning as proposed by Niemeyer et al. [NMR*25]. As we will show in our experiments, these changes are necessary to adjust 3DGS's densification strategy to work with our ray-splat intersection-based approach. For our blending formulation, we use a core with $K = 16$ and employ a second threshold $\tau_K = 0.05$ specifying the minimum α of a fragment to be considered for the core. As the insertion sort that we employ for keeping track of per-pixel core causes significant divergence, we reduce the tile size to 8×8 in our implementation. Lastly, we make use of an optimized SSIM implementation [MGK*24] to speed up loss computation.

4.3. Results

The quantitative results in Tab. 1 show that our approach provides significantly faster training and rendering. Compared to 3DGS [KKLD23], we achieve a training speedup of 2.1× and render 1.6× faster on average. The second fastest approach is StopThePop [RSP*24] with opacity decay enabled, which benefits from a large reduction in the number of splats. In terms of image quality, we observe that our method achieves about equal results across the board. However, all 3DGS-based approaches are clearly outperformed by Zip-NeRF [BMV*23]. Taking our contributions into consideration, we observe that our approach outperforms approaches with perspectively correct splat evaluation. Compared to 2DGS [HYC*24], we achieve better results across the board reassuring that 3D Gaussian splats are more expressive than the 2D Gaussian surfels used by 2DGS. Huang et al.'s projection method [HBG*24] significantly slows down training and rendering while achieving image quality similar to that of 3DGS. GOF [YSG24] achieves slightly higher image quality but requires longer training and fails to render in real-time. It should be noted that 2DGS and GOF were fine-tuned for mesh extraction after optimization, which is an interesting topic but not a focus of this work. Seeing that we achieve similar image quality but faster training and rendering compared to either configuration of StopThePop [RSP*24], the results confirm that our approach is a valid alternative to prevent popping artifacts for rasterization-based 3D Gaussian splat rendering.

We also show visual comparisons for multiple scenes in Fig. 5. Beyond those comparisons, we often observe more accurate reconstruction of the background when inspecting trained models. This is likely caused by our perspectively correct splat evaluation that allows better use of the multi-view signal provided by the training images where the background is generally located near image borders where perspective distortion has its highest influence.

Performance Breakdown. In Tab. 2, we compare how parts of the rendering influence the total runtime. We denote the creation of per-tile primitive lists (cf. Kerbl et al. [KKLD23]) as *tiling* and per-tile color computations as *blending*. The speed of 3DGS is equally limited by tiling and blending. StopThePop [RSP*24] implements significant optimizations for the tiling step in an attempt to reduce the overhead of their hierarchical sorting during blending, which is the main bottleneck. In contrast, we use a 16-bit unsigned integer as the key for each Gaussian/tile instance, as we do not require any global depth-sorting. This speeds up tiling by a similar amount as the revised culling strategy of StopThePop. Regarding the blending step, our hybrid transparency approach requires only partial depth-ordering making it significantly faster than the approach of StopThePop.

Figure 5: *Visual comparisons of baselines that allow for real-time rendering.*

Perspectively Correct Splat Evaluation. As previously discussed, our contributions can be integrated into 3D Gaussian splatting implementations independently. Our hybrid transparency approach (Sec. 3.3) is mainly a method for addressing popping artifacts without sacrificing computational efficiency. However, our perspectively accurate splat evaluation (Sec. 3.2) should prove beneficial across a much broader range of applications due to its accuracy. As can be seen in Tab. 3, using it alongside the approximately depth-ordered alpha blending from 3DGS [KKLD23] improves results for all image quality metrics. Note that a major portion of the speed increase over 3DGS comes from our approach providing tight screen-space bounds for each splat. These significantly reduce the amount of global memory accesses, sorting overhead, and the number of splats processed during blending.

Hybrid Transparency Ablations. To analyze the behavior of our hybrid transparency-based blending, we perform ablations on the nine scenes from the Mip-NeRF360 dataset [BMV*22]. The quantitative results shown in Tab. 4 clearly indicate that $K = 16$ is the optimal choice for our approach. Note that rendering is slightly slower with $K = 8$ as the number of splats created during optimization increases as core size decreases.

We also show visual comparisons in Fig. 6 showing that blending without any sorting is not a valid option as it causes the foreground to become transparent. However, we do find that background reconstruction seems to improve in this case. With our default configuration of $K = 16$, omitting tail computation during inference causes a small drop in image quality, which is most noticeable in the sky. Importantly, not using the tail during training causes catastrophic failure.

Densification Ablations. As described in Sec. 4.2, we make multiple adjustments to the adaptive density control mechanism proposed by Kerbl et al. [KKLD23]. The ablations in Tab. 4 clearly show why this is necessary for our ray-splat intersection-based approach. Not scaling the gradient of the splat means by half the respective distance to the camera [MLMP*24] results in significant under-reconstruction, whereas dropping either the 3D filter [YCH*24] or using the standard opacity reset [KKLD23] every 3,000 iterations instead of an opacity decay [RSP*24] results in the optimization running out of memory due to over-reconstruction. Furthermore, the visibility score-based pruning [NMR*25] simultaneously improves quality and speed as it helps to prevents over-reconstruction by pruning low-opacity splats, which would otherwise cause overfitting.

Method	Preprocess	Tiling	Blending	Total	#Splats
	Bicycle [BMV*22]				
3DGS [KKLD23]	1.099	3.494	3.422	8.015	6.03M
StopThePop [RSP*24]	1.364	1.358	5.116	7.838	6.05M
↪ w/ opacity decay	**0.554**	**0.547**	2.442	3.543	**2.91M**
2DGS [HYC*24]	0.674	1.909	5.267	7.850	3.99M
Ours	0.556	0.754	**2.007**	**3.317**	4.35M
	Francis [KPZK17]				
3DGS [KKLD23]	0.235	2.078	1.644	3.957	0.81M
StopThePop [RSP*24]	0.268	0.423	2.869	3.560	0.82M
↪ w/ opacity decay	0.154	**0.310**	1.980	2.444	0.45M
2DGS [HYC*24]	0.177	4.397	8.700	13.27	0.51M
Ours	**0.107**	0.678	**1.228**	**2.013**	**0.44M**

Table 2: *Breakdown of render timings in milliseconds at a resolution of 1920×1080 pixels measured on an RTX 4090 GPU. We select the scenes where the trained 3DGS model contains the highest (Bicycle) and lowest (Francis) number of splats across all tested scenes. For each method, we repeatedly render all test-set views and average measurements obtained via NVIDIA Nsight Systems across 100 runs.*

Method	SSIM↑	PSNR↑	LPIPS↓	Training	Render
3DGS	0.730	24.64	0.265	21m17s	3.41ms
Ours	0.742	24.62	0.237	**12m09s**	**2.08ms**
Ours w/o per-pixel HT	**0.748**	**24.81**	**0.230**	17m26s	2.23ms

Table 3: *We showcase the potential of our perspectively correct splat evaluation by omitting our hybrid transparency-based blending in favor of the approximately depth-ordered alpha blending from 3DGS [KKLD23]. As it forgoes per-pixel ordering, this version of our model is subject to popping artifacts, and it is slightly less efficient than our default configuration, but outperforms 3DGS across all metrics. The reported values are averaged results for the five outdoor scenes from the Mip-NeRF360 dataset [BMV*22].*

5. Discussion

The experiments validate that our approach improves robustness and accuracy of the optimization due to our perspective-accurate and numerically stable splat evaluation. Similarly, they show that hybrid transparency can effectively be integrated into differentiable rasterizers for 3D Gaussian splats to accelerate training and rendering without sacrificing image quality. Importantly, both of our contributions improve the rendering quality beyond what shows in the quantitative results for standardized benchmarks. To address this, our supplemental includes two videos that clearly demonstrate the advantage of our two-part contribution, which we encourage the reader to view. The perspectively correct splat evaluation allows our rendering to avoid artifacts that arise from the approximate projection used in 3DGS [KKLD23]. We find that the commonly used benchmark datasets largely fail to reveal this artifact making it invisible in quantitative evaluation. However, when inspecting trained 3D Gaussian splat models in a real-time viewer, it is very easy to find scenarios where the perspectively inaccurate projection causes disturbing artifacts.

Configuration	SSIM↑	PSNR↑	LPIPS↓	Training	Render	#Splats
	Hybrid Transparency					
$K = 8$	0.811	26.70	0.245	**10m02s**	2.28ms	2.45M
$K = 32$	0.821	27.12	0.236	31m11s	2.67ms	**2.10M**
Ours ($K = 16$)	**0.822**	**27.17**	**0.234**	10m07s	2.19ms	2.19M
Inference w/o tail	0.817	26.92	0.240	10m07s	**2.14ms**	2.19M
	Densification					
No 3D filter†	0.777	25.58	0.303	**1m44s**	2.24ms	3.65M
No opacity decay†	0.784	25.92	0.292	2m42s	3.81ms	4.35M
No scaled grads	0.784	26.30	0.317	4m05s	**1.35ms**	**0.30M**
No vis. pruning	0.820	26.96	0.235	12m17s	2.34ms	2.42M
Ours-7K	0.780	25.72	0.299	1m45s	2.10ms	2.09M
Ours	**0.822**	**27.17**	**0.234**	10m07s	2.19ms	2.19M

Table 4: *Ablations for our hybrid transparency approach and our adaptation of the densification strategy from 3DGS [KKLD23] computed on the nine scenes from the Mip-NeRF360 dataset [BMV*22]. Configurations marked with † are only trained for* 7,000 *iterations as the optimization would otherwise run out of memory due to over-densification.*

Equally as noticeable are the popping artifacts caused by the approximate depth-ordering in 3DGS. StopThePop [RSP*24] uses a hierarchical sorting approach to resolve this issue. They sort splats globally based on their means [KKLD23] and then use a sliding window approach to sort the per-tile lists approximately. In contrast, our approach guarantees correct sorting order for the first K elements in each pixel and then uses fast blending without sorting for the rest. Our results show that this strategy results in a good tradeoff between quality and performance: The correct sorting of the front-most fragments leads to high rendering quality and temporal stability, with little impact on performance. While the remaining elements from the tail are essential for robust optimization as demonstrated in Fig. 6, our results show that the much cheaper unsorted blending is sufficient to maintain high performance, very good rendering quality, and robust optimization.

The artifacts addressed by our approach are most apparent while the camera is moving (see our supplemental videos), which is an issue for VR/AR applications or fast-paced games where the user rarely keeps their head or the camera perfectly still. As VR also requires high frame rates to mitigate cybersickness and view-consistent peripheral rendering to avoid breaking immersion [FFS24], we believe that the combined advantages of our approach – fast, perspective-correct rendering without popping artifacts – make it a great fit for such applications. Another area where we think our approach could prove beneficial are 3DGS implementations for low-end devices, which currently require expensive CPU sorting. As our approach naturally supports depth peeling due to the fixed size of the core, it should be possible to implement a hardware-accelerated viewer for our approach that does not require a dedicated sorting routine.

Regarding reconstruction and image quality, we observe that the adaptive density control mechanism by Kerbl et al. [KKLD23] introduced for 3DGS has a significant influence while being very sensitive to changes. As outlined in Sec. 4.2, approaches evaluating splats along per-pixel viewing rays do not naturally provide the

Figure 6: *Visual comparisons for different model configurations regarding our hybrid transparency approach. Using a smaller core size K causes issues for reflective surfaces, as radiance fields commonly model these using semi-transparency. Disabling the order-independent tail only slightly reduces quality, especially in the sky, whereas not using it during optimization results in catastrophic failure.*

information used by the densification approach of 3DGS, i.e., the gradient of the splat's position in view-space. After experimenting with the solutions presented in recent works [KRS*24, HYC*24, MLMP*24], we find scaling the gradients of the 3D positions by half the distance to the camera [MLMP*24] works best for our approach. Nonetheless, we find that the direct connection between local splat attributes and the global densification scheme introduces issues regarding the robustness of all 3DGS-based approaches we investigated. Additional challenges, such as the photometric variation in images from the Tanks and Temples dataset [KPZK17] further amplify this issue.

Limitations and Future Work. Naturally, our approach is not without limitations. While sole use of alpha blending allows for early stopping based on a preset transmittance threshold [KKLD23], this is not an option for hybrid transparency by default. In our current implementation, we accumulate the RGB color and alpha of all per-pixel contributions that are not part of the core. Intuitively, this is a problem when scenes become larger due to the increase in depth complexity. However, we are confident that this issue can be resolved, e.g., by combining a depth prepass with a depth-weighting function as in WBOIT [MB13]. Additionally, we find that our approach achieves slightly lower PSNR than that of 3DGS, which we think is due to the absence of a screen-space aliasing filter or multi-sampling approach. As suggested by Huang et al. [HYC*24], we tried using the approach of Botsch et al. [BHZK05], but did not find it beneficial. We think it is possible to extend our splat evaluation to efficiently handle lens distortion thus enabling training on images without applying lossy undistortion operations during data pre-processing. Lastly, we would like to highlight that our hybrid transparency approach unveils an avenue for integrating otherwise too expensive ideas into real-time rendering of 3D Gaussians splats. While correct volume rendering of all contributing splats is not computationally feasible, it should be possible for the first $K = 16$ Gaussians. Similarly, we think that the fixed size of the core in our hybrid transparency approach allows the use of more powerful appearance models [DHR*24] to improve image quality.

6. Conclusion

In this paper, we addressed two significant challenges prevalent in 3D Gaussian splat rendering. First, we introduced a novel approach for fast and perspective-accurate splat evaluation, which eliminates artifacts that arise from the affine approximation used by current 3D Gaussian Splatting implementations. Our approach elegantly avoids matrix inversion thus ensuring robust optimization and accurate rendering without numerical instabilities, even when splats degenerate. Second, we proposed an approach for view-consistent rendering through hybrid transparency, which uses fast, approximate per-pixel depth sorting that guarantees correct order for the first K contributions and thus prevents popping artifacts. Importantly, these improvements have a largely positive impact on performance, as evidenced by our analysis of training time, rendering speed, image quality metrics, and visual fidelity. We believe our two-part contribution can both independently and jointly improve accuracy and efficiency of 3D Gaussian splat rendering in the future.

https://fhahlbohm.github.io/htgs/

Acknowledgments

We thank Timon Scholz and Carlotta Harms for their help with comparisons and supplemental material. This work was partially funded by the DFG – projects "Real-Action VR" (ID 523421583) and "Increasing Realism of Omnidirectional Videos in Virtual Reality" (ID 491805996) – as well as the L3S Research Center, Hanover, Germany. Linus Franke was supported by the 5G innovation program of the German Federal Ministry for Digital and Transport under the funding code 165GU103B. Open Access funding enabled and organized by Projekt DEAL.

References

[ASK*20] Aliev K.-A., Sevastopolsky A., Kolos M., Ulyanov D., Lempitsky V.: Neural point-based graphics. In *Eur. Conf. Comput. Vis.* (2020), Springer-Verlag, pp. 696–712. doi:10.1007/978-3-030-58542-6_42. 3

[BCL*07] Bavoil L., Callahan S. P., Lefohn A., Comba J. a. L. D., Silva C. T.: Multi-fragment effects on the gpu using the k-buffer. In *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games* (2007), Association for Computing Machinery, pp. 97–104. doi:10.1145/1230100.1230117. 4

[BHZK05] Botsch M., Hornung A., Zwicker M., Kobbelt L.: High-quality surface splatting on today's gpus. In *Proc. Eurographics/IEEE VGTC Symposium Point-Based Graphics* (2005), IEEE, pp. 17–141. doi:10.1109/PBG.2005.194059. 3, 10

[BKL*24] Bagdasarian M. T., Knoll P., Li Y.-H., Barthel F., Hilsmann A., Eisert P., Morgenstern W.: 3DGS.zip: A survey on 3D Gaussian splatting compression methods, 2024. arXiv:2407.09510. 3

[BM08] Bavoil L., Myers K.: Order independent transparency with dual depth peeling. *NVIDIA OpenGL SDK 1*, 12 (2008), 2–4. 4

[BMT*21] Barron J. T., Mildenhall B., Tancik M., Hedman P., Martin-Brualla R., Srinivasan P. P.: Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Int. Conf. Comput. Vis.* (2021), pp. 5835–5844. doi:10.1109/ICCV48922.2021.00580. 3

[BMV*22] Barron J. T., Mildenhall B., Verbin D., Srinivasan P. P., Hedman P.: Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.* (2022), pp. 5460–5469. doi:10.1109/CVPR52688.2022.00539. 3, 6, 7, 8, 9

[BMV*23] Barron J. T., Mildenhall B., Verbin D., Srinivasan P. P., Hedman P.: Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *Int. Conf. Comput. Vis.* (2023), pp. 19640–19648. doi:10.1109/ICCV51070.2023.01804. 3, 7

[Car84] Carpenter L.: The A-buffer, an antialiased hidden surface method. In *SIGGRAPH* (1984), Association for Computing Machinery, pp. 103–108. doi:10.1145/800031.808585. 4

[CFHT23] Chen Z., Funkhouser T., Hedman P., Tagliasacchi A.: MobileNeRF: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.* (2023), pp. 16569–16578. doi:10.1109/CVPR52729.2023.01590. 3

[CXG*22] Chen A., Xu Z., Geiger A., Yu J., Su H.: TensoRF: Tensorial radiance fields. In *Eur. Conf. Comput. Vis.* (2022), Springer-Verlag, pp. 333–350. doi:10.1007/978-3-031-19824-3_20. 3

[DHR*24] Duckworth D., Hedman P., Reiser C., Zhizhin P., Thibert J.-F., Lučić M., Szeliski R., Barron J. T.: SMERF: Streamable memory efficient radiance fields for real-time large-scene exploration. *ACM Trans. Graph. 43*, 4 (2024). doi:10.1145/3658193. 3, 7, 10

[ESSL10] Enderton E., Sintorn E., Shirley P., Luebke D.: Stochastic transparency. In *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games* (2010), Association for Computing Machinery, pp. 157–164. doi:10.1145/1730804.1730830. 4

[Eve01] Everitt C.: Interactive order-independent transparency. *NVIDIA OpenGL Applications Engineering 2*, 6 (2001), 7. 4

[FEE21] Friederichs F., Eisemann E., Eisemann M.: Layered weighted blended order-independent transparency. In *Proc. Graphics Interface* (5 2021), pp. 1–6. 4

[FFS24] Franke L., Fink L., Stamminger M.: VR-Splatting: Foveated radiance field rendering via 3D Gaussian splatting and neural points. *arXiv preprint arXiv:2410.17932* (2024). 9

[FKYT*22] Fridovich-Keil S., Yu A., Tancik M., Chen Q., Recht B., Kanazawa A.: Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.* (2022), pp. 5491–5500. doi:10.1109/CVPR52688.2022.00542. 3

[FRF*23] Franke L., Rückert D., Fink L., Innmann M., Stamminger M.: VET: Visual error tomography for point cloud completion and high-quality neural rendering. In *SIGGRAPH Asia Conference Papers* (2023), Association for Computing Machinery. doi:10.1145/3610548.3618212. 3

[FRFS24] Franke L., Rückert D., Fink L., Stamminger M.: TRIPS: Trilinear point splatting for real-time radiance field rendering. *Comput. Graph. Forum 43*, 2 (2024). doi:10.1111/cgf.15012. 3, 6

[GD98] Grossman J. P., Dally W. J.: Point sample rendering. In *Rendering Techniques: Proc. of the Eurographics Workshop* (1998), Springer, pp. 181–192. 3

[HBG*24] Huang L., Bai J., Guo J., Li Y., Guo Y.: On the error analysis of 3D Gaussian splatting and an optimal projection strategy. In *Eur. Conf. Comput. Vis.* (2024), Springer-Verlag, pp. 247–263. doi:10.1007/978-3-031-72643-9_15. 4, 6, 7, 8

[HFF*23] Harrer M., Franke L., Fink L., Stamminger M., Weyrich T.: Inovis: Instant novel-view synthesis. In *SIGGRAPH Asia Conference Papers* (2023), Association for Computing Machinery. doi:10.1145/3610548.3618216. 3

[HFK*25] Hahlbohm F., Franke L., Kappel M., Castillo S., Eisemann M., Stamminger M., Magnor M.: INPC: Implicit neural point clouds for radiance field rendering. In *International Conference on 3D Vision* (2025). 3

[HSM*21] Hedman P., Srinivasan P. P., Mildenhall B., Barron J. T., Debevec P.: Baking neural radiance fields for real-time view synthesis. In *Int. Conf. Comput. Vis.* (2021), pp. 5855–5864. doi:10.1109/ICCV48922.2021.00582. 3

[HYC*24] Huang B., Yu Z., Chen A., Geiger A., Gao S.: 2D Gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH* (2024), Association for Computing Machinery. doi:10.1145/3641519.3657428. 1, 2, 3, 4, 6, 7, 8, 9, 10

[KKLD23] Kerbl B., Kopanas G., Leimkuehler T., Drettakis G.: 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph. 42*, 4 (2023). doi:10.1145/3592433. 1, 2, 3, 4, 6, 7, 8, 9, 10

[KLR*22] Kopanas G., Leimkühler T., Rainer G., Jambon C., Drettakis G.: Neural point catacaustics for novel-view synthesis of reflections. *ACM Trans. Graph. 41*, 6 (2022). doi:10.1145/3550454.3555497. 3

[KMK*24] Kerbl B., Meuleman A., Kopanas G., Wimmer M., Lanvin A., Drettakis G.: A hierarchical 3D Gaussian representation for real-time rendering of very large datasets. *ACM Trans. Graph. 43*, 4 (2024). doi:10.1145/3658160. 3

[KPLD21] Kopanas G., Philip J., Leimkühler T., Drettakis G.: Point-based neural rendering with per-view optimization. *Comput. Graph. Forum 40*, 4 (2021), 29–43. doi:10.1111/cgf.14339. 3

[KPZK17] Knapitsch A., Park J., Zhou Q.-Y., Koltun V.: Tanks and Temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph. 36*, 4 (2017). doi:10.1145/3072959.3073599. 6, 7, 9, 10

[KRS*24] Kheradmand S., Rebain D., Sharma G., Sun W., Tseng Y.-C., Isack H., Kar A., Tagliasacchi A., Yi K. M.: 3D Gaussian splatting as Markov chain monte carlo. In *Adv. Neural Inform. Process. Syst.* (2024). 3, 10

[KS23] Kulhanek J., Sattler T.: Tetra-NeRF: Representing neural radiance fields using tetrahedra. In *Int. Conf. Comput. Vis.* (2023), pp. 18412–18423. doi:10.1109/ICCV51070.2023.01692. 3

[LKLR24] Luiten J., Kopanas G., Leibe B., Ramanan D.: Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. In *International Conference on 3D Vision* (2024), pp. 800–809. doi:10.1109/3DV62453.2024.00044. 3

[LZ21] Lassner C., Zollhöfer M.: Pulsar: Efficient sphere-based neural rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.* (2021), pp. 1440–1449. doi:10.1109/CVPR46437.2021.00149. 3, 6

[MB13] McGuire M., Bavoil L.: Weighted blended order-independent transparency. *Journal of Computer Graphics Techniques 2*, 2 (2013), 122–141. 4, 10

[MBDM97] Montrym J. S., Baum D. R., Dignam D. L., Migdal C. J.: InfiniteReality: a real-time graphics system. In *SIGGRAPH* (1997), ACM Press, pp. 293–302. doi:10.1145/258734.258871. 6

[MCTB13] Maule M., Comba J., Torchelsen R., Bastos R.: Hybrid transparency. In *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games* (2013), pp. 103–118. doi:10.1145/2448196.2448212. 4, 6

[Mes07] Meshkin H.: Sort-independent alpha blending. *GDC Talk* (2007). 4

[MESK22] Müller T., Evans A., Schied C., Keller A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph. 41*, 4 (2022). doi:10.1145/3528223.3530127. 3

[MGK*24] Mallick S. S., Goel R., Kerbl B., Steinberger M., Carrasco F. V., De La Torre F.: Taming 3DGS: High-quality radiance fields with limited resources. In *SIGGRAPH Asia Conference Papers* (2024), Association for Computing Machinery. doi:10.1145/3680528.3687694. 7

[MKKP18] Münstermann C., Krumpen S., Klein R., Peters C.: Moment-based order-independent transparency. *Proc. ACM Comput. Graph. Interact. Tech. 1*, 1 (2018). doi:10.1145/3203206. 4

[MLMP*24] Moenne-Loccoz N., Mirzaei A., Perel O., de Lutio R., Martinez Esturo J., State G., Fidler S., Sharp N., Gojcic Z.: 3D Gaussian Ray Tracing: Fast tracing of particle scenes. *ACM Trans. Graph. 43*, 6 (2024). doi:10.1145/3687934. 2, 3, 6, 7, 8, 10

[MST*20] Mildenhall B., Srinivasan P. P., Tancik M., Barron J. T., Ramamoorthi R., Ng R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis.* (2020), Springer International Publishing, pp. 405–421. doi:10.1007/978-3-030-58452-8_24. 2, 3

[NMR*25] Niemeyer M., Manhardt F., Rakotosaona M.-J., Oechsle M., Duckworth D., Gosula R., Tateno K., Bates J., Kaeser D., Tombari F.: RadSplat: Radiance field-informed Gaussian splatting for robust real-time rendering with 900+ fps. In *International Conference on 3D Vision* (2025). 3, 7, 8

[PD84] Porter T., Duff T.: Compositing digital images. *SIGGRAPH 18*, 3 (1984), 253–259. doi:10.1145/964965.808606. 4

[PZvBG00] Pfister H., Zwicker M., van Baar J., Gross M.: Surfels: surface elements as rendering primitives. In *SIGGRAPH* (2000), ACM Press, pp. 335–342. doi:10.1145/344779.344936. 3

[RFS22] Rückert D., Franke L., Stamminger M.: ADOP: Approximate differentiable one-pixel point rendering. *ACM Trans. Graph. 41*, 4 (2022). doi:10.1145/3528223.3530122. 3

[RGS*24] Reiser C., Garbin S., Srinivasan P., Verbin D., Szeliski R., Mildenhall B., Barron J., Hedman P., Geiger A.: Binary Opacity Grids: Capturing fine geometric detail for mesh-based view synthesis. *ACM Trans. Graph. 43*, 4 (2024). doi:10.1145/3658130. 3

[RL00] Rusinkiewicz S., Levoy M.: QSplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH* (2000), ACM Press, pp. 343–352. doi:10.1145/344779.344940. 3

[RSP*24] Radl L., Steiner M., Parger M., Weinrauch A., Kerbl B., Steinberger M.: StopThePop: Sorted Gaussian splatting for view-consistent real-time rendering. *ACM Trans. Graph. 43*, 4 (2024). doi:10.1145/3658187. 1, 2, 3, 5, 6, 7, 8, 9

[RSV*23] Reiser C., Szeliski R., Verbin D., Srinivasan P., Mildenhall B., Geiger A., Barron J., Hedman P.: MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Trans. Graph. 42*, 4 (2023). doi:10.1145/3592426. 3

[SML11] Salvi M., Montgomery J., Lefohn A.: Adaptive transparency. In *Proc. of the ACM SIGGRAPH Symposium on High Performance Graphics* (2011), Association for Computing Machinery, pp. 119–126. doi:10.1145/2018323.2018342. 4

[SSC22] Sun C., Sun M., Chen H.: Direct Voxel Grid Optimization: Super-fast convergence for radiance fields reconstruction. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.* (2022), pp. 5449–5459. doi:10.1109/CVPR52688.2022.00538. 3

[SV14] Salvi M., Vaidyanathan K.: Multi-layer alpha blending. In *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games* (2014), Association for Computing Machinery, pp. 151–158. doi:10.1145/2556700.2556705. 4

[SWBG06] Sigg C., Weyrich T., Botsch M., Gross M.: Gpu-based ray-casting of quadratic surfaces. In *Proc. Eurographics/IEEE VGTC Symposium Point-Based Graphics* (2006), Eurographics Association, pp. 59–65. doi:10.2312/SPBG/SPBG06/059-065. 2, 3, 4, 5

[WHA*07] Weyrich T., Heinzle S., Aila T., Fasnacht D. B., Oetiker S., Botsch M., Flaig C., Mall S., Rohrer K., Felber N., Kaeslin H., Gross M.: A hardware architecture for surface splatting. *ACM Trans. Graph. 26*, 3 (2007), 90–es. doi:10.1145/1276377.1276490. 2, 3, 4, 5

[WYF*24] Wu G., Yi T., Fang J., Xie L., Zhang X., Wei W., Liu W., Tian Q., Wang X.: 4D Gaussian splatting for real-time dynamic scene rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.* (2024), pp. 20310–20320. doi:10.1109/CVPR52733.2024.01920. 3

[Wym16] Wyman C.: Exploring and expanding the continuum of OIT algorithms. In *Proc. of High Performance Graphics* (2016), Eurographics Association, pp. 1–11. doi:10.5555/2977336.2977338. 2, 4

[YCH*24] Yu Z., Chen A., Huang B., Sattler T., Geiger A.: Mip-Splatting: Alias-free 3D Gaussian splatting. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.* (2024), pp. 19447–19456. doi:10.1109/CVPR52733.2024.01839. 3, 7, 8

[YHR*23] Yariv L., Hedman P., Reiser C., Verbin D., Srinivasan P. P., Szeliski R., Barron J. T., Mildenhall B.: BakedSDF: Meshing neural SDFs for real-time view synthesis. In *ACM SIGGRAPH Conference Papers* (2023), Association for Computing Machinery. doi:10.1145/3588432.3591536. 3

[YLT*21] Yu A., Li R., Tancik M., Li H., Ng R., Kanazawa A.: PlenOctrees for real-time rendering of neural radiance fields. In *Int. Conf. Comput. Vis.* (2021), pp. 5732–5741. doi:10.1109/ICCV48922.2021.00570. 3

[YSG24] Yu Z., Sattler T., Geiger A.: Gaussian Opacity Fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Trans. Graph. 43*, 6 (2024). doi:10.1145/3687937. 2, 3, 6, 7

[ZPvBG01a] Zwicker M., Pfister H., van Baar J., Gross M.: EWA volume splatting. In *Proc. of the Conference on Visualization* (2001), IEEE Computer Society, pp. 29–36. doi:10.5555/601671.601674. 4, 7

[ZPvBG01b] Zwicker M., Pfister H., van Baar J., Gross M.: Surface splatting. In *SIGGRAPH* (2001), Association for Computing Machinery, pp. 371–378. doi:10.1145/383259.383300. 3