When Gaussian Meets Surfel: Ultra-fast High-fidelity Radiance Field Rendering

KEYANG YE, State Key Lab of CAD&CG, Zhejiang University, China
TIANJIA SHAO, State Key Lab of CAD&CG, Zhejiang University, China
KUN ZHOU*, State Key Lab of CAD&CG, Zhejiang University, China

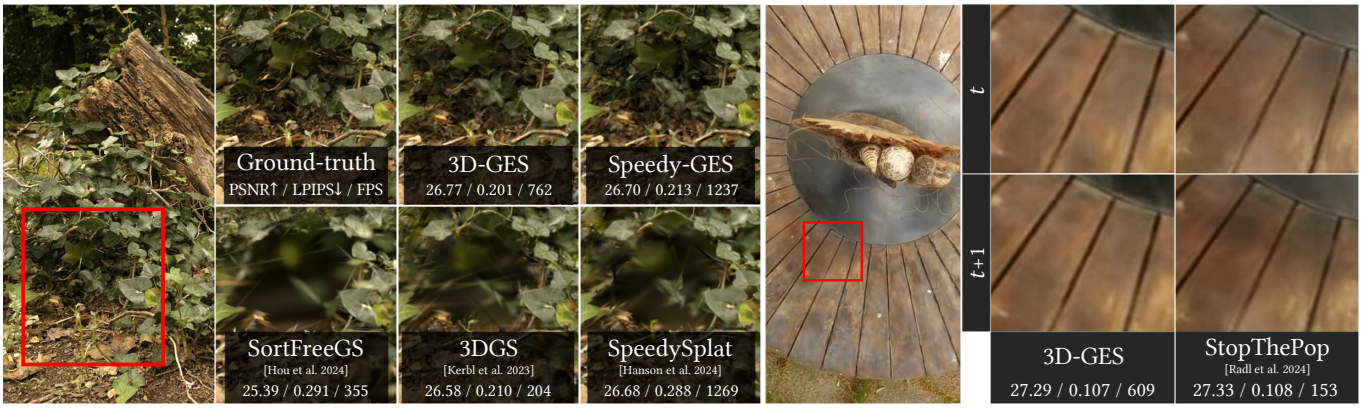

Fig. 1. Our Gaussian-enhanced Surfels (GESs) and its extensions achieve ultra-fast high-fidelity radiance field rendering and successfully avoid popping artifacts under view changes, compared to the state-of-the-art radiance field methods: 3DGS [Kerbl et al. 2023], SpeedySplat [Hanson et al. 2024], SortFreeGS [Hou et al. 2024a] and StopThePop [Radl et al. 2024].

We introduce Gaussian-enhanced Surfels (GESs), a bi-scale representation for radiance field rendering, wherein a set of 2D opaque surfels with view-dependent colors represent the coarse-scale geometry and appearance of scenes, and a few 3D Gaussians surrounding the surfels supplement fine-scale appearance details. The rendering with GESs consists of two passes – surfels are first rasterized through a standard graphics pipeline to produce depth and color maps, and then Gaussians are splatted with depth testing and color accumulation on each pixel order independently. The optimization of GESs from multi-view images is performed through an elaborate coarse-to-fine procedure, faithfully capturing rich scene appearance. The entirely sorting-free rendering of GESs not only achieves very fast rates, but also produces view-consistent images, successfully avoiding popping artifacts under view changes. The basic GES representation can be easily extended to achieve anti-aliasing in rendering (Mip-GES), boosted rendering speeds (Speedy-GES) and compact storage (Compact-GES), and reconstruct better scene geometries by replacing 3D Gaussians with 2D Gaussians (2D-GES). Experimental results show that GESs advance the state-of-the-arts as a compelling representation for ultra-fast high-fidelity radiance field rendering.

CCS Concepts: • **Computing methodologies** → **Rasterization**; **Point-based models**; **Machine learning approaches**; *Rendering*.

Additional Key Words and Phrases: Novel view synthesis, radiance field, Gaussian splatting, point-based rendering, real-time rendering

ACM Reference Format:
Keyang Ye, Tianjia Shao, and Kun Zhou. 2025. When Gaussian Meets Surfel: Ultra-fast High-fidelity Radiance Field Rendering. *ACM Trans. Graph.* 44, 4 (August 2025), 15 pages. https://doi.org/10.1145/3730925

1 INTRODUCTION

Free-view synthesis of 3D scenes from multi-view images has been a long-standing research topic for decades. Starting from the seminal work of light fields [Gortler et al. 1996; Levoy and Hanrahan 1996], various scene representations have been developed for this task, with remarkable advances in recent years, e.g., Neural Radiance Fields (NeRFs) [Mildenhall et al. 2020] and 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023].

Both NeRFs and 3DGS construct volumetric radiance fields of 3D scenes. NeRFs represent scenes as neural volumes of view-dependent color and density, using Multi Layer Perceptrons (MLPs), and synthesize high-quality images through volumetric ray marching, which imposes considerable costs on training and rendering. 3DGS models radiance fields as sparsely distributed 3D Gaussians with view-dependent colors and performs rendering using differentiable rasterization and α-blending, achieving the state-of-the-art

*Corresponding author

Authors' addresses: Keyang Ye, yekeyang@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China; Tianjia Shao, tjshao@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China; Kun Zhou, kunzhou@acm.org, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China.

ACM 0730-0301/2025/8-ART
https://doi.org/10.1145/3730925

visual quality and real-time frame rates at high resolutions. The α-blending requires depth sorting of 3D Gaussians, which is computationally expensive if accurately performed for each pixel. 3DGS approximates the per-pixel sorting by a tile-based pre-sorting of Gaussians for the whole image at a time. While retaining high rendering speeds, such an approximation could produce view-inconsistent images, such as patchy colors appearing and disappearing during view changes, as known as the popping artifacts.

In this paper, we introduce *Gaussian-enhanced Surfels* (GESs), a bi-scale representation for radiance field rendering. At the coarse scale, a set of 2D opaque surfels with view-dependent colors constitute an approximation of the surface radiance field of the scene, called the *surfel radiance field*. Each surfel is a 2D opaque ellipse associated with geometry attributes of position, rotation and scaling, and appearance attributes of spherical harmonics (SH) coefficients. At the fine scale, a few 3D Gaussians surrounding the coarse-scale surfels form a volumetric radiance field to supplement the scene appearance not represented well by the surfel radiance field. Each Gaussian has the same geometry and appearance attributes as in 3DGS [Kerbl et al. 2023]. Our key observation is that the surfel radiance field can capture a major portion of the scene appearance (see Fig. 2, left), while the Gaussian radiance field effectively enhances the surfel radiance field with fine details (see Fig. 2, middle). Combining radiance renderings from both coarse and fine scales, the GES radiance field is able to synthesize high-quality images (see Fig. 2, right), competitive with state-of-the-art methods.

The rendering of GES radiance fields consists of two passes, and is entirely sorting-free. Firstly, the opaque surfels are rasterized through a standard graphics pipeline, producing the color and depth maps. Secondly, we splat the Gaussians to the screen with depth testing, and accumulate the Gaussian color weighted by opacity on each pixel of the surfel color map, in an order-independent way. For each pixel, the Gaussians whose center depths fail to pass the depth testing with the surfel depth map will not accumulate color on the pixel, which means Gaussians are occluded by the geometry represented by the surfels. Such a sorting-free rendering not only bypasses the computation bottleneck of Gaussian sorting and achieves very fast frame rates, but also successfully avoids popping artifacts under view changes (see Fig. 1, right).

The GES representation can be efficiently constructed from multi-view input images through a coarse-to-fine procedure, which first optimizes surfels, and then performs jointly optimization for both surfels and Gaussians. It is a challenging problem to optimize the opaque surfels from the image color loss, because the forward process from the surfel geometry parameters to pixel colors is non-differentiable. That is, the change of surfel geometry yields either no color changes or sudden color changes, as the color is the same everywhere on each surfel. To tackle this challenge, we introduce an opacity modulating parameter during optimization to gradually evolve the translucent surfels into opaque ones. When the parameter is below 1, the whole surfel is translucent with Gaussian distributed opacity. When the parameter increases, the surfel gradually becomes more opaque from the center outward, until the opacity is 1 within the whole surfel range. During optimization, the outer ring of the translucent surfel is semi-transparent with Gaussian distributed opacity, which allows the color-based gradients to be backpropagated to the surfel geometry parameters. Then semi-transparent regions gradually shrink along with the increased opacity modulating parameter, and the surfel geometry is gradually optimized and stabilized until the translucent surfel becomes fully opaque.

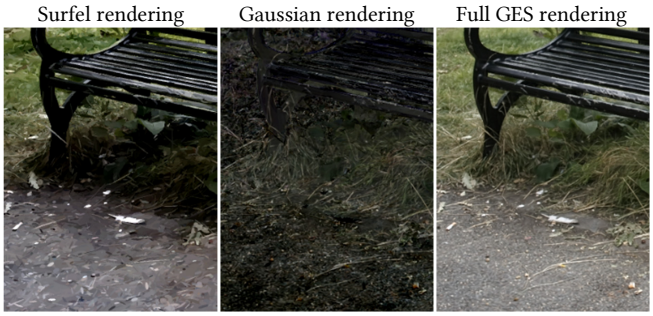

Fig. 2. In our GES representation, 2D surfels represent the coarse-scale geometry and appearance (left), 3D Gaussians supplement fine-scale details (middle), and the full rendering of GES achieves high-quality novel view synthesis by combining both 2D surfels and 3D Gaussians (right).

The basic GES representation can be easily extended to further enhance its capability by incorporating recent improvements of the vanilla 3DGS method. By combining the filtering algorithm for Gaussian splatting [Yu et al. 2024a] with the standard multi sampling anti-aliasing (MSAA) for surfel rasterization, we are able to significantly reduce aliasing artifacts in renderred images (Mip-GES). The Hessian pruning score [Hanson et al. 2024] can be employed to largely reduce the number of Gaussians, further boosting our rendering speed (Speedy-GES). Following [Lee et al. 2024], we can replace the SH coefficients of both surfels and Gaussians by querying colors from a hash grid, and quantize the scaling and rotation of surfels and Gaussians, obtaining a compact storage of GES (Compact-GES). We can also replace the 3D Gaussians with 2D Gaussians [Huang et al. 2024] to reconstruct better scene geometries (2D-GES).

Experiments on various datasets show that our GESs advance the state-of-the-arts as a compelling representation for ultra-fast high-fidelity radiance field rendering. The basic GES rendering achieves 675 fps at 1080p resolutions on average across all tested scenes, exhibiting competitive visual quality without popping artifacts under view changes. The speedy extension of GES, Speedy-GES, boosts the rendering performance to 1135 fps with very little quality loss.

In summary, the main contributions of our work include:

- The introduction of Gaussian-enhanced Surfels (GESs), the first representation that combines the surfel radiance field with Gaussian radiance field to achieve ultra-fast view-consistent rendering with state-of-the-art visual quality.
- A coarse-to-fine optimization method to effectively optimize opaque surfels and Gaussians from multi-view images.
- A series of extensions of the basic GES representation to achieve anti-aliasing in rendering, boosted rendering speeds and compact storage, and reconstruct better scene geometries.

2 RELATED WORK

2.1 Traditional Scene Reconstruction and Rendering

Traditional approaches for novel view synthesis construct light fields from densely sampled images [Buehler et al. 2001; Davis et al. 2012; Gortler et al. 1996; Levoy and Hanrahan 1996]. With the development of Structure-from-Motion (SfM) [Snavely et al. 2006] and multi-view stereo (MVS) [Goesele et al. 2007], full 3D reconstruction is enabled from a collection of photos, based on which different view synthesis methods are proposed [Chaurasia et al. 2013; Hedman et al. 2018a; Kopanas et al. 2021]. Among these prior works, the surface light field [Chen et al. 2018; Wood et al. 2000] is the most relevant representation to ours, which represents the radiance of rays originating from any points on the surface in any directions. The construction of surface light fields requires high-resolution geometry and dense sets of images. Differently, our GES representation uses surfels with view-dependent colors to approximate the surface light field and employs Gaussians to supplement the scene appearance not captured well by surfels. We do not require high-resolution geometry either.

2.2 Neural Radiance Fields

Different neural rendering approaches have been studied for novel view synthesis [Lombardi et al. 2019, 2021; Riegler and Koltun 2020; Thies et al. 2019; Wizadwongsa et al. 2021; Zhou et al. 2018]. Among the many successful approaches, NeRFs [Mildenhall et al. 2021] in particular have achieved remarkable success and have produced an explosion of follow-up works [Barron et al. 2021, 2022a; Fridovich-Keil et al. 2022; Garbin et al. 2021; Reiser et al. 2021; Sun et al. 2022; Takikawa et al. 2022; Tancik et al. 2022; Turki et al. 2022; Yu et al. 2021]. For example, to accelerate the training and rendering of NeRFs, InstantNGP [Müller et al. 2022] replaces the deep MLP by a shallow MLP with the multiresolution hash encoding as input, and can be trained in a few minutes. Adaptive Shells [Wang et al. 2023] extract two meshes from SDFs as scene boundaries. During rendering, it only samples points between ray-mesh intersections to significantly reduce the number of volumerric samples. Quadrature Fields [Sharma et al. 2024] extract meshes from a quadrature field built with the help of NeRFs and bake view-dependent colors obtained from NeRFs as compressed spherical Gaussians stored in textures, after which ray tracing is performed to get all ray-mesh intersections as samples for volume rendering.

2.3 Point-based Rendering

Point-based rendering has been explored for decades for rendering disconnected and unstructured point clouds. To achieve high rendering quality, a common solution is to render points as larger primitives such as circular or elliptic discs, ellipsoids, or surfels [Botsch et al. 2005; Pfister et al. 2000; Ren et al. 2002; Zwicker et al. 2001, 2004]. With the development of differentiable point-based rendering [Wiles et al. 2020; Yifan et al. 2019], point based rendering techniques are also adopted for novel view synthesis [Aliev et al. 2020; Kopanas et al. 2022, 2021; Lassner and Zollhöfer 2021; Rückert et al. 2022]. The most recent work [Zhang et al. 2022] in this category presents point-based radiance fields, where each point is associated with the properties of position and SH coefficients, and has no other geometry properties (e.g., size, rotation or normal). The points are rendered with point splatting, where an image-space Gaussian radial basis function is employed to compute the alpha value of each point on each pixel. The points are then sorted by z-distance and the image is rendered via alpha blending from front to back. The method is designed for foreground scene components and requires masks for initialization, and it is unclear how it can scale to general scenes. Point-NeRF [2022] also uses points to represent radiance fields, but it still requires MLPs and volumetric ray-marching, and hence cannot achieve real-time rendering.

2.4 3D Gaussian Splatting

3DGS [Kerbl et al. 2023] represents radiance fields as sparsely distributed 3D Gaussians with view-dependent colors. Due to the exceptionally fast speed and high quality in view synthesis, it has inspired a great amount of follow-up research [Fan et al. 2024; Hahlbohm et al. 2024; Kerbl et al. 2024; Lyu et al. 2024; Moenne-Loccoz et al. 2024a; Papantonakis et al. 2024; Wu et al. 2024a; Yu et al. 2024a,c]. [Chen and Wang 2024] and [Wu et al. 2024b] provide a comprehensive coverage of recent advancements in this field.

Among these works, 2DGS [Huang et al. 2024] and Gaussian surfels [Dai et al. 2024] adopt 2D Gaussians instead of 3D Gaussians to represent radiance fields. By introducing a perspective-accurate 2D splatting process and incorporating depth distortion and normal consistency terms, 2DGS can optimize 2D Gaussians to be distributed more closely around the surface, hence producing more geometrically accurate radiance fields. SolidGS [Shen et al. 2024] modifies the 2D Gaussian kernel function to create a larger opaque region for improved view consistency. Geometry Field Splatting [Jiang et al. 2024] uses 2D Gaussians to represent a geometry field and converts it into a density field for volumetric rendering. Nevertheless, all of these methods are still volumetric representations of radiance fields – its rendering requires sorting and α-blending of all Gaussians along the ray for each pixel as in 3DGS.

It is known that 3DGS suffers from the popping artifacts during camera movements, because it approximates the accurate per-pixel depth sorting with a tile-based global sorting of Gaussian center depths. StopThePop [Radl et al. 2024] proposes a hierarchical sorting strategy to alleviate the popping artifacts Hahlbohm et al. [2024] use hybrid transparency [Wyman 2016] to approximate the accurate blending per-pixel. Both methods cannot guarantee the per-pixel ordering is fully correct, and thus could produce popping artifacts in novel view synthesis (see Fig. 12 and the supplementary video). Hou et al. [2024a] present a sorting-free method for 3DGS by approximating alpha blending with weighted sums, thereby eliminating the popping artifacts. However, because for each pixel it calculates the weighted sum of all Gaussians along the ray, this method will produce color leakage artifacts of occluded objects.

Many methods have been proposed to accelerate 3DGS rendering and reduce storage by pruning redundant Gaussians [Fan et al. 2024; Hanson et al. 2024; Lee et al. 2024; Niemeyer et al. 2024]. While very fast rendering can be achieved, the excessive pruning of Gaussians tends to lose appearance details. More importantly, as the remaining Gaussians contribute more significantly to the color, the popping artifact is exacerbated. Other methods improve the rasterization

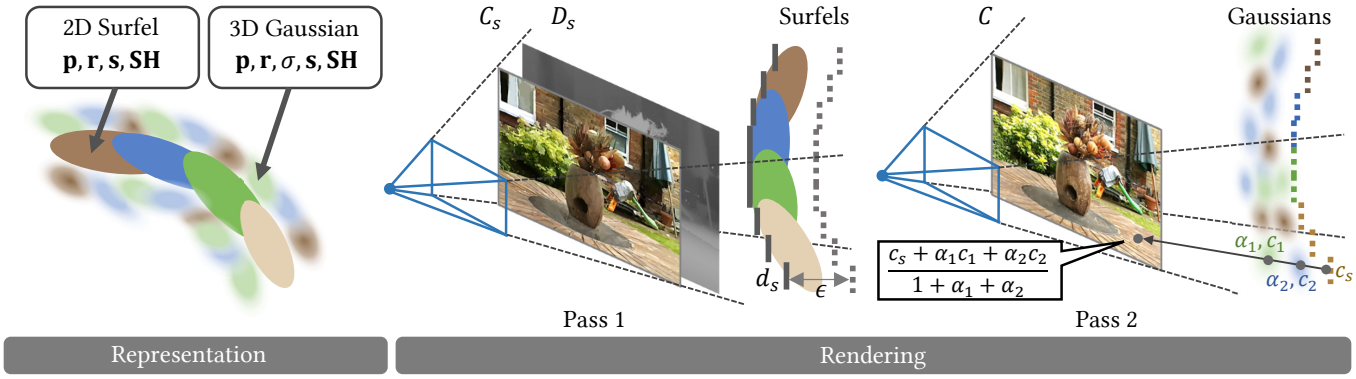

Fig. 3. The representation and rendering pipeline of Gaussian-enhanced Surfels (GESs). The GES representation is composed of a set of 2D opaque surfels $\mathcal{S} = \{\mathbf{p}_i, \mathbf{r}_i, \mathbf{s}_i, \mathbf{SH}_i\}_{i=1}^{N}$, and a few 3D Gaussians $\mathcal{G} = \{\mathbf{p}_i, \sigma_i, \mathbf{r}_i, \mathbf{s}_i, \mathbf{SH}_i\}_{i=1}^{M}$ surrounding the surfels, where $\mathbf{p}_i, \sigma_i, \mathbf{r}_i, \mathbf{s}_i, \mathbf{SH}_i$ are the position, maximum opacity, rotation, scaling and spherical harmonic coefficients, respectively. The rendering of GESs consists of two passes. A surfel rendering pass is first performed in a standard graphics pipeline to produce a color map C_S and a depth map D_S. In the second pass, 3D Gaussians are splatted to the screen, and the colors and weights are accumulated with depth testing based on the $\epsilon-$modified surfel-rendered depth map. The final image C is computed as the linear combination of surfel-rendered color map and Gaussian-rendered image with normalized weights. The rendering pipeline is entirely sorting-free.

pipeline [Feng et al. 2024; Wang et al. 2024], such as reducing the number of covered tiles of screen space Gaussians. However, as sorting still exists as a computational bottleneck, their rendering speed is slower than our method.

RTG-SLAM [Peng et al. 2024] presents a real-time 3D reconstruction system with an RGBD camera for large-scale environments using 3D Gaussian splatting. It uses opaque Gaussians to fit the geometry and dominant colors and nearly-transparent Gaussians to fit residual colors. The opacity value of the opaque Gaussians is 0.99 in the center and decays outwards with the Gaussian function, while the opacity value of our opaque surfels is 1 within the whole surfel range. The rendering of RTG-SLAM is the same as 3DGS.

Different from 3DGS and its follow-ups that all use Gaussians to construct volumetric radiance fields, our method uses 2D opaque surfels with view-depedent colors to construct surfel radiance fields to represent the coarse-level geometry and appearance, and employs Gaussians to construct volumetric radiance fields to enhance surfel radiance fields with fine-scale details. Our rendering is entirely sorting-free, which not only achieves very fast rates, but also produces view-consistent images.

Our surfel optimization is performed by utilizing the opacity modulating parameter to gradually evolve the translucent surfels into opaque ones. This idea is related to differentiable mesh rasterizer methods. They typically leverage the subpixel antialiasing operation [Laine et al. 2020] or convert the mesh into a collection of 3D Gaussians [Rhodin et al. 2015] or build smooth probability maps of each triangle [Liu et al. 2019] to get the gradients of image loss related to vertex positions. Our design is different, which gradually optimizes translucent surfels into opaque ones using the opacity modulating parameter.

3 GAUSSIAN-ENHANCED SURFELS

Gaussian-enhanced Surfels are a bi-scale representation of radiance fields, with a set of 2D opaque surfels with view-dependent colors approximating a surface radiance field to represent the coarse-scale geometry and appearance, and a few 3D Gaussians surrounding the surfels forming a volumetric radiance field to supplement fine-scale appearance details not represented well by the surfel radiance field.

The 2D surfel $\mathcal{S}$ is defined as a 2D unit circular disc on the XY-plane in its local coordinate, associated with a set of properties as $\mathcal{S} = \{\mathbf{p}_i, \mathbf{r}_i, \mathbf{s}_i, \mathbf{SH}_i\}_{i=1}^{N}$, where $\mathbf{p}_i \in \mathbb{R}^3$ is the surfel center position, $\mathbf{r}_i \in \mathbb{R}^4$ is the rotation quaternion, $\mathbf{s}_i \in \mathbb{R}^2$ is the anisotropic scaling, and $\mathbf{SH}_i$ is the spherical harmonics coefficients representing view-dependent colors. This 2D surfel forumla is similar to the 2D Gaussian of 2DGS [Huang et al. 2024], except that our surfels are fully opaque. The surfel is transformed to the world space by sequentially applying the scaling, rotation and translation with $\mathbf{s}_i, \mathbf{r}_i, \mathbf{p}_i$. The color of the whole surfel is the same as the SH color in the direction from the surfel center to the camera position as

$$\mathbf{c}_i = Y(||\mathbf{o} - \mathbf{p}_i||, \mathbf{SH}_i), \tag{1}$$

where $Y(\cdot)$ indicates spherical harmonic function, $||\cdot||$ indicates the norm vector and $\mathbf{o}$ is the camera position. The 3D Gaussians $\mathcal{G}$ are defined as $\mathcal{G} = \{\mathbf{p}_i, \sigma_i, \mathbf{r}_i, \mathbf{s}_i, \mathbf{SH}_i\}_{i=1}^{M}$, with the properties of center position $\mathbf{p}_i$, maximum opacity σ_i, scaling $\mathbf{s}_i$, rotation $\mathbf{r}_i$ and SH coefficients $\mathbf{SH}_i$.

The rendering with GESs consists of two passes. In the first pass, the surfels are rasterized through a standard graphics pipeline. We compute the depths of surfel fragments, perform the depth test with a z-buffer, and write the colors of fragments passing the depth test. The surfel rendered color map C_s and depth map D_s are available after rendering. In the second pass, we splat the Gaussians to the screen, and accumulate the colors and weights of Gaussians for each pixel. During accumulation, we also perform depth testing on the Gaussians using the surfel rendered depth map D_s, and ignore the Gaussians occluded by the surface whose center depths fail to pass the depth testing. The accumulated Gaussian color and weight for a

pixel $\hat{\mathbf{x}}$ are defined as

$$\mathbf{C}_G(\hat{\mathbf{x}}) = \sum_{i=1}^{K} [\mathbb{1}(d_i < d_s(\hat{\mathbf{x}}) + \epsilon)] \mathbf{c}_i \alpha_i(\hat{\mathbf{x}}), \tag{2}$$

$$W_G(\hat{\mathbf{x}}) = \sum_{i=1}^{K} [\mathbb{1}(d_i < d_s(\hat{\mathbf{x}}) + \epsilon)] \alpha_i(\hat{\mathbf{x}}), \tag{3}$$

$$\alpha_i(\hat{\mathbf{x}}) = \sigma_i \exp(-\frac{(\hat{\mathbf{x}} - \hat{\mathbf{p}}_i)^T \Sigma^{-1} (\hat{\mathbf{x}} - \hat{\mathbf{p}}_i)}{2}), \tag{4}$$

where $\mathbb{1}(\cdot)$ is the indicator function, d_i is the Gaussian center depth, d_s is the pixel depth obtained from the depth map D_s, Σ is the projected covariance matrix determined by the rotation and scaling, ϵ is a small positive value introduced to prevent the 3D Gaussians distributed close to the surface from being wrongly truncated. $\hat{\mathbf{p}}_i$ is the projected Gaussian center, and $\mathbf{c}_i$ is the Gaussian color in the current view. α_i is clamped to zero if $\alpha_i(\hat{\mathbf{x}}) < 1/255$, as in 3DGS [Kerbl et al. 2023].

The final image C is computed as the weighted combination of surfel-rendered color map and Gaussian-rendered image

$$C = \frac{C_s W_s + C_G}{W_s + W_G}, \tag{5}$$

where $W_s = 1$ is the fixed weight for surfel color. Since the color of 3D Gaussians is accumulated and normalized by weight, the whole rendering process of GESs is entirely sorting-free. Consequently, GESs not only bypass the computation bottleneck of Gaussian sorting to achieve ultra-fast frame rates, but also produce view-consistent images, successfully avoiding popping artifacts under view changes.

4 COARSE-TO-FINE OPTIMIZATION

Given a set of multi-view images of a static scene, whose corresponding cameras are calibrated by SfM [Schönberger and Frahm 2016], our goal is to optimize a set of 2D surfels and 3D Gaussians faithfully representing the scene for free-view synthesis. The optimization is performed using a coarse-to-fine method. The coarse stage optimizes the opaque surfels to reconstruct the coarse-scale geometry and appearance. In the fine stage, the surfel geometry properties (i.e., $\mathbf{p}_i, \mathbf{r}_i$ and $\mathbf{s}_i$) are fixed, and the 3D Gaussians are added and are jointly optimized together with the surfel SH coefficients to reconstruct rich appearance details.

4.1 Surfel Optimization

The surfels are initialized with the sparse SfM points, where the surfel positions and colors are set as the SfM point positions and colors, the surfel scaling is set as the distance to its nearest neighbor, and the rotation is randomly generated. Directly optimizing opaque surfels from image color loss is very challenging, because the forward process from the surfel geometry parameters to pixel colors is non-differentiable. That is, the change of surfel geometry yields either no color change or a sudden color change (e.g., 0 to $\mathbf{c}_i$), as the color is the same everywhere on the surfel.

To tackle this challenge, we introduce an opacity modulating parameter w_i to gradually evolve translucent surfels into opaque ones during optimization. For a surfel defined as a circular disc on the XY-plane in its local coordinate, the opacity for a point (x, y) on the surfel now is defined as

$$\alpha_i(x, y) = \min(1, w_i G(x, y)), \tag{6}$$

$$G(x, y) = \exp(-\frac{x^2 + y^2}{2}), \tag{7}$$

where $\alpha_i(x, y)$ is clamped to 0 when $\min(\alpha_i(x, y), G(x, y)) < 1/255$. The rasterization of translucent surfels is the same as in 2DGS [Huang et al. 2024]. We use the opacity modulating parameter $w_i \in [0, 255]$ to establish the connection between the translucent surfel and the opaque surfel. When $w_i < 1$, the surfel is a 2D Gaussian. When w_i increases, the Gaussian gradually becomes more opaque from the center outward. When w_i equals 255, the opacity is 1 within the range where $G(x, y) > 1/255$, and 0 outside this range. This results in a circular disc with a radius of $r = \sqrt{2 \log(255)} \approx 3.3$ where the opacity is 1 uniformly, as illustrated in Fig. 4.

During optimization, the outer ring of the surfel is semi-transparent with Gaussian distributed opacity, which allows color-based gradients to be backpropagated to the surfel geometry parameters. The optimization starts with $w_i = 0.1$, which means each surfel is a 2D Gaussians. In the early iterations, the large semi-transparent regions provide effective gradients for optimizing the position and shape of the surfels. We increase w_i gradually in the later iterations, and the semi-transparent regions shrink accordingly, with the surfel geometry eventually stabilized until the translucent surfel becomes a fully opaque one. To obtain more gradients in semi-transparent regions and achieve anti-aliasing, we use 4× supersampling once the w_i values of all surfels are increased over 30.

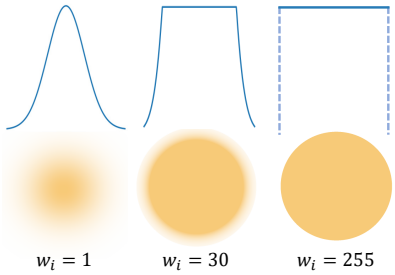

Fig. 4. The shape of $\alpha_i(x, y)$ in Eq. (6) changes as w_i increases.

We use α-blending to render the translucent surfels during optimization, which requires depth sorting of surfels. However, performing accurate depth sorting on each pixel is too expensive, and the tile-based sorting of surfels using the surfel center depths like 3DGS is inaccurate, which yields the final opaque surfels to interleave each other and degrades the reconstruction quality. Observing when w_i is large (i.e., the surfels are mostly opaque), the frontmost surfel (nearest to the camera) covering a pixel contributes the most to the pixel color, we design a more efficient way to fairly approximate α-blending. Specifically, when $w_i < 30$, we perform the tile-based sorting for the surfels in the same way as in 3DGS. Once w_i is increased over 30, after the tile-based sorting, for each pixel we compute the accurate pixel-level depths of the surfels covering it. The surfel with the minimum depth is selected for the first blending computation, while the blending order of the other surfels is not adjusted. This strategy can well approximate the accurate sorting, and reduces both time and memory overhead. More importantly,

the rendering result at the end of optimization is consistent with the z-buffer-based opaque surfel rendering after optimization.

The loss function includes an image loss $\mathcal{L}_1$ and a D-SSIM loss as in [Kerbl et al. 2023]:

$$\mathcal{L}_{rgb} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM}, \tag{8}$$

where $\lambda = 0.2$ in our implementation. In the first 10K iterations, we use the same strategy as in [Yu et al. 2024a] for surfel densification and pruning. At the 10K-th iteration, we discard the surfels with $w_i < 0.8$ (but record the position of these surfels for subsequent 3D Gaussian initialization), increase w_i of the remaining surfels to no smaller than 30, keep w_i from optimization, and disable the densification and pruning. In the 18K-th and 19K-th iteration, we increase w_i of each surfel to no smaller than 60 and 90, respectively. In the 20K-th iteration, we set $w_i = 255$ for all surfels and terminate surfel geometry optimization.

Since we only need the coarse-scale color and depth map from surfel rendering, $\mathcal{S}$ does not need to provide excessive appearance details. After the 15K-th iteration, we disable densification and use a new strategy to further prune surfels. Specifically, for each surfel $\mathcal{S}_i$ in the j-th training view, we compute the number of pixels $n_{i,j}$ where $\mathcal{S}_i$ is the frontmost surfel covering them. The covering score for $\mathcal{S}_i$ is defined as $n_i = \max\{n_{i,j}\}_{j=1}^{T}$, where T is the number of training views. We then prune all surfels with $n_i < n_{thr}$ to remove tiny or invisible surfels. We set $n_{thr} = 16$ for real scenes and $n_{thr} = 4$ for synthetic scenes.

4.2 Gaussian-Surfel Joint Optimization

After surfel optimization, we add 3D Gaussians and jointly optimize them with the SH coefficients of surfels. The loss function is still $\mathcal{L}_{rgb}$ in this stage.

The Gaussians are initialized with the positions of the surfels with $w_i < 0.8$ disgarded at the 10K-th iteration during surfel optimization. The other Gaussian properties are initialized the same way as in [Kerbl et al. 2023]. We periodically densify and prune 3D Gaussians during joint optimization. Every 1000 iterations, we compute the squared error map between the ground truth and the rendered image across all training views. The error in each pixel is normalized in the error map, and we treat the normalized error as the sampling probability for the pixel. We then sample a fixed number of pixels based on the probabilities for each image. Using the corresponding surfel depth map and camera parameters, we obtain a point cloud, which serves as the initial positions for the newly added 3D Gaussians. The other Gaussian properties are also initialized the same way as in [Kerbl et al. 2023]. For Gaussian pruning, similar to RadSplat [Niemeyer et al. 2024], we compute the contribution score for each 3D Gaussian $\mathcal{G}_i$ in the j-th training view using

$$s_{i,j} = \max_{\hat{\mathbf{x}}} \frac{c_{i,j}\alpha_{i,j}(\hat{\mathbf{x}})}{W_s + W_G}, \tag{9}$$

where $\hat{\mathbf{x}}$ is the pixel the Gaussian covers and $\alpha_{i,j}(\hat{\mathbf{x}})$ is the opacity defined in Eq. (4) in the j-th training view. $c_{i,j} = \max\{c_{i,j}^k\}_{k=1}^{3}$ and $c_{i,j}^k$ is the k-th channel of the Gaussian color $\mathbf{c}_{i,j}$. 3D Gaussians with $\max\{s_{i,j}\}_{j=1}^{T} < 0.02$ among all the T training views are pruned to remove invisible Gaussians or Gaussians with low contributions.

5 IMPLEMENTATION DETAILS

For the optimization of GESs, we implement our method upon the Pytorch framework of 3DGS/2DGS, based on per-pixel rendering (i.e., launching a GPU thread per pixel for querying its surfels and Gaussians). For the GES rendering after optimization, we develop an equivalent per-primitive renderer (i.e., launching a GPU thread per fragment of primitives for querying its covered pixels) using OpenGL. The 4× supersampling used in surfel rendering, which is implemented by rendering a larger resolution image and downsampling it to the target resolution, is replaced by 4× multi-sampling anti-aliasing (4×MSAA) in the OpenGL implementation. Since an opaque surfel only has a single color, the two implementations are equivalent. The OpenGL rendering also consists of two rendering passes. In the first pass, the depth testing and writing are enabled and the blending is disabled. We use a geometry shader to generate surfels from points and render the surfel color map and depth map. Then, the multisampling textures are resolved (note that MSAA is not applied for the subsequent 3D Gaussian rendering) and the depth map is modified according to offset ϵ using a post-processing shader. In the second pass, depth testing and additive blending is enabled and depth writing is disabled. We also use a geometry shader to generate 3D Gaussians from points, and splat 3D Gaussians to obtain C_G and W_G. Then, we compute the final image using Eq. (5) with a post-processing shader. Our method is entirely based on the programmable graphics pipeline and does not rely on compute shaders, making it easy to be integrated into existing rendering engines and mobile devices.

The depth offset ϵ in Eq. (2) and Eq. (3) has a significant impact on rendering quality. If it is too large, some 3D Gaussians that should be occluded by the surface may leak out; if it is too small, many 3D Gaussians close to the surface will be wrongly truncated. We set $\epsilon_i = \frac{5}{D}\sum_{j=1}^{D} s_{i,j}$ for each Gaussian, where D is the dimension of scaling and $s_{i,j}$ is the j-th axis length of scaling $\mathbf{s}_i$, which allows ϵ_i to vary adaptively to the geometric granularity, achieving high-quality results while reducing leakage.

6 EXTENSIONS

The basic GES representation can be easily extended by incorporating recent improvements of the vanilla 3DGS method, to achieve anti-aliasing in rendering (Mip-GES), boosted rendering speeds (Speedy-GES) and compact storage (Compact-GES), and reconstruct better scene geometries (2D-GES).

6.1 Mip-GES

To reduce aliasing artifacts, we utilize 4×MSAA in surfel rendering, and employ a screen-space fixed-size EWA filter when rasterizing 3D Gaussians. However, the fixed-size EWA filter shows dilation or high-frequency artifacts when rendering at varying scales. Therefore, we apply the world space filter and the approximated screen space box filter proposed by MipSplat [2024b] to our 3D Gaussians, which can significantly reduce aliasing artifacts in renderred images.

6.2 Speedy-GES

Our surfel pruning strategy effectively removes more than 80% of invisible or low-coverage surfels, providing reliable and compact

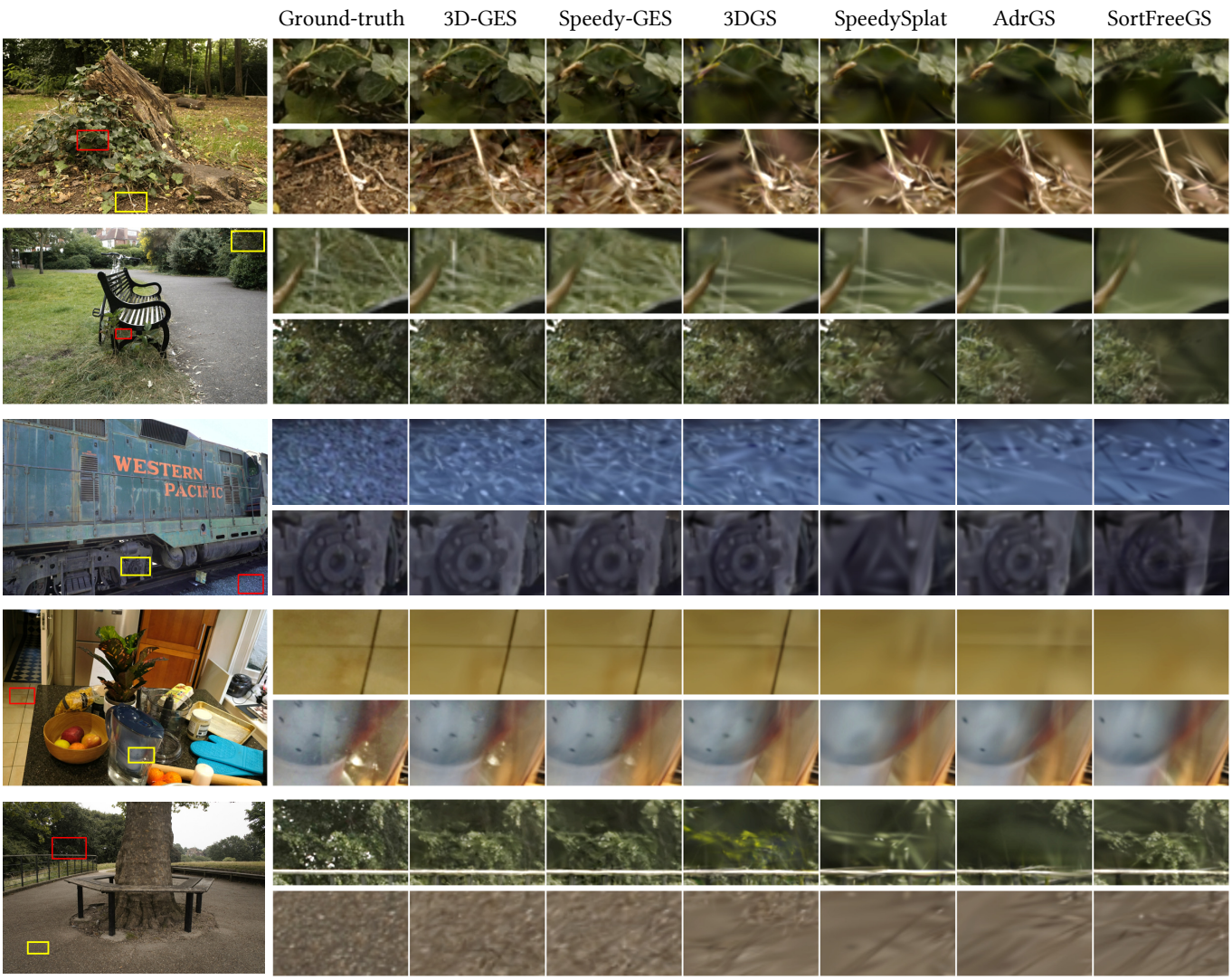

Fig. 5. Qualitative comparisons on image quality. From top to bottom: *Stump*, *Bicycle*, *Train*, *Counter* and *Treehill*. Our GES representations reconstruct rich texture details while maintaining high frame rates with sorting-free rendering.

geometry representation. Although our Gaussian pruning strategy also removes low-contribution Gaussians, a considerable amount of redundant Gaussians can be further pruned at the cost of a slight quality loss. In Speedy-GES, we replace our contribution score used in Gaussian pruning by the Hessian pruning score proposed in SpeedySplat [2024], resulting in 1.6× acceleration in rendering.

6.3 Compact-GES

We use the storage compression method in C3DGS [Lee et al. 2024] to further compact the basic GES. Specifically, we replace the SH coefficients of both surfels and Gaussians by querying colors from the hash grid and use residual vector quantization to quantize the scaling and rotation parameters of both surfels and Gaussians, achieving over 20 times storage compression with only a little quality loss.

6.4 2D-GES

As our opaque surfels are 2D planar discs, the normals and depths on the intersection regions of surfels can be discontinuous, which motivates us to use Gaussians to smooth the geometry of surfels. Given the surfel depth map D_s and normal map N_s, we compute the smoothed depth map D_{smooth} and normal map N_{smooth} as

$$D_G(\hat{\mathbf{x}}) = \sum_{i=1}^{N} [\mathbb{1}(d_i < d_s(\hat{\mathbf{x}}) + \epsilon)] d_i \alpha_i(\hat{\mathbf{x}}), \tag{10}$$

$$N_G(\hat{\mathbf{x}}) = \sum_{i=1}^{N} [\mathbb{1}(d_i < d_s(\hat{\mathbf{x}}) + \epsilon)] \mathbf{n}_i \alpha_i(\hat{\mathbf{x}}), \tag{11}$$

$$D_{smooth} = \frac{D_s W_s + D_G}{W_s + W_G}, \; N_{smooth} = \frac{N_s W_s + N_G}{W_s + W_G}, \tag{12}$$

Table 1. Dataset-averaged quantitative evaluation on image quality. Results marked with * are evaluated using our reproduced code.

Datasets	Mip-NeRF360			Deep Blending			Tanks & Temples			NeRF Synthetic		
Method|Metric	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓
MipNeRF	0.792	27.58	0.237	0.15	29.40	0.245	0.759	22.22	0.257	0.951	33.09	0.060
Plenoxel	0.625	23.08	0.462	0.795	23.06	0.510	0.719	21.08	0.379	0.958	31.76	0.049
INGP	0.699	25.59	0.331	0.817	24.97	0.390	0.745	21.92	0.305	0.959	33.18	0.055
3DGS	0.814	27.43	0.214	0.903	29.41	0.243	0.841	23.62	0.183	0.969	33.31	0.037
SortFreeGS	0.804	27.19	0.211	0.902	29.63	0.229	0.842	23.61	0.178	N/A	N/A	N/A
SortFreeGS*	0.790	27.04	0.263	0.910	30.22	0.254	0.825	23.49	0.223	0.964	32.72	0.040
StopThePop	0.816	27.44	0.217	0.904	29.69	0.248	0.843	23.43	0.178	0.970	33.32	0.035
AdrGS	0.792	26.95	0.259	0.906	29.88	0.254	0.836	23.51	0.203	0.968	33.24	0.036
MipSplat	0.815	27.49	0.214	0.904	29.48	0.243	0.846	23.71	0.158	0.969	33.33	0.037
AbsGS	0.821	27.49	0.208	0.902	29.65	0.243	0.851	23.75	0.167	0.969	33.32	0.037
3D-GES	0.813	27.38	0.208	0.906	30.00	0.241	0.841	23.95	0.181	0.967	33.37	0.033
Mip-GES	0.812	27.42	0.208	0.906	30.06	0.239	0.847	23.97	0.177	0.969	33.37	0.032
SpeedySplat	0.782	26.92	0.296	0.887	29.32	0.311	0.818	23.39	0.241	0.955	32.50	0.056
Speedy-GES	0.806	27.07	0.226	0.908	30.03	0.247	0.829	23.73	0.208	0.962	32.60	0.043
C3DGS	0.800	27.03	0.243	0.906	29.80	0.258	0.835	23.40	0.201	0.968	33.24	0.034
Compact-GES	0.808	26.98	0.221	0.906	29.93	0.251	0.832	23.62	0.194	0.964	33.12	0.033
2DGS	0.798	26.82	0.253	0.905	29.67	0.260	0.833	23.17	0.213	0.966	32.82	0.037
2D-GES	0.808	26.76	0.219	0.911	30.02	0.235	0.834	22.76	0.190	0.967	33.24	0.034

where d_i and n_i are the depth and normal of the Gaussian $\mathcal{G}_i$ covering the corresponding pixel $\hat{\mathbf{x}}$. We follow previous works [Guédon and Lepetit 2023; Ye et al. 2024a] to use the central depth and minimal axis direction of a 3D Gaussian as d_i and n_i, respectively, which can effectively improve the geometry quality. However, as mentioned in 2DGS [Huang et al. 2024], simply using the central depth can not ensure the geometry consistency in different views. Therefore, we further propose 2D-GES, which replaces 3D Gaussians with 2D Gaussians, to further improve the geometry reconstruction. The d_i and n_i are replaced by the planar depth and normal of the 2D Gaussian in 2D-GES. We also design a specific optimization algorithm for 2D-GES with geometry regularization terms. For further details, please refer to the supplementary material.

7 RESULTS AND EVALUATION

We conduct extensive experiments and comparisons on a workstation with an i7-13700KF CPU, 32GB memory and an NVIDIA RTX 4090 GPU, to demonstrate the effectiveness and efficiency of GESs. We also perform ablation studies to validate our representation and optimization designs.

Datasets. Our evaluation uses the same datasets as in [Kerbl et al. 2023], which includes eight synthetic scenes from the NeRF Synthetic Dataset [Mildenhall et al. 2020], nine real scenes from the Mip-NeRF360 Dataset [Barron et al. 2022b], two real scenes from the DeepBlending Dataset [Hedman et al. 2018b] and two real scenes from the Tanks & Temples Dataset [Knapitsch et al. 2017]. These scenes include synthetic objects with complex geometries as well as various texture-rich indoor and outdoor environments, suitable for a comprehensive evaluation of different methods. Following 3DGS, we use the pre-downscaled images for the training and evaluation on real scenes. We also use the DTU Dataset [Jensen et al. 2014], which comprises 15 real scenes with corresponding masks and ground truth point clouds, to evaluate the geometry reconstruction quality of 2D-GES.

Baselines and metrics. We compare our basic GES representation (named **3D-GES**) and its extended versions (**Mip-GES**, **Speedy-GES**, **Compact-GES**, **2D-GES**) against the following state-of-the-art baselines: **3DGS** [Kerbl et al. 2023], the vanilla 3D Gaussian Splatting; **2DGS** [Huang et al. 2024], a method using 2D Gaussians as primitives to improve geometry reconstruction; **SortFreeGS** [Hou et al. 2024b], a method replacing the alpha blending of Gaussians with the weighted sum of Gaussians; **AdrGS** [Wang et al. 2024], a method using more precise bounding boxes for splatted Gaussians to accelerate rendering; **SpeedySplat** [Hanson et al. 2024], a method applying a new pruning score to prune about 90% of Gaussians to accelerate rendering; **MipSplat** [Yu et al. 2024a], a method using a 3D world space filter and a 2D screen space filter to achieve alias-free rendering; **AbsGS** [Ye et al. 2024b], a method using homodirectional view-space positional gradient as the criterion for densification; **C3DGS** [Lee et al. 2024], a method focusing on reducing the storage overhead; **StopThePop** [Radl et al. 2024], a method using a hierarchical sorting strategy to alleviate popping artifacts; as well as three NeRF-related methods: **MipNeRF** [Barron et al. 2022b], **INGP** [Müller et al. 2022] and **Plenoxel** [Fridovich-Keil et al. 2022]. These baseline methods are the state-of-the-arts in the directions of rendering quality, rendering speed, storage overhead, geometry

Ground-truth	Speedy-GES	SpeedySplat	SpeedySplat	SpeedySplat
FPS	1286	1338 (90%)	960 (60%)	558 (30%)
PSNR↑ / LPIPS↓	22.59 / 0.316	22.51 / 0.463	22.56 / 0.372	22.60 / 0.362

Fig. 6. Comparisons between Speedy-GES and SpeedySplat [Hanson et al. 2024] in *Treehill*. We adjust the proportion (90%, 60% and 30%) of Gaussians pruned in SpeedySplat to make its rendering quality close to Speedy-GES. When pruning 30% of Gaussians, SpeedySplat's quality is still not so good as Speedy-GES, and its frame rate drops to less than half of ours.

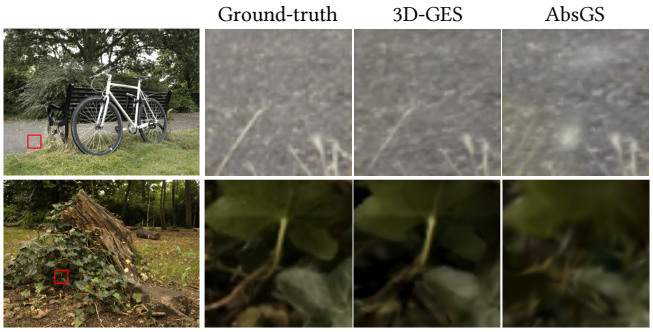

Fig. 7. Qualitative comparisons between our 3D-GES and AbsGS [Ye et al. 2024b]. AbsGS exhibits floaters and missing details.

reconstruction, and view consistency, allowing for a comprehensive evaluation of our approach.

We use the standard PSNR, LPIPS, and SSIM metrics to evaluate the rendering quality. We use FPS, computed by the reciprocal of the average rendering time, to evaluate the rendering speed. We use the chamfer distance (CD) to evaluate the geometry reconstruction quality. We also follow StopThePop [Radl et al. 2024] to use ꟻLIP_1 and ꟻLIP_7 to evaluate the short-term and long-term popping artifacts on the camera paths we generated.

7.1 Comparisons With Baselines

Rendering quality. As shown in Table 1, our 3D-GES and Mip-GES achieve high rendering quality competitive with the state-of-the-art methods. As shown in Fig. 5, 3D-GES reconstructs fine appearance details comparable to or even better than 3DGS. Among the rendering acceleration methods, AdrGS uses the load balancing loss

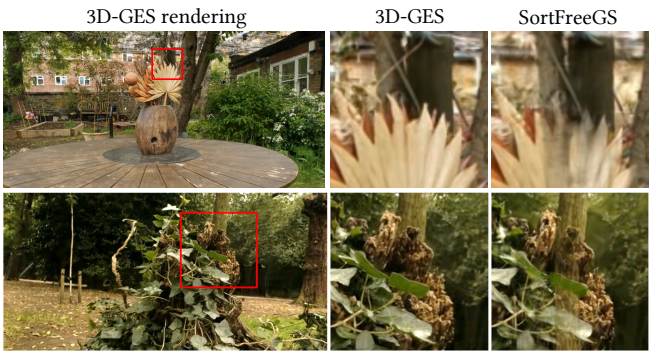

Fig. 8. Qualitative comparisons between our 3D-GES and SortFreeGS [Hou et al. 2024a]. The background color leaks through the foreground objects in SortFreeGS.

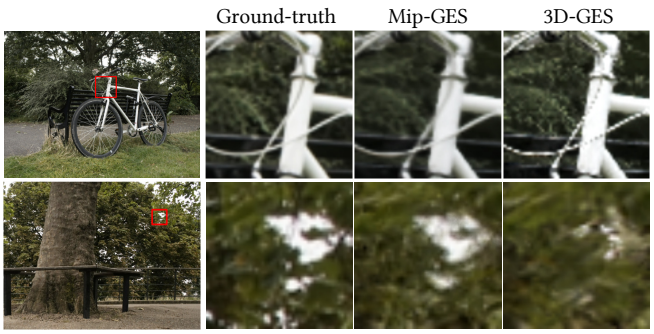

Fig. 9. Qualitative comparisons between 3D-GES and Mip-GES. By applying the world space filter and screen space filter proposed by MipSplat [Yu et al. 2024a] to our 3D Gaussians, our rendering quality is further improved.

to equalize the number of Gaussians covering each pixel. However, it results in smeared or blurry rendering results, as observed in the floor in *Counter* and the grass in *Bicycle*. SpeedySplat achieves significant rendering acceleration by discarding a large number of Gaussians, but this comes at the cost of missing appearance details, leading to poor LPIPS scores. Our Speedy-GES, by avoiding the sorting time, is able to retain more Gaussians to preserve high quality rendering and meanwhile possess ultra-fast rendering speed comparable to SpeedySplat. As shown in Fig. 6, in order to achieve a rendering quality comparable to our Speedy-GES, SpeedySplat needs to prune much fewer Gaussians and its frame rates drops to less than half of ours. In Fig. 7, we compare our method with AbsGS [Ye et al. 2024b]. The densification strategy in AbsGS reduces image blurs, but still misses some details, and the aggressive densification also tends to generate floaters. Compared to the sorting-free method, SortFreeGS, our method leverages surfels to construct the coarse geometry, avoiding color leakage of occluded objects. As shown in Fig. 8, SortFreeGS exhibits noticeable color bleeding when the distance between the camera and objects is different from the training set. It is because SortFreeGS directly uses the Gaussian depths to compute the rendering weights of Gaussians, which cannot guarantee the occlusion correctness from arbitrary views. By

Table 2. Averaged quantitative evaluation of popping artifacts on the MipNeRF360 dataset. STP: StopThePop, Speedy: SpeedySplat, SFGS: SortFreeGS.

	STP	3DGS	Speedy	SFGS	2D-GES	3D-GES
$\text{ꟻLIP}_1 \downarrow$	0.037	0.041	0.043	0.034	0.032	0.032
$\text{ꟻLIP}_7 \downarrow$	0.126	0.128	0.130	0.120	0.114	0.117

Table 3. Dataset-averaged quantitative evaluation on frame rate (FPS) at 1080p and 2160p resolutions, storage (MB) and training time (minute). Results marked with * are evaluated using our reproduced code. The training time is measured for the native image resolutions of training images.

	FPS (1080p)	FPS (2160p)	MEM	Train
3DGS	185	62	734	28
SortFreeGS*	321	168	506	42
StopThePop	167	55	830	40
MipSplat	131	43	1054	37
AdrGS	537	195	274	16
AbsGS	176	60	746	29
SpeedySplat	1140	369	78	14
C3DGS	139	47	49	36
2DGS	90	24	504	26
3D-GES	675	233	366	43
2D-GES	718	247	343	52
Mip-GES	640	213	394	49
Speedy-GES	1135	348	185	36
Compact-GES	300	128	47	39

applying the world space filter and screen space filter to 3D Gaussians, our Mip-GES achieves alias-free rendering, alleviating the dilation artifacts in distant background and high-frequency artifacts in thin structures, as shown in Fig. 9.

View Consistency. Our GESs are sorting-free, fundamentally avoiding the popping artifacts caused by the sorting approximation in 3DGS. To evaluate view consistency, we follow [Hou et al. 2024b] to slightly rotate the camera and show two adjacent frames in Fig. 12. Please also refer to our supplementary video for a more pronounced comparison. We can see that Both 3D-GES and 2D-GES have no popping artifacts, while 3DGS, StopThePop and SpeedySplat all show popping artifacts, especially for SpeedySplat, because it removes a large number of Gaussians, resulting in a higher overall opacity of the remaining Gaussians, which exacerbates popping artifacts. StopThePop cannot guarantee to eliminate the popping artifacts, especially when the view is far from the training views. Table 2 shows the short-term ꟻLIP_1 and long-term ꟻLIP_7 metrics, which measure the consistency between rendered frames and warped frames with optical flow [Radl et al. 2024]. As shown, our sort-free rendering effectively eliminates the popping artifacts and achieves the best results. Please note that as mentioned in 2DGS [Huang et al. 2024], 3DGS utilizes view-dependent intersection planes of 3D Gaussians for splatting, which also results in inconsistency. For the same reason, our 3D-GES with 3D Gaussians cannot fully guarantee the view consistency, while our 2D-GES, using 2D surfels and 2D Gaussians as primitives, achieves complete view consistency. Nevertheless, in our view consistency evaluation, we do not observe obvious view inconsistency artifacts in 3D-GES rendering.

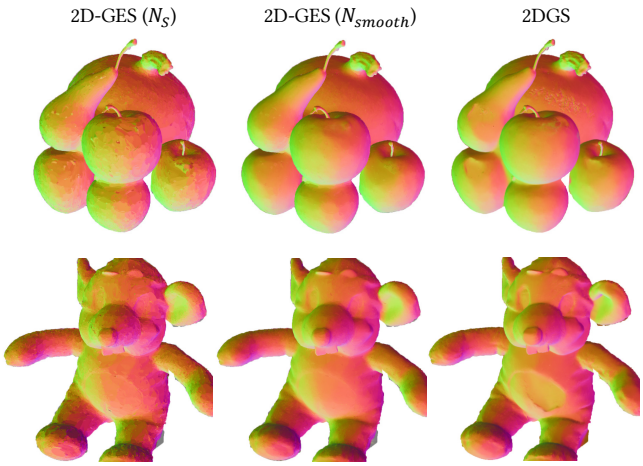

Fig. 10. Qualitative comparisons between the surfel normal map (N_S) and smoothed normal map (N_{smooth}) of 2D-GES and normal map of 2DGS [Huang et al. 2024].

7.2 Ablation Study

In this section, we isolate our algorithmic choices and evaluate their effects on the rendering quality and frame rate. We conduct experiments on Mip-NeRF360 Dataset and report average metrics across all scenes.

Rendering speed. As listed in Table 3, our GESs possess ultra-fast frame rates at 1080p and 2160p resolutions while maintaining the competitive SOTA rendering quality. 3D-GES achieves 675 fps at 1080p resolution and 233 fps at 2160p resolution, and Speedy-GES achieves 1135 fps at 1080p resolution and 348 fps at 2160p resolution. The rendering speed of 3D-GES at 1080p resolution is about 5.2× as fast as MipSplat, 4.0× as fast as StopThePop, 3.6× as fast as 3DGS, 2.1× as fast as SortFreeGS, and 1.3× as fast as AdrGS, whose rendering qualities are comparable. Though SpeedySplat is faster than 3D-GES, it is at the cost of degrading rendering quality. In constrast, our Speedy-GES achieves competitive rendering speed as SpeedySplat, while still maintaining high quality rendering.

Storage. Our method requires only a small number of surfels to construct coarse geometry, while Gaussians are used to further enrich appearance details. In contrast, 3DGS requires Gaussians to simultaneously handle both geometry and color reconstruction, tending to densify much more Gaussians. As a result, the storage consumption of 3D-GES is less than half of that of 3DGS (366MB v.s. 734MB in Table 3). After pruning with the same strategy as in SpeedySplat, the storage overhead is further reduced by about 50%. Finally, by applying the same quantization and hash-grid techniques to compress primitive parameters as in C3DGS, we achieve even lower storage overhead than C3DGS (47MB v.s. 49MB in Table 3) with a comparable rendering quality as C3DGS (see Table 1).

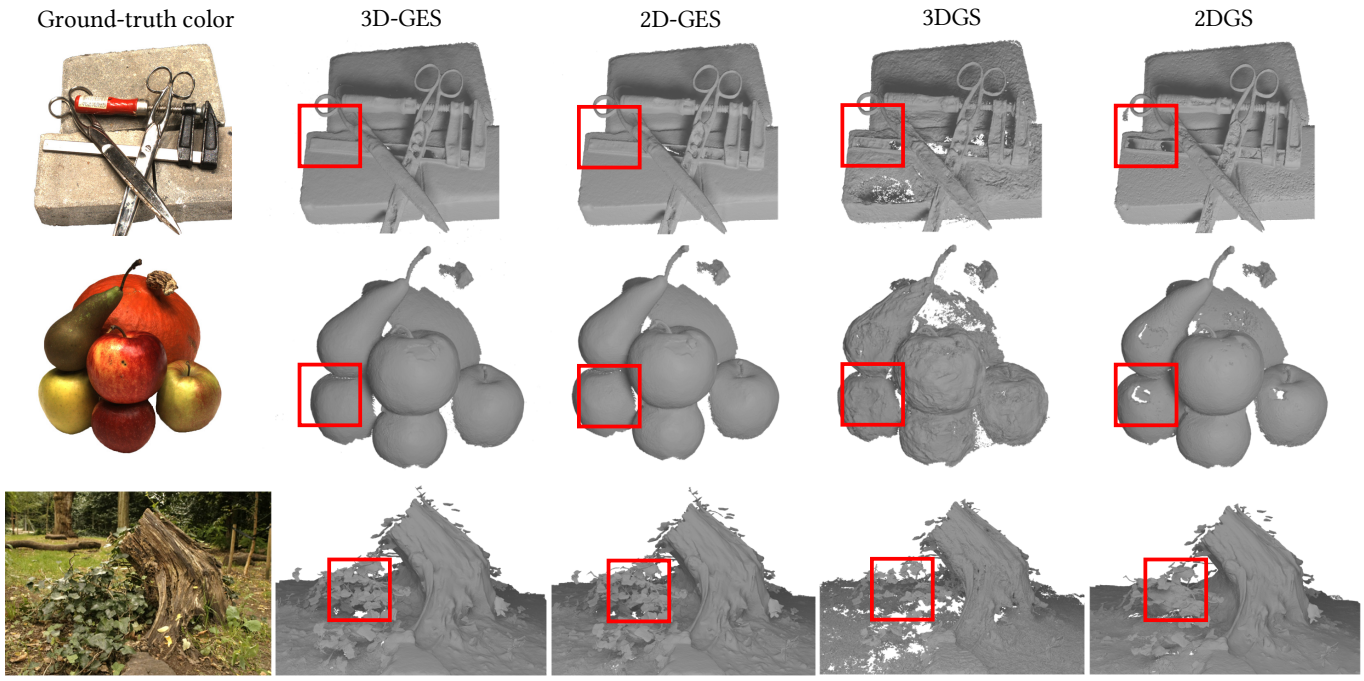

Fig. 11. Qualitative comparisons between our 2D-GES and 2DGS [Huang et al. 2024]. The reconstructed meshes of 2DGS show some holes on glossy surfaces.

Table 4. Quantitative comparisons of geometry quality using the chamfer distance (CD) metric on DTU Datasets [Jensen et al. 2014]. We use the central depth and the shortest axis direction of 3D Gaussians as the depth and normal, and apply the same regularization terms and training process as 2D-GES to our 3D-GES, achieving better reconstruction quality than 3DGS.

Scan	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean
3DGS	2.14	1.53	2.08	1.68	3.49	2.21	1.43	2.07	2.22	1.75	1.79	2.55	1.53	1.52	1.50	1.97
2DGS	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	0.80
3D-GES	0.56	0.89	0.49	0.37	1.15	0.96	0.82	1.35	1.28	0.80	0.67	1.51	0.47	0.87	0.58	0.85
2D-GES	0.52	0.73	0.41	0.38	1.19	0.88	0.78	1.26	1.18	0.69	0.66	1.47	0.44	0.80	0.52	0.79

Geometry reconstruction. In our 2D-GES implementation, 2D Gaussians not only enrich appearance details but also play a role in smoothing the geometry. As shown in Fig. 10, the normal map rendered from surfels exhibits discontinuities, whereas 2D Gaussians effectively smooth the discontinuous areas across surfels. As shown in Table 4, 2D-GES achieves comparable geometry reconstruction quality as 2DGS on the DTU dataset. Moreover, the opaque surfels is geometrically consistent across different views, making 2D-GES capable of well reconstructing glossy surfaces, as shown in Fig. 11, while 2DGS has hole artifacts on glossy areas. It is worth noting that if we follow previous methods [Guédon and Lepetit 2023; Ye et al. 2024a] and use the central depth and the shortest axis direction of the 3D Gaussian as the depth and normal for each point on the Gaussian, the geometry can also be optimized using the same optimization process as 2D-GES. This approach achieves better geometry compared to 3DGS, as shown in Table 4.

Two-stage optimization. To validate the necessity of our two-stage optimization, we experimented with a single-stage optimization scheme. Specifically, we use translucent surfels to render both depth map and color map as in 2DGS [Huang et al. 2024]. The Gaussians then accumulate colors and weights according to the depth map, and the final color is computed using Eq. (5). We gradually increase w_i to make all surfels fully opaque, identical to Sec. 4.1. As demonstrated in the original 3DGS paper, the 3D Gaussian primitives are highly flexible and can fit input images well alone. The single-stage optimization tends to favor 3D Gaussians for image fitting rather than surfels. As shown in Fig. 13, without sufficient surfels covering the scene surface, color leakage often happens in novel views, because our Gaussian primitives are blended order-independently. Quantitatively, as shown in Table 5, the averaged PSNR of the single-stage optimization on Mip-NeRF360 dataset is 27.04, while the two-stage optimization achieves 27.38, which also shows the necessity of our two-stage optimization design.

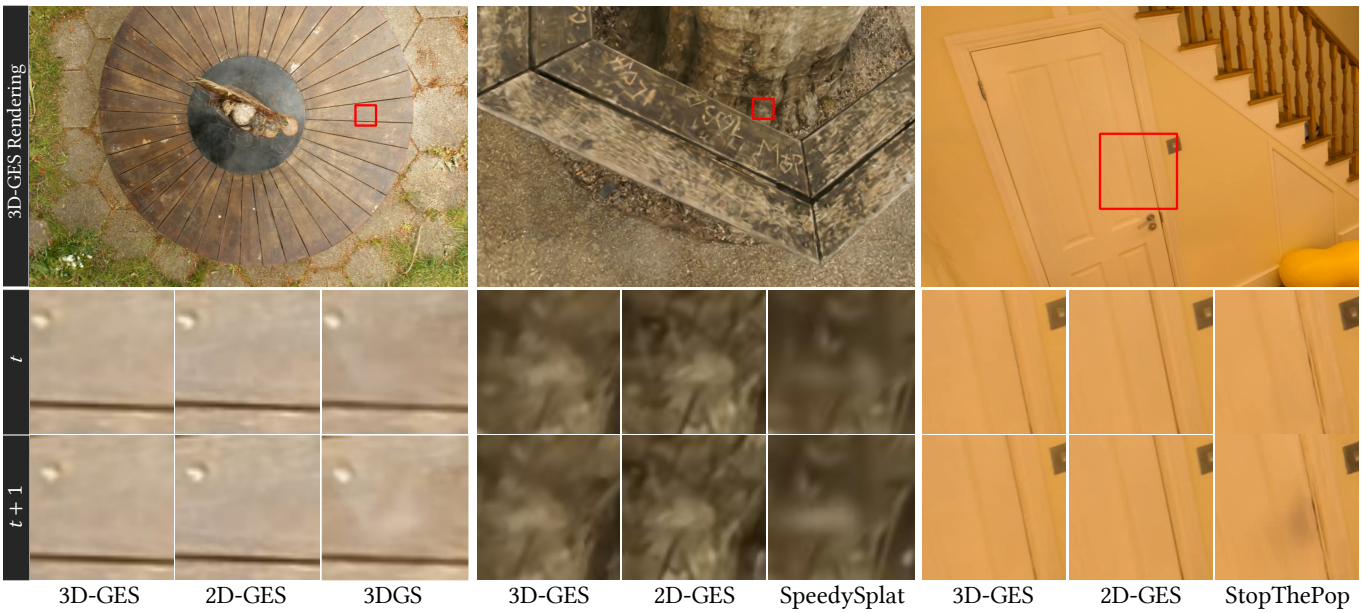

Fig. 12. Qualitative comparisons of "popping" artifacts. Both 2D-GES and 3D-GES achieves view-consistent rendering due to our sort-free rendering.

Fig. 13. Qualitative comparisons between two-stage (3D-GES) and single-stage optimization schemes. Without sufficient surfels covering the surface, color leakage happens in the results of single-stage approach.

Table 5. Ablation study. Dataset-averaged image quality and frame rate (1080p) on Mip-NeRF360 Dataset [Barron et al. 2022a] with isolated algorithm choices.

	SSIM ↑	PSNR ↑	LPIPS ↓	FPS ↑
3D-GES	0.813	27.38	0.208	675
Single stage optimization	0.809	27.04	0.216	624
Only surfels	0.611	24.22	0.501	4771
Only Gaussians	0.774	26.02	0.272	705
w/o order adjustment	0.801	26.97	0.213	663
$\epsilon = 0$	0.806	27.11	0.212	678
$\epsilon = 0.1$	0.811	27.19	0.216	666
RGB surfels	0.808	27.29	0.208	688

Rendering using only surfels or Gaussians. As shown in Fig. 14, rendering using only either of the two primitives significantly degrades the rendering quality. When only surfels are used for rendering ($C = C_S$), color discontinuity occurs across the surfels, and detailed structures are missing. When only Gaussians are used ($C = C_G/W_G$), the background can be reconstructed with high quality, since Gaussians are optimized to fit the images without occlusion conflicts. However, foreground objects exhibit noticeable color leakages similar to SortFreeGS. The quantitative results are listed in Table 5. We also show the surfel rendered image ($C = C_S/(1 + W_G)$) and Gaussian rendered image ($C = C_G/(1 + W_G)$) in our GES rendering pipeline, whose sum is the full rendering result, which demonstrates that opaque surfels model the coarse geometry and appearance, and Gaussians supplement high-frequency details, aligning with the motivation of our representation.

The impact of adjusting sorting order. If the sorting order is not adjusted in each pixel during the surfel optimization (i.e., without selecting the frontmost surfel for the first blending computation), some surfels will protrude and interleave each other when rendered with z-buffer-based depth testing, especially in textureless indoor scenes, as shown in Fig. 15. The rendering quality is degraded under this choice, as shown in Table 5.

The impact of ϵ. We present the quantitative rendering results using different ϵ strategies in Table 5, including setting ϵ to zero and a small constant (0.1 in our experiments), and setting ϵ adaptively in 3D-GES. Compared to the ϵ strategy used in 3D-GES, the fixed ϵ results in a slight loss of rendering quality. As shown in Fig. 16, when ϵ is set to zero, some Gaussians distributed close to the surfel depth are wrongly truncated, causing color discontinuities. Setting ϵ to a fixed small value may lead to color leakage in fine structures. Adjusting ϵ based on the geometry granularity as in our 3D-GES is an appropriate strategy for high quality rendering.

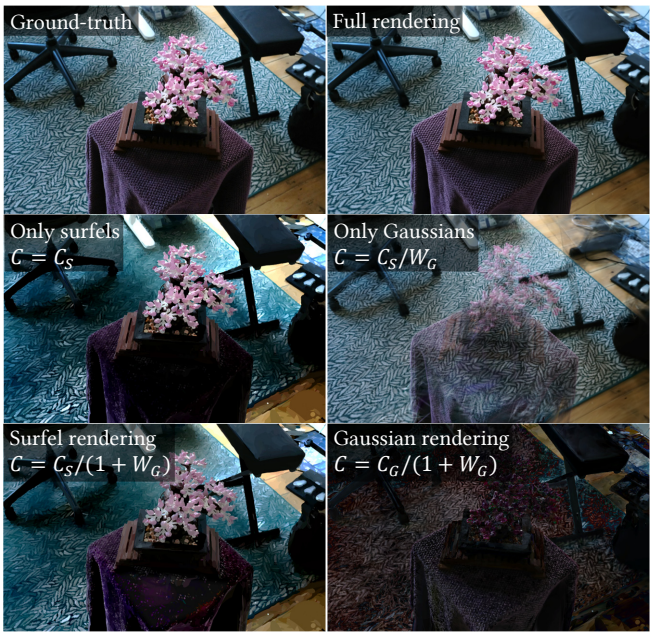

Fig. 14. Qualitative comparisons of images rendered with only surfels or only 3D Gaussians.

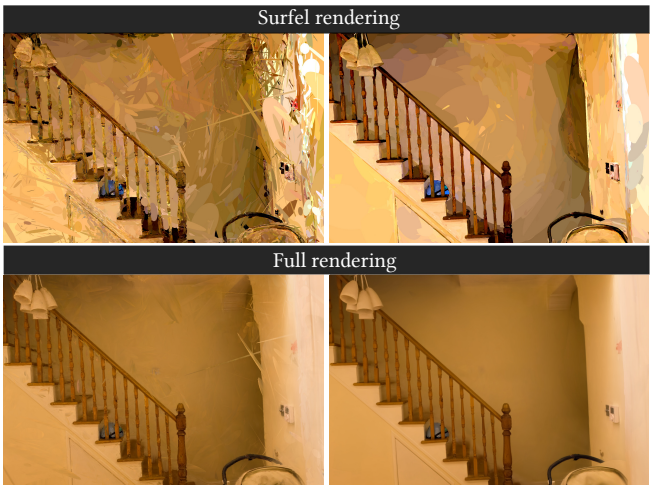

Fig. 15. Qualitative comparisons of images rendered with or without sorting order adjustment during surfel optimization.

Using view-independent RGB colors for surfel rendering. As demonstrated in Table 5 (RGB surfels), we find that replacing spherical harmonics coefficients by view-independent RGB colors in surfel rendering does not cause a noticeable degradation in quality, and it can slightly accelerate rendering and reduce storage overhead.

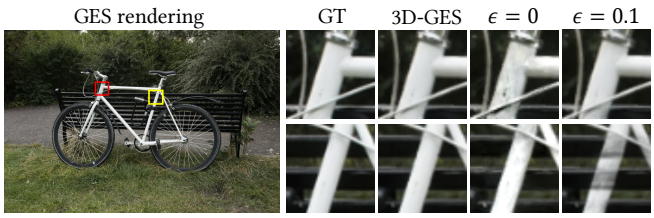

Fig. 16. Qualitative comparisons of different ϵ strategies.

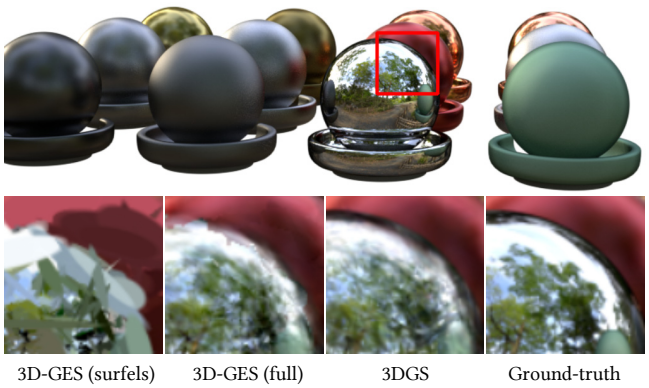

Fig. 17. Qualitative comparisons between GES and 3DGS [Kerbl et al. 2023] on specular objects reconstruction.

8 DISCUSSION AND CONCLUSION

The training time for 3D-GES is approximately 1.3× to 1.6× longer than that of 3DGS, mainly due to the time required for surfel optimization. Additionally, as shown in Fig. 17, for specular surfaces, 3DGS can rely on a large number of low-opacity Gaussians to mimic the reflection effect, but our opaque surfels lack this capability, leading to a decrease in rendering quality. Using ray tracing [Moenne-Loccoz et al. 2024b] or environment maps [Ye et al. 2024a] to capture reflections might help address this issue. Moreover, our methods and 3DGS share the same problem of initialization sensitivity. The reconstruction quality degrades when using randomly initialized points instead of SfM points in real scenes. Incorporating the deterministic state transition of MCMC samples proposed by 3DGS-MCMC [Kheradmand et al. 2024] may improve reconstruction quality when using randomly initialized points.

Compared to 3DGS, our method provides explicit geometry and efficient computation of the coarse-scale depth, which can be useful for shadow casting and physical collisions. In the future, we plan to explore the rendering of large-scale scenes (e.g., urban scenes) using GESs. By combining the Level of Detail (LOD) and occlusion culling, we believe GESs hold promise for real-time rendering of large urban environments.

In conclusion, we have presented a novel bi-scale representation for ultra-fast high-fidelity radiance field rendering, achieving the state-of-the-art rendering quality and much faster speeds than the existing solutions. Particularly, GESs are sorting-free, successfully avoid the popping artifacts under view changes.

ACKNOWLEDGEMENTS

This work is partially supported by NSF China (No. 62421003, 62227806 & 62322209) and the XPLORER PRIZE.

REFERENCES

Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. 2020. Neural Point-Based Graphics. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII* (Glasgow, United Kingdom). Springer-Verlag, Berlin, Heidelberg, 696–712. https://doi.org/10.1007/978-3-030-58542-6_42

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 5835–5844.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022a. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5460–5469.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022b. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF CVPR*. 5470–5479.

M. Botsch, A. Hornung, M. Zwicker, and L. Kobbelt. 2005. High-quality surface splatting on today's GPUs. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* 17–141. https://doi.org/10.1109/PBG.2005.194059

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured lumigraph rendering *(SIGGRAPH '01)*. ACM, New York, NY, USA, 425–432. https://doi.org/10.1145/383259.383309

Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.* 32, 3, Article 30 (July 2013), 12 pages.

Anpei Chen, Minye Wu, Yingliang Zhang, Nianyi Li, Jie Lu, Shenghua Gao, and Jingyi Yu. 2018. Deep Surface Light Fields. In *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. 1–17.

Guikun Chen and Wenguan Wang. 2024. A Survey on 3D Gaussian Splatting. arXiv:2401.03890 [cs.CV] https://arxiv.org/abs/2401.03890

Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. 2024. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

Abe Davis, Marc Levoy, and Fredo Durand. 2012. Unstructured light fields. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 305–314.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. arXiv:2311.17245 [cs.CV] https://arxiv.org/abs/2311.17245

Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Zhilin Pei, Hengjie Li, Xingcheng Zhang, and Bo Dai. 2024. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. *arXiv preprint arXiv:2408.07967* (2024).

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields without Neural Networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5491–5500.

Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. FastNeRF: High-Fidelity Neural Rendering at 200FPS . In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 14326–14335.

Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. 2007. Multi-View Stereo for Community Photo Collections. In *2007 IEEE 11th International Conference on Computer Vision*. 1–8. https://doi.org/10.1109/ICCV.2007.4408933

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The Lumigraph. In *Proceedings of SIGGRAPH '96*. ACM, New York, NY, USA, 43–54.

Antoine Guédon and Vincent Lepetit. 2023. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. *arXiv preprint arXiv:2311.12775* (2023).

Florian Hahlbohm, Fabian Friederichs, Tim Weyrich, Linus Franke, Moritz Kappel, Susana Castillo, Marc Stamminger, Martin Eisemann, and Marcus Magnor. 2024. Efficient Perspective-Correct 3D Gaussian Splatting Using Hybrid Transparency. arXiv:2410.08129 [cs.GR]

Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. 2024. Speedy-Splat: Fast 3D Gaussian Splatting with Sparse Pixels and Sparse Primitives. *arXiv preprint arXiv:2412.00578* (2024).

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018a. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.* 37, 6, Article 257 (Dec. 2018), 15 pages.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018b. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.* 37, 6 (2018), 1–15.

Qiqi Hou, Randall Rauwendaal, Zifeng Li, Hoang Le, Farzad Farhadzadeh, Fatih Porikli, Alexei Bourd, and Amir Said. 2024a. Sort-free Gaussian Splatting via Weighted Sum Rendering. arXiv:2410.18931 [cs.CV]

Qiqi Hou, Randall Rauwendaal, Zifeng Li, Hoang Le, Farzad Farhadzadeh, Fatih Porikli, Alexei Bourd, and Amir Said. 2024b. Sort-free Gaussian Splatting via Weighted Sum Rendering. *arXiv preprint arXiv:2410.18931* (2024).

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Article 32, 11 pages.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. 2014. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 406–413.

Kaiwen Jiang, Venkataram Sivaram, Cheng Peng, and Ravi Ramamoorthi. 2024. Geometry Field Splatting with Gaussian Surfels. *arXiv preprint arXiv:2411.17067* (2024).

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (2023).

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. 2024. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM Trans. Graph.* 43, 4, Article 62 (2024), 15 pages.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 2024. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems* 37 (2024), 80965–80986.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.* 36, 4 (2017), 1–13.

Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. 2022. Neural Point Catacaustics for Novel-View Synthesis of Reflections. *ACM Trans. Graph.* 41, 6, Article 201 (nov 2022), 15 pages.

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. 2021. Point-Based Neural Rendering with Per-View Optimization. *Computer Graphics Forum* 40, 4 (2021), 29–43.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (2020).

Christoph Lassner and Michael Zollhöfer. 2021. Pulsar: Efficient Sphere-based Neural Rendering. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1440–1449. https://doi.org/10.1109/CVPR46437.2021.00149

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. 2024. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21719–21728.

Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. In *Proceedings of SIGGRAPH '96*. Association for Computing Machinery, New York, NY, USA, 31–42.

Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 7707–7716.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural volumes: learning dynamic renderable volumes from images. *ACM Trans. Graph.* 38, 4, Article 65 (July 2019), 14 pages.

Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. 2021. Mixture of volumetric primitives for efficient neural rendering. *ACM Trans. Graph.* 40, 4, Article 59 (July 2021), 13 pages.

Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 2024. 3DGSR: Implicit Surface Reconstruction with 3D Gaussian Splatting. *ACM Trans. Graph.* 43, 6, Article 198 (2024), 12 pages.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of ECCV*. 405–421.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (Dec. 2021), 99–106.

Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 2024a. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM Trans. Graph.* 43, 6, Article 232 (Nov. 2024), 19 pages.

Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 2024b. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM Trans. Graph.* 43, 6 (2024), 1–19.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*

41, 4 (2022), 1–15.

Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. 2024. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv preprint arXiv:2403.13806* (2024).

Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. 2024. Reducing the Memory Footprint of 3D Gaussian Splatting. *Proc. ACM Comput. Graph. Interact. Tech.* 7, 1, Article 16 (May 2024), 17 pages.

Zhexi Peng, Tianjia Shao, Yong Liu, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. 2024. RTG-SLAM: Real-time 3D Reconstruction at Scale using Gaussian Splatting. In *ACM SIGGRAPH 2024 Conference Papers*. Article 30, 11 pages.

Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. 2000. Surfels: surface elements as rendering primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 335–342. https://doi.org/10.1145/344779.344936

Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. 2024. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Trans. Graph.* 43, 4, Article 64 (July 2024), 17 pages.

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 14315–14325. https://doi.org/10.1109/ICCV48922.2021.01407

Liu Ren, Hanspeter Pfister, and Matthias Zwicker. 2002. Object Space EWA Surface Splatting: A Hardware Accelerated Approach to High Quality Point Rendering. *Computer Graphics Forum* 21, 3 (2002), 461–470. https://doi.org/10.1111/1467-8659.00606 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00606

Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. 2015. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*. 765–773.

Gernot Riegler and Vladlen Koltun. 2020. Free View Synthesis. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX* (Glasgow, United Kingdom). Springer-Verlag, Berlin, Heidelberg, 623–640. https://doi.org/10.1007/978-3-030-58529-7_37

Darius Rückert, Linus Franke, and Marc Stamminger. 2022. ADOP: approximate differentiable one-pixel point rendering. *ACM Trans. Graph.* 41, 4, Article 99 (July 2022), 14 pages. https://doi.org/10.1145/3528223.3530122

Johannes L. Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4104–4113. https://doi.org/10.1109/CVPR.2016.445

Gopal Sharma, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. 2024. Volumetric rendering with baked quadrature fields. In *European Conference on Computer Vision*. 275–292.

Zhuowen Shen, Yuan Liu, Zhang Chen, Zhong Li, Jiepeng Wang, Yongqing Liang, Zhengming Yu, Jingdong Zhang, Yi Xu, Scott Schaefer, Xin Li, and Wenping Wang. 2024. SolidGS: Consolidating Gaussian Surfel Splatting for Sparse-View Surface Reconstruction. *arXiv preprint arXiv:2412.15400* (2024).

Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. *ACM Trans. Graph.* 25, 3 (July 2006), 835–846.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5449–5459. https://doi.org/10.1109/CVPR52688.2022.00538

Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. 2022. Variable Bitrate Neural Fields. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Article 41, 9 pages.

Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben P. Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. 2022. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8238–8248.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: image synthesis using neural textures. *ACM Trans. Graph.* 38, 4, Article 66 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323035

Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. 2022. Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly- Throughs. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12912–12921. https://doi.org/10.1109/CVPR52688.2022.01258

Xinzhe Wang, Ran Yi, and Lizhuang Ma. 2024. AdR-Gaussian: Accelerating Gaussian Splatting with Adaptive Radius. In *SIGGRAPH Asia 2024 Conference Papers (SA '24)*. ACM, Article 73, 10 pages.

Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. 2023. Adaptive Shells for Efficient Neural Radiance Field Rendering. *ACM Trans. Graph.* 42, 6, Article 259 (2023), 15 pages.

Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. 2020. SynSin: End-to-End View Synthesis From a Single Image. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7465–7475. https://doi.org/10.1109/CVPR42600.2020.00749

Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2021. NeX: Real-time View Synthesis with Neural Basis Expansion. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8530–8539. https://doi.org/10.1109/CVPR46437.2021.00843

Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. 2000. Surface light fields for 3D photography. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM, 287–296. https://doi.org/10.1145/344779.344925

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024a. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 20310–20320.

Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. 2024b. Recent advances in 3D Gaussian splatting. *Computational Visual Media* 10, 4 (2024), 613–642.

Chris Wyman. 2016. Exploring and expanding the continuum of OIT algorithms. In *Proceedings of High Performance Graphics*. Eurographics Association, Goslar, DEU, 1–11.

Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-NeRF: Point-based Neural Radiance Fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5428–5438. https://doi.org/10.1109/CVPR52688.2022.00536

Keyang Ye, Qiming Hou, and Kun Zhou. 2024a. 3D Gaussian Splatting with Deferred Reflection. In *ACM SIGGRAPH 2024 Conference Papers*. Article 40, 10 pages.

Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. 2024b. Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 1053–1061.

Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Trans. Graph.* 38, 6, Article 230 (Nov. 2019), 14 pages. https://doi.org/10.1145/3355089.3356513

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. PlenOctrees for Real-time Rendering of Neural Radiance Fields . In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 5732–5741.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024a. Mip-Splatting: Alias-Free 3D Gaussian Splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 19447–19456.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024b. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19447–19456.

Zehao Yu, Torsten Sattler, and Andreas Geiger. 2024c. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM Trans. Graph.* 43, 6, Article 271 (Nov. 2024), 13 pages.

Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. 2022. Differentiable Point-Based Radiance Fields for Efficient View Synthesis. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 7, 12 pages. https://doi.org/10.1145/3550469.3555413

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. Graph.* 37, 4, Article 65 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201323

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. 2001. Surface splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 371–378. https://doi.org/10.1145/383259.383300

Matthias Zwicker, Jussi Räsänen, Mario Botsch, Carsten Dachsbacher, and Mark Pauly. 2004. Perspective accurate splatting *(GI '04)*. Canadian Human-Computer Communications Society, Waterloo, CAN, 247–254.

When Gaussian Meets Surfel: Ultra-fast High-fidelity Radiance Field Rendering – Supplementary Materials

ACM Reference Format:
. 2025. When Gaussian Meets Surfel: Ultra-fast High-fidelity Radiance Field Rendering – Supplementary Materials. 1, 1 (April 2025), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 IMPLEMENTATION DETAILS OF 2D-GES

We use the same normal consistency loss $\mathcal{L}_n$ as in [Huang et al. 2024] in the Gaussian-Surfel joint optimization stage, which aligns N_{smooth} with the surface normals derived from the gradient of D_{smooth}. To ensure that the 2D Gaussians are distributed around surfels, we introduce

$$\mathcal{L}_{gd} = \text{MAE}(\frac{D_G}{W_G} - D_s), \tag{1}$$

which constrains the weighted depth of the 2D Gaussians to be close to the surfel depth. The full loss function in the joint optimization stage is $\mathcal{L} = \lambda_n \mathcal{L}_n + \lambda_{gd}\mathcal{L}_{gd} + \mathcal{L}_{rgb}$, where $\lambda_n = 0.05$ and $\lambda_{gd} = 0.1$.

In the surfel optimization stage, we also apply a similar $\mathcal{L}_n$ loss to align the surfel rendered normal map with the normals from surfel rendered depth map. Note $\mathcal{L}_n$ only performs well during the early iterations on translucent surfels. In later iterations, $\mathcal{L}_n$ has no effect on the opaque regions of the surfels with large w, because the depth gradients and normals in opaque regions are naturally consistent. To obtain the accurate surfel depth map D_s and normal map N_s as a good basic geometry, we use the alpha-blended normal map rendered at the midpoint of optimization to supervise the alpha-blended depth and normal maps rendered in later iterations. Specifically, we render the alpha-blended depth and normal maps using the equation following 2DGS [Huang et al. 2024]

$$D(\hat{\mathbf{x}}) = \sum_{i=1}^{N} d_i \alpha_i(\hat{\mathbf{x}}) \prod_{j=1}^{i-1}(1-\alpha_j(\hat{\mathbf{x}})), N(\hat{\mathbf{x}}) = \sum_{i=1}^{N} \mathbf{n}_i \alpha_i(\hat{\mathbf{x}}) \prod_{j=1}^{i-1}(1-\alpha_j(\hat{\mathbf{x}})). \tag{2}$$

At the midpoint of the surfel optimization, we render the normal maps N^{supv} for all training views. In the subsequent iterations, we use these normal maps to supervise the rendered normal maps and the surface normals obtained from the depth map gradients using

$$\mathcal{L}_{sn} = \sum_{\hat{\mathbf{x}}}(1 - N^{supv}(\hat{\mathbf{x}}) \cdot N(\hat{\mathbf{x}})), \tag{3}$$

$$\mathcal{L}_{sd} = \sum_{\hat{\mathbf{x}}}(1 - N^{supv}(\hat{\mathbf{x}}) \cdot \nabla(D(\hat{\mathbf{x}}))), \tag{4}$$

where $\nabla(\cdot)$ is the depth-to-normal operator. During surfel optimization, we use $\mathcal{L} = \lambda_n \mathcal{L}_n + \mathcal{L}_{rgb}$ in early iterations and use $\mathcal{L} = \lambda_{sn}\mathcal{L}_{sn} + \lambda_{sd}\mathcal{L}_{sd} + \mathcal{L}_{rgb}$ in later iterations, where $\lambda_n = \lambda_{sn} = \lambda_{sd} = 0.05$.

Author's address:

2025. ACM XXXX-XXXX/2025/4-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Fig. 1. Qualitative comparisons of 2D-GES with or without object space filtering.

We also implemented an OpenGL version of 2D-GES, which differs from 3D-GES only in that the second pass generates 2D Gaussians from points. It is worth noting that since the 2D Gaussian depth uses the planar depth rather than the center depth and the perspective projection is accurate [Huang et al. 2024], it is not suitable to use existing screen space filters for anti-aliasing in per-splat rendering. Therefore, we back-project the screen space EWA filter into the object space [Ren et al. 2002], converting the convolution into an adjustment of the 2D Gaussian scaling for anti-aliasing, which makes our shader implementation concise and efficient. Specifically, we denote a 2D Gaussian function as $G_{\Sigma_i}(\mathbf{x})$, where the subscript Σ_i is its covariance. The filtered 2D Gaussian function in screen space is

$$\rho_i(\hat{\mathbf{x}}) = \frac{1}{|J_i^{-1}|} G_{J_i \Sigma_i J_i^T + rI}(\hat{\mathbf{x}} - \hat{\mathbf{p}}), \tag{5}$$

where J_i is the affine approximation of the object space to screen space transformation matrix, r is the EWA filter size, $\hat{\mathbf{x}}$ and $\hat{\mathbf{p}}$ is the pixel position and the rasterized position of Gaussian center. Following [Ren et al. 2002], we back-project the screen space filter into the local Gaussian space (object space), yielding the filtered 2D Gaussian function in object space

$$\rho_i'(\mathbf{x}) = G_{\Sigma_i + r J_i^{-1} J_i^{-1^T}}(\mathbf{x} - \mathbf{p}). \tag{6}$$

We use the unit 2D Gaussian in object space, and therefore $\Sigma_i = I$. To simplify the computation, we use a scaling transformation $\mathbf{S}_i = \text{diag}(s_{i,1}, s_{i,2})$ to approximate the covariance matrix of the filtered Gaussian, which yields $\rho_i'(\mathbf{x}) \approx G_{\mathbf{S}_i \mathbf{S}_i^T}(\mathbf{x} - \mathbf{p})$. Therefore, we can simply multiply the original scaling of 2D Gaussians by the $\mathbf{S}_i$, and multiply the original opacity σ_i by $1/(s_{i,1}s_{i,2})$ before rendering to eliminate aliasing artifacts, as shown in Fig. 1.

2 NUMBER OF PRIMITIVES

In Table 1, we report the number of primitives in an outdoor scene (*Garden*) and an indoor scene (*Room*) of different methods. Our surfel pruning strategy efficiently prunes a considerable amount surfels, and only a small number of surfels are retained to produce coarse geometry and appearance. The total number of primitives in 3D-GES is close to that of AdrGS [Wang et al. 2024], but 3D-GES

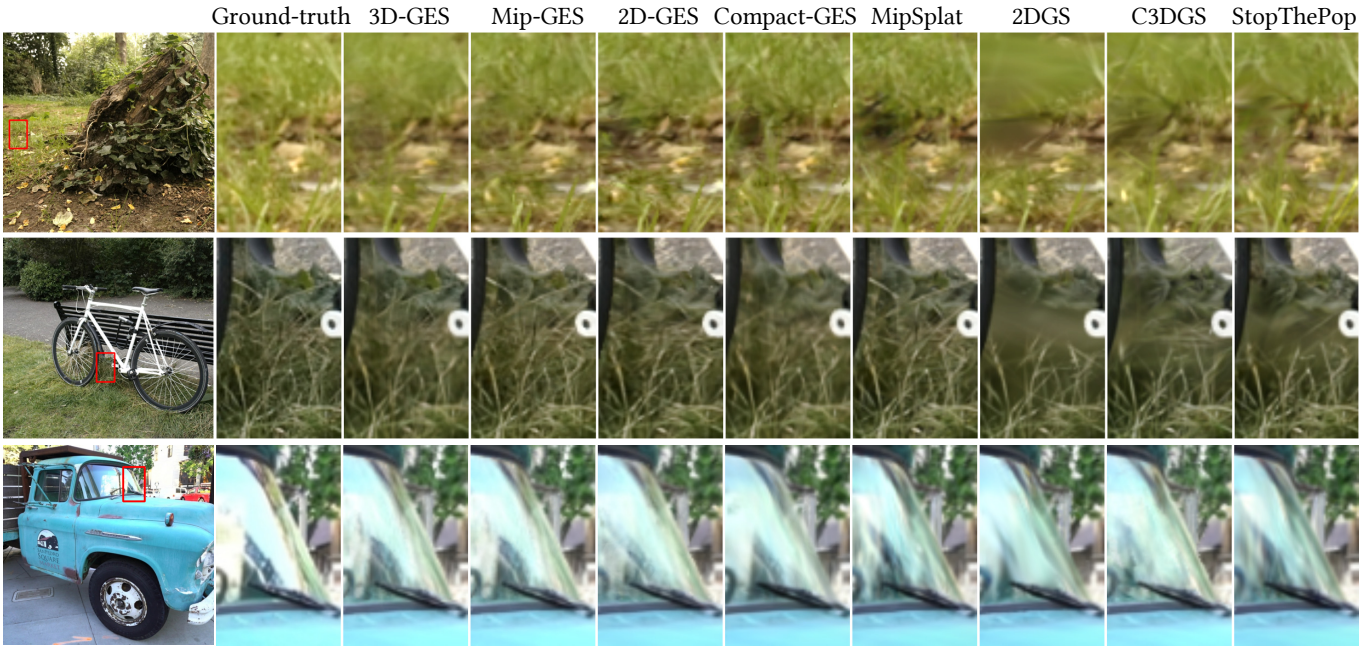

Fig. 2. More qualitative comparisons on rendering quality. From top to bottom: *Stump*, *Flowers*, *Bicycle* and *Truck*.

Table 1. The number of primitives (1M = 10^6) used by different methods in *Garden* and *Room* (from the Mip-NeRF360 dataset [Barron et al. 2022]). For our basic GES representation (3D-GES) and its extensions, we list the numbers as the sum of surfels and Gaussians separately.

	Garden	Room
3DGS	5.83M	1.59M
SortFreeGS*	4.21M	1.14M
StopThePop	5.85M	1.53M
MipSplat	6.25M	1.98M
AdrGS	2.52M	0.63M
SpeedySplat	0.53M	0.12M
C3DGS	2.23M	0.53M
2DGS	2.51M	0.87M
3D-GES	0.19+2.46M	0.15+0.55M
2D-GES	0.18+2.21M	0.13+0.58M
Mip-GES	0.18+2.48M	0.15+0.61M
Speedy-GES	0.17+0.82M	0.13+0.18M
Compact-GES	0.18+2.01M	0.14+0.50M

achieves faster frame rates and higher rendering quality, which further demonstrates the efficiency of GESs.

3 VISUALIZATION OF OPTIMIZATION

In Fig. 4, we visualize the rendering results and L1 loss over different training iterations. At the 10K-th iteration, we remove surfels with $w_i < 0.8$ and increase w_i of remaining surfels to no less than 30, resulting in a sudden spike in L1 loss. During the surfel optimization stage, due to the enlarged opaque regions of the surfels, the rendering results gradually show color discontinuity at surfel boundaries. After the 20K-th iteration, Gaussians are jointly optimized with surfels' SH coefficients, supplementing appearance details and eliminating the color discontinuities, leading to a rapid decrease in L1 loss.

Table 2. Quantitative comparisons with different methods using the PSNR metric, all trained with 50K iterations. T&T: Tanks & Temples dataset.

	Mip-NeRF360	Deep Blending	T&T
3DGS	27.48	29.43	23.90
MipSplat	27.52	29.66	23.91
SpeedySplat	26.92	29.34	23.41
3D-GES	27.38	30.00	23.95

4 MORE COMPARISONS

In Fig. 2, we present additional qualitative comparisons of rendering results from the methods not fully demonstrated in the main paper. Our GES representation and its extensions demonstrate superior ability in capturing appearance details. In particular, in *Truck*, although GESs can hardly reconstruct specular surfaces, they can learn to simulate the reflections on glass by only using Gaussians, achieving a quality comparable to SOTA methods.

In Fig. 3, we show more comparisons on geometry reconstruction quality between 2D-GES and 2DGS [Huang et al. 2024]. With the surfels serving as the primary geometry and ensuring multi-view geometry consistency, our method can reconstruct more robust

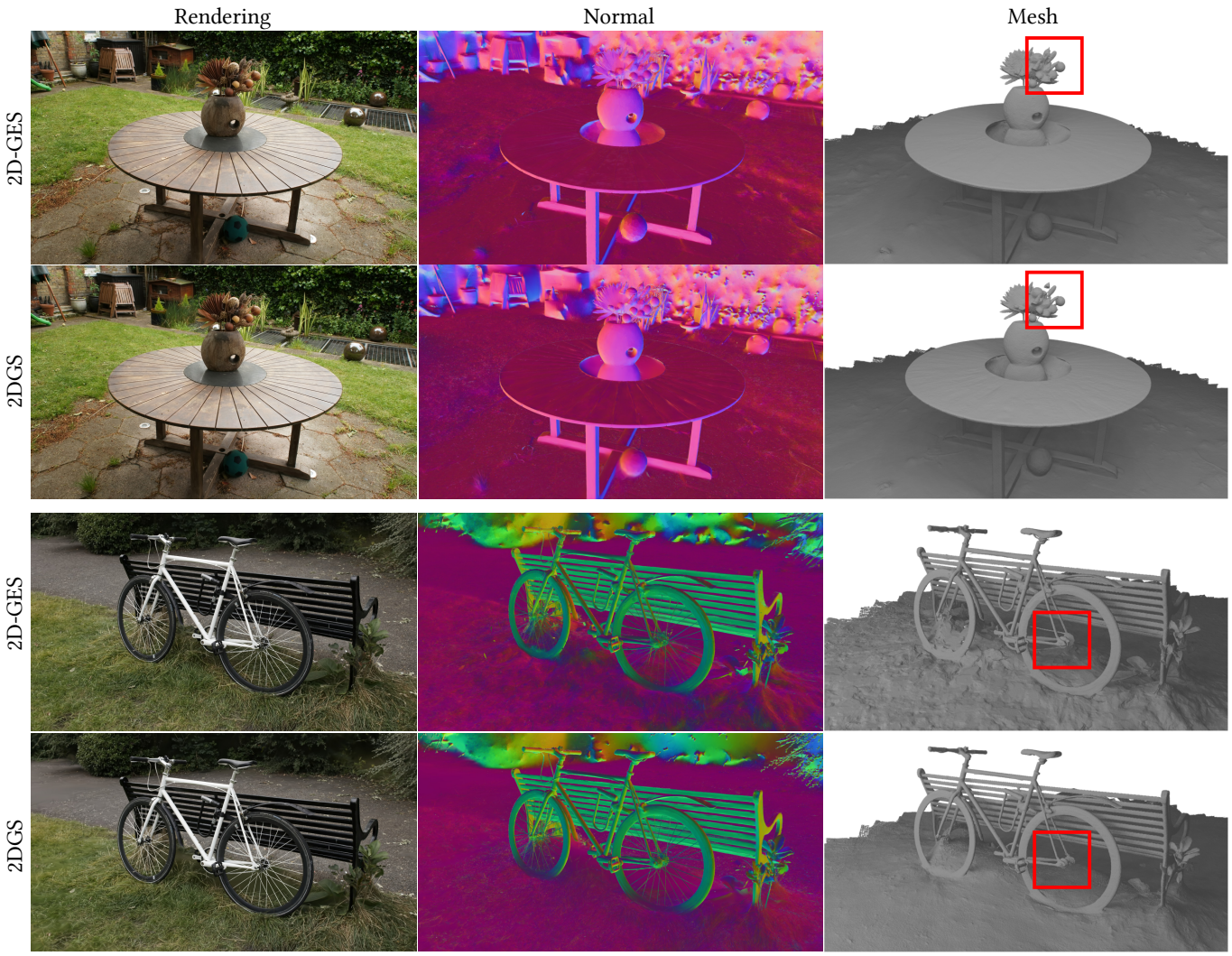

Fig. 3. More qualitative comparisons on geometry reconstruction. From top to buttom: *Garden* and *Bicycle*.

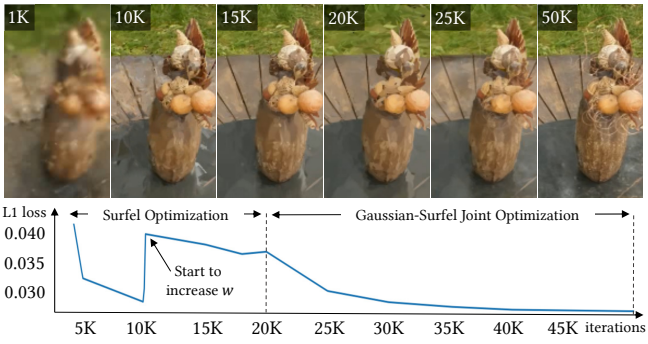

Fig. 4. Visualization of rendering results and the L1 loss in *Garden* (from Mip-NeRF360 dataset [Barron et al. 2022]) as the training step increases.

geometry. Meanwhile, the Gaussians in 2D-GES not only smooth the geometry but also introduce additional geometry details.

In Table 2, we show the quantitative comparisons with several methods, all trained with 50K iterations. Our method still achieves comparable image quality, similar to those reported in Table 1 of the main paper.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF CVPR*. 5470–5479.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Article 32, 11 pages.

Liu Ren, Hanspeter Pfister, and Matthias Zwicker. 2002. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Computer Graphics Forum*, Vol. 21. 461–470.

Xinzhe Wang, Ran Yi, and Lizhuang Ma. 2024. AdR-Gaussian: Accelerating Gaussian Splatting with Adaptive Radius. In *SIGGRAPH Asia 2024 Conference Papers (SA '24)*. ACM, Article 73, 10 pages.