RadSplat: Radiance Field-Informed Gaussian Splatting for Robust Real-Time Rendering with 900+ FPS

Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari

Google

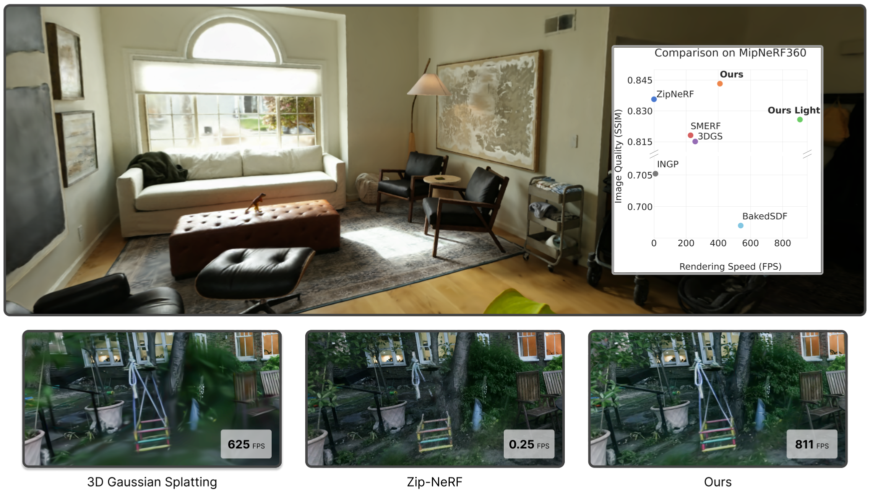
https://m-niemeyer.github.io/radsplat/

Abstract. Recent advances in view synthesis and real-time rendering have achieved photorealistic quality at impressive rendering speeds. While Radiance Field-based methods achieve state-of-the-art quality in challenging scenarios such as in-the-wild captures and large-scale scenes, they often suffer from excessively high compute requirements linked to volumetric rendering. Gaussian Splatting-based methods, on the other hand, rely on rasterization and naturally achieve real-time rendering but suffer from brittle optimization heuristics that underperform on more challenging scenes. In this work, we present RadSplat, a lightweight method for robust real-time rendering of complex scenes. Our main contributions are threefold. First, we use radiance fields as a prior and supervision signal for optimizing point-based scene representations, leading to improved quality and more robust optimization. Next, we develop a novel pruning technique reducing the overall point count while maintaining high quality, leading to smaller and more compact scene representations with faster inference speeds. Finally, we propose a novel test-time filtering approach that further accelerates rendering and allows to scale to larger, house-sized scenes. We find that our method enables state-of-the-art synthesis of complex captures at 900+ FPS.

Keywords: real-time rendering · gaussian splatting · neural fields

1 Introduction

Neural fields [6, 31, 42, 69] have emerged as one of the most popular representations for 3D vision due to their simple design, stable optimization, and state-of-the-art performance. After their introduction in the context of 3D reconstruction [6, 31, 42, 69], neural fields have been widely adopted and set new standards in tasks such as view synthesis [1, 2, 32], 3D and 4D reconstruction [25, 39, 43, 44, 46, 73], and generative modeling [27, 38, 47, 59, 64]. While neural field methods have achieved unprecedented view synthesis quality even for challenging real-world captures [2, 28, 32], most approaches are limited by



3D Gaussian Splatting

Zip-NeRF

Ours

Fig. 1: RadSplat enables high-quality real-time view synthesis for large-scale house-level scenes [2] at over 900 frames per second (top). In contrast to 3D Gaussian Splatting [21], our method allows for robust synthesis of complex captures while rendering 3,000× faster than the state-of-the-art in offline view synthesis, Zip-NeRF [2] (bottom).

the high compute costs of volumetric rendering. In order to achieve real-time frame rates, recent works reduce network complexity [12,35], cache intermediate outputs [10,56], or extract 3D meshes [41,53,72,73]. Nevertheless, all methods essentially trade reduced quality and increased storage costs for faster rendering, and are incapable of maintaining state-of-the-art quality in real-time – the goal of this work.

Recently, rasterization-based 3D Gaussian Splatting (3DGS) [21] has emerged as a natural alternative to neural fields. The representation admits real-time frame rates with view synthesis quality rivaling the state-of-the-art in neural fields. 3DGS, however, suffers from a challenging optimization landscape and an unbounded model size. The number of Gaussian primitives is not known a priori, and carefully-tuned merging, splitting, and pruning heuristics are required to achieve satisfactory results. The brittleness of these heuristics become particularly evident in large scenes where phenomena such as exposure variation, motion blur, and moving objects are unavoidable. An increasing number of primitives further leads to a potentially-unmanageable memory footprint and reduced rendering speed, strongly limiting model quality for larger scenes.

In this work, we present RadSplat, a lightweight method for robust real-time rendering of complex real-world scenes. Our method achieves smaller model sizes and faster rendering than 3DGS while strongly exceeding reconstruction quality. Our key idea is to combine the stable optimization and quality of neural fields to act as a prior and supervision signal for the optimization of point-based scene representations. We further introduce a novel pruning procedure and test-time visibility rendering strategies to significantly reduce memory usage and increase rendering speed without a corresponding loss in quality. In summary, our contributions are as follows:

1. The use of radiance fields as a prior and to handle the complexity of real-world data when optimizing point-based 3DGS representations.
2. A novel pruning strategy that reduces the number of Gaussian primitives by up to 10x whilst improving quality and rendering speed.
3. A novel post-processing step enabling viewpoint-based filtering, further accelerating rendering speed without any reduction in quality.

Our method exhibits state-of-the-art reconstruction quality on both medium and large scenes, with PSNR up to 1.87 dB higher than 3DGS and SSIM exceeding Zip-NeRF, the current state-of-the-art in offline view synthesis (see Fig. 1). At the same time, our method renders up to 907 frames per second, over 3.6× faster than 3DGS and more than 3,000× faster than Zip-NeRF.

2 Related Work

Neural Fields. Since their introduction in the context of 3D reconstruction [6, 31, 42, 69], neural fields have become one of the most promising methods for many 3D vision tasks including 3D/4D reconstruction [25, 39, 43, 44, 46, 73], 3D generative modeling [4, 27, 38, 47, 59, 64], and view synthesis [1, 2, 32]. Key to their success is among others simplicity, state-of-the-art performance, and robust optimization [69]. In contrast to previous representations such as point- [45, 48, 50], voxel- [30, 49], or mesh-based [17, 65] representations, neural fields do not usually require complex regularization, hand-tuned initialization or optimization control modules as they admit end-to-end optimization and can be queried at arbitrary spatial locations. In the context of view synthesis, Neural Radiance Fields [32] (NeRF) in particular have revolutionized the field by leveraging volumetric rendering, which has proven more robust than prior surface-based rendering approaches [40, 61, 74]. In this work, we employ the robustness and simplicity of neural fields to enable real-time rendering for complex scenes at high quality. More specifically, we use the state-of-the-art radiance field Zip-NeRF [2] to act as a robust prior and source of reliable supervision to train a point-based representation better suited for real-time rendering.

Neural Fields for Real-Time Rendering. NeRF-based models lead to state-of-the-art view synthesis but are typically slow to render so that a variety of works are proposed for speeding up training and inference. While neural fields were initially built on large, compute-heavy multi-layer perceptrons (MLPs) [6, 31, 32, 42], recent works propose the use of voxel representations and interpolation to enable fast training and rendering [12, 35, 55, 77]. Instant NGP [35], for example, demonstrates that a multi-resolution hash grid backbone enables higher quality whilst reducing training time to seconds. However, these works rely on powerful GPUs and often do not achieve real-time rendering for arbitrary scenes. Another line of work aims to represent neural fields as meshes, either as a post-processing step [53, 63] or by direct optimization [5, 18, 54, 66, 73]. These approaches can achieve high frame rates but their quality lags behind volumetric approaches. More recently, another line of work [10, 13, 19, 56] aims to represent a neural field as a set of easily-cacheable assets such as sparse voxel grids, triplanes, and occupancy grids. These methods retain their high quality but often exhibit large storage requirements, are slower to render on smaller devices, and rely on complex custom rendering implementations [10]. In contrast, we optimize lightweight point-based representations that achieve state-of-the-art quality, are easily compressed, and naturally integrate with graphics software following a rasterization pipeline.

Point-Based Representations. First works propose to render point sets as independent geometry samples [15, 16], which can be implemented efficiently in graphics software [57] and highly parallelized on GPU hardware [22, 58]. To eliminate holes when rendering incomplete surfaces, a line of works explores the "splatting" of points with extents larger than a pixel, e.g. with circular or elliptic shapes [67, 76]. The recent work 3D Gaussian Splatting (3DGS) [21] achieves unprecedented quality and fast training and rendering speed by introducing adaptive density control in combination with efficient rasterization kernels. As consequence, 3DGS is used in a variety of applications, including 3D human [79] and avatar reconstruction [9, 33, 51], 3D generation [8, 26, 62, 75], SLAM systems [24, 29, 70], 4D reconstruction [68], and open-set segmentation [52, 60]. Further, works are proposed to address aliasing [71, 78] and point densification [7] in the 3DGS representation. Finally, a recent line of works investigate compression for 3DGS [11, 23, 34, 36, 37] leading to more compact scenes and faster rendering. In this work, we combine a NeRF prior for stable optimization with a point-based 3DGS representation for real-time rendering of complex scenes. Compared to prior works, we enable high-quality view synthesis even for complex real-world captures that might contain lighting and exposure variations. Further, we develop pruning and test-time visibility rendering strategies leading to 10× fewer Gaussian primitives at higher quality compared to 3DGS and with inference times of 900+ FPS.

3 Method

Our goal is to develop a lightweight, real-time view synthesis method that is robust even for complex real-world captures. In the following, we discuss the key components for achieving this. First, we optimize radiance fields as a robust prior for complex data (Sec. 3.1). Next, we use the radiance field to first initialize and then to supervise the optimization of point-based 3DGS representations (Sec. 3.2.) We develop a novel pruning technique leading to a significant point count reduction while maintaining high quality (Sec. 3.3). Finally, we cluster input cameras and perform visibility filtering, further accelerating rendering speed to up to 900+ FPS (Sec. 3.4). We show an overview of our method in Fig. 2.

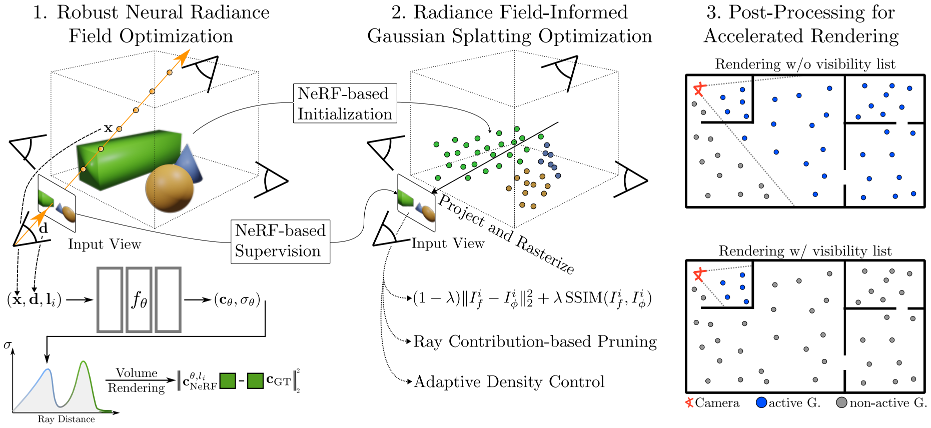

1. Robust Neural Radiance Field Optimization
2. Radiance Field-Informed Gaussian Splatting Optimization
3. Post-Processing for Accelerated Rendering
NeRF-based Initialization
NeRF-based Supervision
x
d
Input View
Input View
Project and Rasterize
$(\mathbf{x}, \mathbf{d}, \mathbf{l}_i)$
f_θ
$(\mathbf{c}_\theta, \sigma_\theta)$
σ
Ray Distance
Volume Rendering
$\|\mathbf{c}^{\theta,l_i}_{\text{NeRF}}$ 🟩 - 🟩 $\mathbf{c}_{\text{GT}}\|_2^2$
$(1-\lambda)\|I^i_f - I^i_\phi\|_2^2 + \lambda\,\text{SSIM}(I^i_f, I^i_\phi)$
Ray Contribution-based Pruning
Adaptive Density Control
Rendering w/o visibility list
Rendering w/ visibility list
Camera
● active G.
● non-active G.

Fig. 2: Overview. 1. Given posed input images of a complex real-world scene, we train a robust neural radiance field with GLO embeddings $\mathbf{l}_i$. 2. We use the radiance field prior to initialize and supervise our point-based 3DGS representation that we optimize with a novel pruning technique for more compact, high-quality scenes. 3. We perform viewpoint-based visibility filtering to further accelerate test-time rendering speed.

3.1 Neural Radiance Fields as a Robust Prior

Neural Radiance Fields. A radiance field f is a continuous function that maps a 3D point $\mathbf{x} \in \mathbb{R}^3$ and a viewing direction $\mathbf{d} \in \mathbb{S}^2$ to a volume density $\sigma \in \mathbb{R}^+$ and an RGB color value $\mathbf{c} \in \mathbb{R}^3$. Inspired by classical volume rendering [20], a pixel's final color prediction is obtained by approximating the integral via quadrature using sample points:

$$\mathbf{c}_{\text{NeRF}} = \sum_{j=1}^{N_s} \tau_j \alpha_j \mathbf{c}_j \quad \text{where} \quad \tau_j = \prod_{k=1}^{j-1}(1-\alpha_k), \quad \alpha_j = 1 - e^{-\sigma_j \delta_j} \tag{1}$$

where τ_j is the transmittance, α_j the alpha value for $\mathbf{x}_j$, and $\delta_j = ||\mathbf{x}_{j+1} - \mathbf{x}_j||_2$ the distance between neighboring sample points. In Neural Radiance Fields [32], f is parameterized as an MLP with ReLU activation f_θ and the network parameters θ are optimized using gradient descent on the reconstruction loss:

$$\mathcal{L}(\theta) = \sum_{\mathbf{r} \in \mathcal{R}_{\text{batch}}} \|\mathbf{c}^\theta_{\text{NeRF}}(\mathbf{r}) - \mathbf{c}_{\text{GT}}(\mathbf{r})\|_2^2 \tag{2}$$

where $\mathbf{r} \in \mathcal{R}_{\text{batch}}$ are batches of rays sampled from the set of all pixels / rays $\mathcal{R}$. To further boost training time and quality, Zip-NeRF [2] uses multisampling and an efficient multi-resolution grid backbone [35]. Due to the state-of-the-art performance, we adopt Zip-NeRF as our radiance field prior.

Robust Optimization on Real-World Data. Real-world captures often contain effects such as lighting and exposure variation or motion blur. Crucial for the success of neural fields on such in-the-wild data [28] is the use of Generative Latent Optimization [3] (GLO) embedding vectors or related techniques. More specifically, a per-image latent vector is optimized along with the neural field that enables explaining away these view-dependent effects

$$\mathcal{L}(\theta, \{\mathbf{l}_i\}_{i=1}^N) = \sum_{\mathbf{r}_i \in \mathcal{R}_{\text{batch}}} \|\mathbf{c}_{\text{NeRF}}^{\theta, l_i}(\mathbf{r}_i) - \mathbf{c}_{\text{GT}}(\mathbf{r}_i)\|_2^2 \tag{3}$$

where $\{\mathbf{l}_i\}_{i=1}^N$ indicates the set of GLO vectors and N the number of input images. This allows the model to express appearance changes captured in the input images without introducing wrong geometry such as floating artifacts. At test time, images can be rendered with a constant latent vector (usually the zero vector) to obtain stable and high-quality view synthesis. For all experiments, we follow [2] and optimize a per-image latent vector representing an affine transformation for the bottleneck vector in the Zip-NeRF representation.

3.2 Radiance Field-Informed Gaussian Splatting

Gaussian Splatting. In contrast to neural fields, in 3D Gaussian Splatting [21] an explicit point-based scene representation is optimized. More specifically, the scene is represented as points that are associated with a position $\mathbf{p} \in \mathbb{R}^3$, opacity $o \in [0, 1]$, third-degree spherical harmonics (SH) coefficients $\mathbf{k} \in \mathbb{R}^{16}$, 3D scale $\mathbf{s} \in \mathbb{R}^3$, and the 3D rotation $R \in SO(3)$ represented by 4D quaternions $\mathbf{q} \in \mathbb{R}^4$. Similar to (1), such a representation can be rendered to the image plane for a camera and a list of correctly-sorted points as

$$\mathbf{c}_{\text{GS}} = \sum_{j=1}^{N_p} \mathbf{c}_j \alpha_j \tau_i \quad \text{where} \quad \tau_i = \prod_{i=1}^{j-1} (1 - \alpha_i) \tag{4}$$

where $\mathbf{c}_j$ is the color predicted using the SH coefficients $\mathbf{k}$ and α_j is obtained by evaluating the projected 2D Gaussian with covariance $\mathit{\Sigma}' = J M \mathit{\Sigma} M^T J^T$, multiplied by the per-point opacity o [21], with M being the viewing transformation, J denoting the Jacobian of the affine approximation of the projective transformation [80], and $\mathit{\Sigma}$ denoting the 3D covariance matrix. To ensure that $\mathit{\Sigma}$ is a positive semi-definite matrix, it is expressed using the per-point scale matrix $S = \text{diag}(s_1, s_2, s_3)$ and rotation R according to $\mathit{\Sigma} = RSS^T R^T$ [21]. The scene is optimized with a reconstruction loss on the input images and regular densification steps consisting of splitting, merging, and pruning points based on gradient and opacity values.

Radiance Field-based Initialization. A key strength of radiance fields lies in the volume rendering paradigm [32], as opposed to prior surface rendering techniques [40, 61, 74], enabling the ability to initialize, remove, and change density freely in 3D space. In contrast, explicit point-based representations can only provide a gradient signal to already existing geometry prediction due to the rasterization-based approach. The initialization of this representation is hence a crucial property in its optimization process.

We propose to use the radiance field prior for obtaining a suitable initialization. More specifically, for each pixel / ray $\mathbf{r}$ from the set of all rays $\mathcal{R}$, we first render the median depth from our NeRF model

$$z_{\text{median}} = \sum_{i=1}^{N_z} \alpha_i \|\mathbf{x}_i\|_2 \quad \text{where} \quad \tau_{N_z} \geq 0.5 \quad \text{and} \quad \tau_{N_z-1} < 0.5 \tag{5}$$

where $\mathbf{x}_i$ are the ray sampling points. We project all pixels / rays into 3D space to obtain our initial point set

$$\mathcal{P}_{\text{init}} = \{\mathbf{p}_i\}_{i \in \mathcal{K}_{\text{random}}} \quad \text{with} \quad \mathbf{p}_i = \mathbf{r}_0(i) + \mathbf{d}_{r(i)} \cdot z_{\text{median}}(\mathbf{r}(i)) \tag{6}$$

where $\mathcal{K}_{\text{random}}$ are uniformly randomly-sampled indices for the list of all rays / pixels, $\mathbf{r}_0(\cdot)$ indicates the ray origin and $\mathbf{d}_{r(\cdot)}$ the normalized ray direction. We found the median depth estimation to perform better than other common choices such as expected depth by being exact sampling point estimates, and we found setting $|\mathcal{K}_{\text{random}}|$ to 1 million for all scenes to work well in practice. Further, we initialize

$$\begin{aligned} \mathbf{k}_i &= (\mathbf{k}_i^{1:3}, \mathbf{k}_i^{4:16}) \quad \text{where} \quad \mathbf{k}_i^{1:3} = \mathbf{c}_{\text{NeRF}}(\mathbf{r}(i)), \mathbf{k}_i^{4:16} = \mathbf{0} \\ \mathbf{s}_i &= (s_i, s_i, s_i) \quad \text{where} \quad s_i = \min_{p \in \{p \neq p_i | p \in \mathcal{P}_{\text{init}}\}} \|\mathbf{p}_i - \mathbf{p}\|_2 \end{aligned} \tag{7}$$

and set $o_i = 0.1$ and $\mathbf{q}_i$ to the identity rotation.[1] Thus, for each scene we optimize

$$\phi = \{(\mathbf{p}_i, \mathbf{k}_i, \mathbf{s}_i, o_i, \mathbf{q}_i)\}_{i=1}^{N_{\text{init}}} \tag{8}$$

Radiance Field-based Supervision. Radiance fields have been shown to excel even on real-world captures where images contain challenging exposure and lighting variations [2, 28]. We leverage this strength of radiance fields to factor out this complexity and noise of the data to provide a more cleaned up supervision signal than the possibly corrupted input images. More specifically, we render all input images with our NeRF model f_θ and with a constant zero GLO vector

$$\mathcal{I}_f = \{I_f^j\}_{j=1}^N \quad \text{where} \quad I_f^j = \{\mathbf{c}_{\text{NeRF}}^{\theta, l_{\text{zero}}}(\mathbf{r}_j(i))\}_{i=1}^{H \times W} \tag{9}$$

where $\mathbf{l}_{\text{zero}}$ indicates the zero GLO vector, H the height and W the width of the images, and $\mathbf{r}_j(\cdot)$ the rays belonging to the j-th image. We can then use these renderings $\mathcal{I}_f$ to train our point-based representations

$$\mathcal{L}(\phi) = (1 - \lambda)\|I_f^i - I_\phi^i\|_2^2 + \lambda\,\text{SSIM}(I_f^i, I_\phi^i) \quad \text{with} \quad i \sim \mathcal{U}(N) \tag{10}$$

[1] We set only $\mathbf{k}^{1:3}$ as we found to progressively optimize $\mathbf{k}$ leads to better results [21].

where $\mathcal{U}$ is the uniform distribution and we use the default value $\lambda = 0.2$. Another practical benefit of this approach is that we can train from arbitrary camera lens types due to NeRF's flexible ray casting, while the 3D Gaussian Splatting gradient formulation assumes a pinhole camera model and it is unclear how this can be efficiently extended to e.g. fisheye or more complex lens types.

3.3 Ray Contribution-Based Pruning

While 3DGS representations can be efficiently rendered thanks to rasterization, real-time performance still requires a powerful GPU and is not yet achieved on all platforms. The most important property for the rendering performance is the number of points in the scene that need to be rendered.

Importance Score. To obtain a more lightweight representation that can be rendered faster across platforms, we develop a novel pruning technique to reduce the number of Gaussians in the scene whilst maintaining high quality. More specifically, we introduce a pruning step during optimization that removes points that do not contribute significantly to any training view. To this end, we define an importance score by aggregating the ray contribution of Gaussian $\mathbf{p}_i$ along all rays of all input images

$$h(\mathbf{p}_i) = \max_{I_f \in \mathcal{I}_f, r \in I_f} \alpha_i^r \tau_i^r \tag{11}$$

where $\alpha_i^r \tau_i^r$ indicates the ray contribution for the pixel's final color prediction in (4) of Gaussian $\mathbf{p}_i$ along ray $\mathbf{r}$. We find that this formulation leads to improved results compared to concurrent works that investigate similar ideas [11, 23] as we use the exact ray contribution (as opposed to e.g. the opacity) as well as the max operator (instead of e.g. the mean) which is independent of the number of input images, hence more robust to different types of scene coverage.

Pruning. We use our importance score during optimization to reduce the overall point count in the scene while maintaining high quality. More specifically, we add a pruning step where we calculate mask values as

$$m_i = m(\mathbf{p}_i) = \mathbb{1}\left(h(\mathbf{p}_i) < t_{\text{prune}}\right) \quad \text{where} \quad t_{\text{prune}} \in [0, 1] \tag{12}$$

and we remove all Gaussians from our scene that have a mask value of one. We apply the pruning step twice over the the course of optimization similar to [11]. The threshold t_{prune} provides a control mechanism over the number of points that are used to represent the scene. In our experiments, we define two values, one value for our default model, and a higher value for a lightweight variant.

3.4 Viewpoint-Based Visibility Filtering

Our pruning technique ensures a compact scene representation with a small overall point count. To scale to larger, more complex scenes such as entire houses or apartments, we introduce a novel viewpoint-based filtering as post-processing step that further speeds up test-time rendering without a quality drop.

Input Camera Clustering. First, we group input cameras together to obtain a meaningful tessellation of the scene space. More specifically, let $(\mathbf{x}^i_{\text{cam}})^N_{i=1}$ denote the input camera locations for the set of input images $\mathcal{I}$. We run k-means clustering on the input camera locations to obtain k cluster centers $(\mathbf{x}^i_{\text{cluster}})^k_{i=1}$ and assign the input cameras to the respective cluster centers.

Visibility Filtering. Next, for each cluster center $\mathbf{x}^j_{\text{cluster}}$, we select all assigned input cameras, render the images from these viewpoints, and, similar to (11), calculate an importance score and the respective visibility indicator list

$$h^{\text{cluster}}_j(\mathbf{p}_i) = \max_{I \in \mathcal{I}^i_c, r \in I} \alpha^r_i \tau^r_i, \quad m^{\text{cluster}}_j = \mathbb{1}\left(h^{\text{cluster}}_j(\mathbf{p}_i) > t_{\text{cluster}}\right) \tag{13}$$

where $\mathcal{I}^i_c$ is the set of images whose camera positions are assigned to the the cluster center $\mathbf{x}^i_{\text{cluster}}$ and t_{cluster} is a threshold that controls the contribution value of points that should be filtered out (we found setting $t_{\text{cluster}} = 0.001$ to work well in practice). Note that we are not restricted to the input views for calculating these masks. In practice, we hence add random camera samples to $\mathcal{I}^i_c$ to ensure robustness to test views. We calculate the indicator list m^{cluster}_j for each cluster center as a post-processing step after scene optimization.

Visibility List-Based Rendering. To render an arbitrary viewpoint, we first assign its camera center $\mathbf{x}^{\text{test}}_{\text{cam}}$ to the nearest cluster center $\mathbf{x}^i_{\text{cluster}}$. Next, we select the respective indicator list m^{cluster}_i. Finally, we perform default rasterization while only considering the points that are marked as active for the respective cluster. This results in a significant FPS increase without any drop in quality.

3.5 Implementation Details

We set the number of initial points N_{init} to 1 million in all experiments. For threshold value t_{prune}, we use 0.01 and 0.25 for our default and lightweight models, respectively, and for the large Zip-NeRF scenes, we use 0.005 and 0.03. We perform pruning after 16 and 24 thousand steps. We follow [21] and use the same densification parameters except for the densification gradient threshold value which we lower to $8.6e^{-5}$ for the Zip-NeRF dataset. We train our radiance fields on 8 V100 GPUs (∼1h) and our 3DGS models on a A100 GPU (∼1h). For the visibility filtering, we use $k = 64$ clusters and we found a small threshold $t_{\text{cluster}} > 0$ to work well in practice and set it to 0.001 for all scenes. For the radiance field training, we follow [2] and use default parameters for all scenes.

4 Experiments

Datasets. We report results on the MipNeRF360 dataset [1], the most common view synthesis benchmark consisting of 9 unbounded indoor and outdoor scenes.

We further report results on the Zip-NeRF dataset [2] consisting of 4 large-scale scenes (apartments and houses) with challenging captures that partly contain lighting and exposure variations.

Baselines. On all datasets, we compare against 3DGS [21] as well as MERF [56] and SMERF [10] as the state-of-the-art volumetric approaches that construct efficient voxel and triplane representations together with accelerating structures for empty space skipping. On MipNeRF360, we further compare against mesh-based BakedSDF [73], hash-grid based INGP [35], and point-based approaches LightGaussian [11], CompactGaussian [23], and EAGLES [14]. For reference, we always report Zip-NeRF [2], the state-of-the-art offline view synthesis method.

Metrics and Evaluation. We follow common practice and report the view synthesis metrics PSNR, SSIM, and LPIPS. While using techniques such as GLO vectors is essential for high quality on real-world captures (see Sec. 3.1), the evaluation of such models is an open problem such that recent methods [1,2,10] train two separate models, one for visualizations, and one (without GLO vectors) purely for the quantitative comparison. In this work, we always train a single model that is robust thanks to the radiance field prior. For evaluation, we simply finetune the trained models on the original image data to match potential color shifts and to ensure a fair comparison. Next to measuring quality, we report the rendering speed in frames per second (FPS) on a RTX 3090 GPU and the number of Gaussians in the scenes (only applicable for point-based methods).

4.1 Real-Time View Synthesis

	SSIM↑	PSNR↑	LPIPS↓	FPS↑	#G(M)↓
INGP [35]	0.705	25.68	0.302	9.26	-
BakedSDF [73]	0.697	24.51	0.309	539	-
MERF [56]	0.722	25.24	0.311	171	-
SMERF [10]	0.818	27.99	0.211	228	-
CompactG [23]	0.798	27.08	0.247	128	1.388
LightG [11]	0.799	26.99	0.25	209	1.046
EAGLES [14]	0.809	27.16	0.238	137	1.712
3DGS [21]	0.815	27.20	0.214	251	3.161
Ours Light	0.826	27.56	0.213	907	0.370
Ours	0.843	28.14	0.171	410	1.924
Zip-NeRF [2]	0.836	28.54	0.177	0.25	-

(a) Mip-NeRF360 dataset [1]

	SSIM↑	PSNR↑	LPIPS↓	FPS↑
MERF [56]	0.747	23.49	0.445	318
SMERF [10] ($K = 1$)	0.776	25.44	0.412	356
SMERF [10] ($K = 5$)	0.829	27.28	0.340	221
3DGS [21]	0.809	25.50	0.369	470
Ours Light	0.838	26.11	0.368	748
Ours	0.839	26.17	0.364	630
Zip-NeRF [2]	0.836	27.37	0.305	0.25

(b) Zip-NeRF dataset [2]

Table 1: Quantitative Comparison. We compare against top-performing real-time rendering approaches and report offline method ZipNeRF as reference. Our models outperform both NeRF- and GS-based approaches, achieving state-of-the-art view synthesis at higher FPS. Ours Light achieves a 10× point count reduction (shown as #G in mio.) compared to 3DGS [21] while improving quality (1a). Our default model improves even over Zip-NeRF [2] in SSIM and LPIPS while rendering 3,600× faster. On the large-scale scenes in 1b, our models produce the highest SSIM values while rendering up to 3.3× faster than top-performing real-time methods such as SMERF [10].

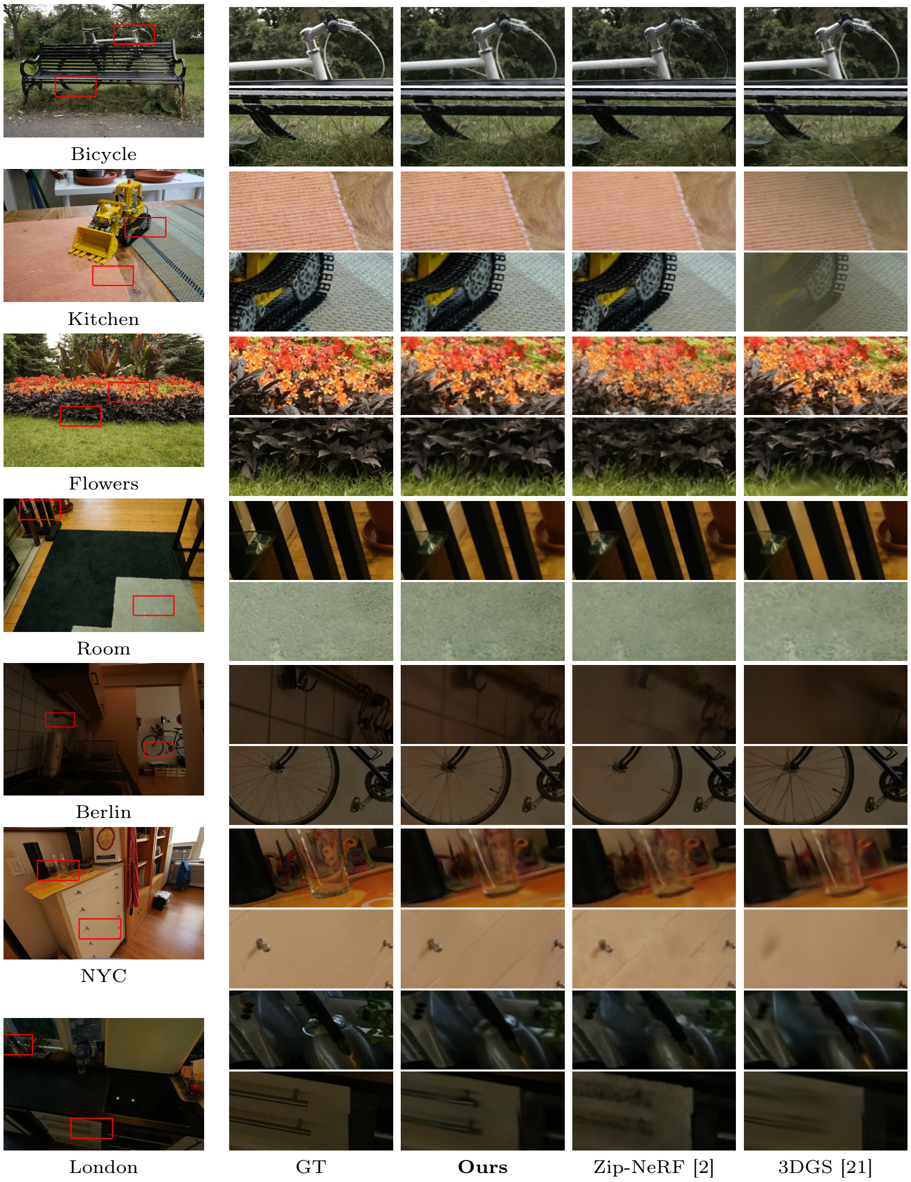


Fig. 3: Qualitative Comparison. We show results on the mip-NeRF 360 dataset [1] (top four) and Zip-NeRF dataset [2] (bottom three). Compared to Zip-NeRF, our method better captures high-frequency texture details (e.g., tablecloth in Kitchen and carpet in Room) and geometric details (e.g., leaves in Flowers and bicycle spokes in Berlin). Compared to 3DGS, we obtain sharper (e.g., grass below bench in Bicycle and shiny surfaces in London) and more stable reconstructions (e.g., color shift in Kitchen).

Unbounded Scenes. We observe in Tab. 1a that our method leads to the best quantitative results while achieving faster rendering times than prior state-of-the-art real-time methods such as SMERF [10]. Notably, our model even outperforms the state-of-the-art non-real-time method Zip-NeRF [2] in both SSIM and LPIPS while rendering 1,600× faster. Our lightweight variant ("Ours Light") also exceeds prior works with a mean rendering speed of 907 FPS outpacing even state-of-the-art mesh-based methods such as BakedSDF [73]. Also qualitatively in Fig. 3, we observe that our model achieves the best results. Compared to Zip-NeRF, our method better captures high-frequency textures (e.g., see tablecloth in "Kitchen" scene in Fig. 3) and fine geometric details (e.g., see bicycle spokes in "Bicycle" scene in Fig. 3). Compared to 3DGS, we find that our reconstructions are sharper and more stable while achieving a 2× and 10× overall point count reduction with our default and lightweight variant, respectively.

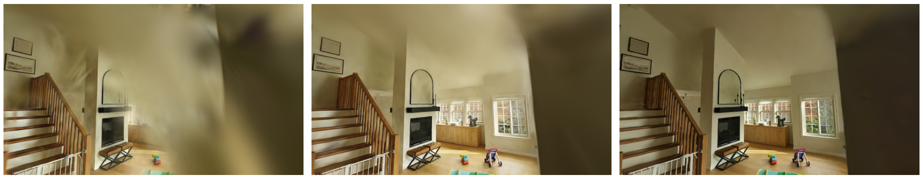
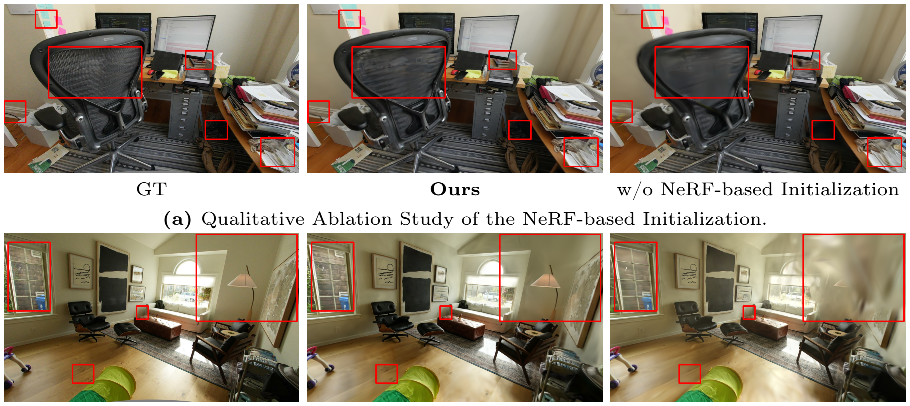

(a) 3DGS **(b)** 3DGS w/ exposure module **(c)** Ours

Fig. 4: Robustness. On complex captures with lighting variations, 3DGS [21] leads to degraded results (4a). When equipped with exposure handling modules [10, 21], results improve yet still suffer from floating artifacts and are overly smooth (4b). Our radiance field-informed approach instead achieves high quality even for challenging captures (4c).

Large-Scale Scenes. For the Zip-NeRF dataset [2], we observe a similar trend in Tab. 1b. Our default and lightweight variant outperform top-performing real-time SMERF and non-real-time Zip-NeRF in SSIM while rendering significantly faster. Notably, our lightweight variant achieves high quality with a mean SSIM of 0.838 while rendering on average at 748 FPS. In contrast, the state-of-the-art real time method for large scenes, i.e. the large variant of SMERF [10] with $5^3 = 125$ submodels, achieves a slightly lower SSIM of 0.829 with a rendering speed of 221 FPS. Also qualitatively in Fig. 3, we observe that our model achieves high visual appeal with sharper and more stable reconstructions. In contrast to 3DGS [21], we find that our method is more robust on challenging captures as shown in Fig. 4 where 3DGS leads to heavily degraded results on the "Alameda" scene. Note that results still contain floating artifacts, even when equipped with a per-image module that can handle exposure and lighting variations [10, 21]. Our method enables high-quality synthesis even for in-the-wild data.



GT **Ours** w/o NeRF-based Initialization

(a) Qualitative Ablation Study of the NeRF-based Initialization.

GT **Ours** w/o NeRF-based Supervision

(b) Qualitative Ablation Study of the NeRF-based Supervision.

	SSIM↑	PSNR↑	LPIPS↓	# Gaus. (M)↓
Ours	0.839	26.17	0.364	2.022
w/o NeRF Inititialization	0.830	25.71	0.382	1.583
w/o NeRF Supervision	0.835	25.79	0.372	1.849
w/o Pruning	0.839	26.14	0.364	3.049

(c) Quantitative Ablation Study on the Zip-NeRF Dataset.

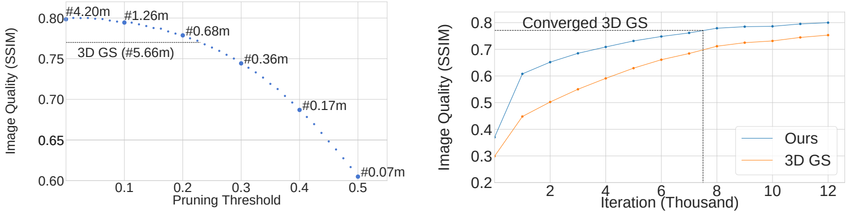
Fig. 5: Ablation Study. Without (w/o) the NeRF-based initialization, geometric and texture details might get lost (5a). Without the NeRF-based supervision, floating artifacts appear if the views exhibit lighting or exposure changes (5b), and w/o pruning, the scene point count is significantly larger without any quality improvements (5c).

4.2 Ablation Study and Limitations

NeRF-based Initialization. The NeRF-based initialization leads to better quantitative and qualitative results (see Fig. 5). In particular, smaller geometric and texture details might get lost, such as the back of the chair, the books behind the monitor, or the sticky notes on the wall in Fig. 5a.

NeRF-based Supervision. The NeRF-based supervision leads to improved results compared to optimizing the scene representation on the input views directly. In particular, for scenes where the input views exhibit exposure or lighting variations, floating artifacts are introduced to model these effects as shown in Fig. 5b. In contrast, our strategy to optimize wrt. the NeRF-based supervision is more stable and leads to better reconstructions for real-world captures.

Pruning. In Tab. 5c, we find that our model without pruning leads to similar quantitative results while exhibiting a significantly larger point count. As a result, our pruning technique enables more compact scene representations while maintaining high quality. In Fig. 6 we show that we can match the quality of 3DGS on the “Bicycle” scene, despite having roughly 10× less Gaussians in the



(a) SSIM against Pruning Threshold. **(b)** SSIM against Iteration (in Thousand).

Fig. 6: Pruning and Optimization Behavior. We show in 6a that lower pruning thresholds up to 0.1 can maintain quality while reducing the point count by 4× for the "Bicycle" scene. We match the 3DGS quality (0.77) with a 10× point reduction (5.66m vs. 0.59m). In 6b, we compare the initial optimization progression of 3DGS and our default model (pruning threshold of 0.01) on the "Bicycle" scene. We observe a steeper incline in SSIM and can match the quality of 3DGS after less than 8k steps.

scene. Further, we observe a faster increase in SSIM over the first iterations such that our default model can match the 3DGS quality even after only 8k iterations.

	SSIM↑	PSNR↑	LPIPS↓	FPS↑
Ours	0.843	28.14	0.171	410
w/o Vis. Fil.	0.843	28.14	0.171	373
Ours Light	0.826	27.56	0.213	907
w/o Vis. Fil.	0.826	27.56	0.213	887

(a) mip-NeRF 360 dataset [1]

	SSIM↑	PSNR↑	LPIPS↓	FPS↑
Ours	0.839	26.17	0.364	630
w/o Vis. Fil.	0.839	26.17	0.364	435
Ours Light	0.838	26.11	0.368	748
w/o Vis. Fil.	0.838	26.11	0.368	607

(b) Zip-NeRF dataset [2]

Table 2: Visibility Filtering. With this postprocessing, we achieve up to 10% FPS increase on scenes with a central object focus (2a) and up to 45% improvement in rendering speed when scaling to larger-scale scenes (2b) while keeping the quality fixed.

Visibility List-Based Rendering. Our visibility list-based rendering enables up to 10% mean FPS speed up on the central object-focused MipNeRF360 scenes and a up to 45% FPS increase on the larger house and apartment-level ZipNeRF scenes (see Tab. 2). We conclude that this post-processing step is in particular important when scaling to more complex larger-scale scenes.

Limitations. We optimize a radiance field and a 3DGS representation. While achieving state-of-the-art performance with 900+ FPS, our training time (approx. 2h) is longer compared to single-representation models (0.5h - 1h). Further, despite outperforming prior works on the MipNeRF360 dataset, we observe a small gap to ZipNeRF in PSNR and LPIPS on the large-scale ZipNeRF scenes. We aim to investigate how to achieve faster training and higher quality on large house- and district-level scenes.

5 Conclusion

We presented RadSplat, a method combining the strengths of radiance fields and Gaussian Splatting for robust real-time rendering of complex scenes with 900+ frames per second. We demonstrated that using radiance fields to act as a prior and supervision signal leads to improved results and more stable optimization of point-based 3DGS representations. Our novel pruning technique leads to more compact scenes with a significantly smaller point count, whilst improving quality. Finally, our novel test-time filtering further improves rendering speed without a quality drop. We showed that our method achieves state-of-the-art on common benchmarks while rendering up to 3,000× faster than prior works.

Acknowledgements. We would like to thank Georgios Kopanas, Peter Zhizhin, Peter Hedman, and Jon Barron for fruitful discussions and advice, Cengiz Oztireli for reviewing the draft, and Zhiwen Fan and Kevin Wang for sharing additional baseline results.

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: CVPR (2022)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. In: ICCV (2023)
3. Bojanowski, P., Joulin, A., Lopez-Paz, D., Szlam, A.: Optimizing the latent space of generative networks. In: ICML (2018)
4. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., Mello, S.D., Gallo, O., Guibas, L., Tremblay, J., Khamis, S., Karras, T., Wetzstein, G.: Efficient geometry-aware 3D generative adversarial networks. In: CVPR (2022)
5. Chen, Z., Funkhouser, T., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: ICCV (2023)
6. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: CVPR (2019)
7. Cheng, K., Long, X., Yang, K., Yao, Y., Yin, W., Ma, Y., Wang, W., Chen, X.: Gaussianpro: 3d gaussian splatting with progressive propagation. arXiv (2024)
8. Chung, J., Lee, S., Nam, H., Lee, J., Lee, K.M.: Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. arXiv (2023)
9. Dhamo, H., Nie, Y., Moreau, A., Song, J., Shaw, R., Zhou, Y., Pérez-Pellitero, E.: Headgas: Real-time animatable head avatars via 3d gaussian splatting. arXiv (2023)
10. Duckworth, D., Hedman, P., Reiser, C., Zhizhin, P., Thibert, J., Lucic, M., Szeliski, R., Barron, J.T.: SMERF: streamable memory efficient radiance fields for real-time large-scene exploration. arXiv (2023)
11. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ FPS. arXiv (2023)
12. Fridovich-Keil and Yu, Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022)
13. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: ICCV (2021)

14. Girish, S., Gupta, K., Shrivastava, A.: Eagles: Efficient accelerated 3d gaussians with lightweight encodings. arXiv (2023)
15. Gross, M., Pfister, H.: Point-based graphics. Elsevier (2011)
16. Grossman, J.P., Dally, W.J.: Point sample rendering. In: Rendering Techniques (1998)
17. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3d surface generation. In: CVPR (2018)
18. Guo, Y.C., Cao, Y.P., Wang, C., He, Y., Shan, Y., Zhang, S.H.: Vmesh: Hybrid volume-mesh representation for efficient view synthesis. In: SIGGRAPH Asia (2023)
19. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. In: ICCV (2021)
20. Kajiya, J.T., Herzen, B.V.: Ray tracing volume densities. In: Christiansen, H. (ed.) SIGGRAPH (1984)
21. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. SIGGRAPH (2023)
22. Laine, S., Karras, T.: High-performance software rasterization on gpus. In: SIGGRAPH (2011)
23. Lee, J., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. arXiv (2023)
24. Li, M., Liu, S., Zhou, H.: Sgs-slam: Semantic gaussian splatting for neural dense slam. arXiv (2024)
25. Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: CVPR (2023)
26. Liang, Y., Yang, X., Lin, J., Li, H., Xu, X., Chen, Y.: Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In: CVPR (2024)
27. Lin, C., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M., Lin, T.: Magic3d: High-resolution text-to-3d content creation. In: CVPR (2023)
28. Martin-Brualla, R., Radwan, N., Sajjadi, M.S.M., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: CVPR (2021)
29. Matsuki, H., Murai, R., Kelly, P.H., Davison, A.J.: Gaussian splatting slam. In: CVPR (2024)
30. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: IROS (2015)
31. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: CVPR (2019)
32. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
33. Moreau, A., Song, J., Dhamo, H., Shaw, R., Zhou, Y., Pérez-Pellitero, E.: Human gaussian splatting: Real-time rendering of animatable avatars. In: CVPR (2024)
34. Morgenstern, W., Barthel, F., Hilsmann, A., Eisert, P.: Compact 3d scene representation via self-organizing gaussian grids. arXiv (2023)
35. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. SIGGRAPH (2022)
36. Navaneet, K., Meibodi, K.P., Koohpayegani, S.A., Pirsiavash, H.: Compact3d: Compressing gaussian splat radiance field models with vector quantization. arXiv (2023)

37. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3d gaussian splatting for accelerated novel view synthesis. arXiv (2023)
38. Niemeyer, M., Geiger, A.: Giraffe: Representing scenes as compositional generative neural feature fields. In: CVPR (2021)
39. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Occupancy flow: 4d reconstruction by learning particle dynamics. In: CVPR (2019)
40. Niemeyer, M., Mescheder, L.M., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: CVPR (2020)
41. Oechsle, M., Peng, S., Geiger, A.: Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: ICCV (2021)
42. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019)
43. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: ICCV (2021)
44. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. SIGGRAPH (2021)
45. Peng, S., Jiang, C., Liao, Y., Niemeyer, M., Pollefeys, M., Geiger, A.: Shape as points: A differentiable poisson solver. NeurIPS (2021)
46. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: ECCV (2020)
47. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. In: ICLR (2022)
48. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017)
49. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: CVPR (2016)
50. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017)
51. Qian, S., Kirschstein, T., Schoneveld, L., Davoli, D., Giebenhain, S., Nießner, M.: Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In: CVPR (2024)
52. Qin, M., Li, W., Zhou, J., Wang, H., Pfister, H.: Langsplat: 3d language gaussian splatting. In: CVPR (2024)
53. Rakotosaona, M.J., Manhardt, F., Arroyo, D.M., Niemeyer, M., Kundu, A., Tombari, F.: Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. In: 3DV (2024)
54. Reiser, C., Garbin, S., Srinivasan, P.P., Verbin, D., Szeliski, R., Mildenhall, B., Barron, J.T., Hedman, P., Geiger, A.: Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. arXiv (2024)
55. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: ICCV (2021)
56. Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P.P., Mildenhall, B., Geiger, A., Barron, J.T., Hedman, P.: MERF: memory-efficient radiance fields for real-time view synthesis in unbounded scenes. SIGGRAPH (2023)
57. Sainz, M., Pajarola, R.: Point-based rendering techniques. Computers & Graphics (2004)
58. Schütz, M., Kerbl, B., Wimmer, M.: Software rasterization of 2 billion points in real time. CGIT (2022)

59. Schwarz, K., Liao, Y., Niemeyer, M., Geiger, A.: Graf: Generative radiance fields for 3d-aware image synthesis. In: NeurIPS (2020)
60. Shi, J.C., Wang, M., Duan, H.B., Guan, S.H.: Language embedded 3d gaussians for open-vocabulary scene understanding. In: CVPR (2024)
61. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: NeurIPS (2019)
62. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In: ICLR (2024)
63. Tang, J., Zhou, H., Chen, X., Hu, T., Ding, E., Wang, J., Zeng, G.: Delicate textured mesh recovery from nerf via adaptive surface refinement. arXiv (2022)
64. Tsalicoglou, C., Manhardt, F., Tonioni, A., Niemeyer, M., Tombari, F.: Textmesh: Generation of realistic 3d meshes from text prompts. arXiv (2023)
65. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: ECCV (2018)
66. Wang, Z., Shen, T., Nimier-David, M., Sharp, N., Gao, J., Keller, A., Fidler, S., Müller, T., Gojcic, Z.: Adaptive shells for efficient neural radiance field rendering. In: SIGGRAPH Asia (2023)
67. Wiles, O., Gkioxari, G., Szeliski, R., Johnson, J.: Synsin: End-to-end view synthesis from a single image. In: CVPR (2020)
68. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Xinggang, W.: 4d gaussian splatting for real-time dynamic scene rendering. In: CVPR (2024)
69. Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S.: Neural fields in visual computing and beyond. In: CFG (2022)
70. Yan, C., Qu, D., Wang, D., Xu, D., Wang, Z., Zhao, B., Li, X.: Gs-slam: Dense visual slam with 3d gaussian splatting. arXiv (2023)
71. Yan, Z., Low, W.F., Chen, Y., Lee, G.H.: Multi-scale 3d gaussian splatting for anti-aliased rendering. arXiv (2023)
72. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: NeurIPS (2021)
73. Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B.: Bakedsdf: Meshing neural sdfs for real-time view synthesis. In: SIGGRAPH (2023)
74. Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. In: NeurIPS (2020)
75. Yi, T., Fang, J., Wang, J., Wu, G., Xie, L., Zhang, X., Liu, W., Tian, Q., Wang, X.: Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In: CVPR (2024)
76. Yifan, W., Serena, F., Wu, S., Öztireli, C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. SIGGRAPH (2019)
77. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: CVPR (2021)
78. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. arXiv (2023)
79. Zheng, S., Zhou, B., Shao, R., Liu, B., Zhang, S., Nie, L., Liu, Y.: Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In: CVPR (2024)
80. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Ewa volume splatting. In: VIS (2001)