INPC: Implicit Neural Point Clouds for Radiance Field Rendering

Florian Hahlbohm[1] Linus Franke[2] Moritz Kappel[1]
Susana Castillo[1] Martin Eisemann[1] Marc Stamminger[2] Marcus Magnor[1]
[1] Computer Graphics Lab, TU Braunschweig, Germany {lastname}@cg.tu-bs.de
[2] Visual Computing Erlangen, FAU Erlangen-Nürnberg, Germany {firstname.lastname}@fau.de

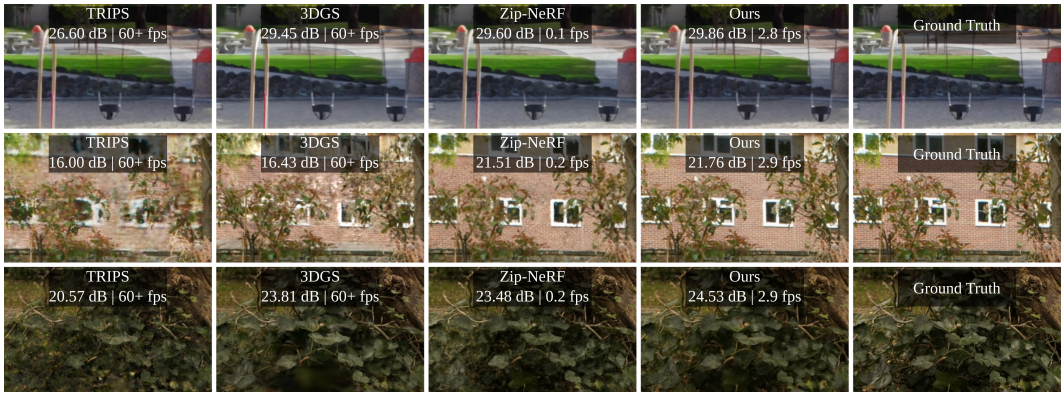

Figure 1. Novel views synthesized by our model and three state-of-the-art baselines. Our implicit point cloud optimization excels at capturing fine detail leading to a higher visual fidelity compared to baselines. While outperformed by explicit point-based methods [17, 28] in terms of inference frame rates, our model renders 17× faster than Zip-NeRF [5]. Per-patch PSNR and per-scene fps values are inset.

Abstract

We introduce a new approach for reconstruction and novel view synthesis of unbounded real-world scenes. In contrast to previous methods using either volumetric fields, grid-based models, or discrete point cloud proxies, we propose a hybrid scene representation, which implicitly encodes the geometry in a continuous octree-based probability field and view-dependent appearance in a multi-resolution hash grid. This allows for extraction of arbitrary explicit point clouds, which can be rendered using rasterization. In doing so, we combine the benefits of both worlds and retain favorable behavior during optimization: Our novel implicit point cloud representation and differentiable bilinear rasterizer enable fast rendering while preserving the fine geometric detail captured by volumetric neural fields. Furthermore, this representation does not depend on priors like structure-from-motion point clouds. Our method achieves state-of-the-art image quality on common benchmarks. Furthermore, we achieve fast inference at interactive frame rates, and can convert our trained model into a large, explicit point cloud to further enhance performance.

1. Introduction

Novel view synthesis describes the task of rendering novel inter- or extrapolated views from a set of input images, which is an inherently difficult problem. Recent methods commonly address this by reconstructing the scene either volumetrically as dense implicit radiance fields [38] or use explicit geometric representations [49] such as point clouds or meshes. Leveraging advances in optimization-based neural rendering techniques, volumetric approaches achieve an impressive increase in quality by optimizing geometric information (i.e., density and appearance) into *Multi-Layer Perceptrons (MLPs)* [35], voxel grids [18], or hash maps [36]. In contrast, explicit point-based methods optimize appearance at discrete 3D scene points, where color is being represented via 3D Gaussians [28] or per-point features [1]. While both directions provide unique advantages, they also entail severe drawbacks such as volumetric methods relying on computationally intensive ray-marching or explicit methods requiring a-priori point proxies.

In this work, we aim at creating a novel, efficient, and robust scene representation which combines the benefits of both worlds while bypassing the need for expensive ray-casting or explicit priors. To this end, we introduce Implicit Neural Point Cloud (INPC), an implicit scene representation that enables accurate scene reconstruction by sampling and subsequent rendering of explicit point clouds using fast differentiable rasterization. More specifically, we take inspiration from current state-of-the-art approaches 3D Gaussian Splatting [28] and Zip-NeRF [5]: We decompose and optimize a scene into two parts, which constitutes a concept that

we dub *implicit point clouds*. Here, the geometric structure is represented as an octree-based point probability field, while appearance features are embedded in an *implicit* coordinate-based multi-resolution hash grid. The octree is progressively subdivided to ensure similar probability distribution across all leaf nodes, which enables the reconstruction of fine articulated geometry while maximizing the capacity of our appearance hash grid. During rendering, we use the probability octree as an estimator for point positions and use either random positions in each leaf or fixed sampling patterns, while per-point appearance features are queried from the hash grid. The resulting *explicit* point cloud is then rendered via fast bilinear splatting, where gradients are backpropagated through our differentiable end-to-end pipeline to the implicit representation.

Notably, our formulation makes use of the favorable optimization properties of volumetric methods by implicitly modeling geometry and appearance. Simultaneously, it elegantly replaces interval-based sampling of per-pixel rays with a unified sampling step for the whole frustum of a given viewpoint. This subsequently allows for forward rendering with a rasterizer, i.e., the driving factor for real-time frame rates of point-based methods. By combining the benefits of both families of approaches, our method achieves robust radiance field reconstruction alongside rendering with state-of-the-art quality on benchmark datasets. In summary, our contributions are:

- The introduction of implicit neural point clouds as a data structure to effectively reconstruct unbounded 3D scenes.
- An algorithm for extracting view-specific point clouds as well as global point clouds from this model.
- A fast and differentiable rendering formulation for this data structure using bilinearly splatted points.

2. Related Work

Traditionally, novel view synthesis was based on light fields [19], however image-based rendering became a popular alternative [49]. It commonly works by warping source views onto geometric proxies [6, 12]. This proxy may contain artifacts, especially near object edges, either due to limited input coverage or misaligned cameras. With image-based rendering, these artifacts result in blurred and inaccurate images, with subsequent methods lessening these artifacts [6, 15]. The geometric proxy can also be a full 3D reconstruction, which with the introduction of *Structure-from-Motion (SfM)* [46] and *Multi-View Stereo (MVS)* [48] gained popularity. Furthermore, the advent of deep learning-based techniques in this field further improved results [51] through learned blending operators [24] and textures [52], lessening failure cases introduced by artifacts in the reconstruction. In the following, we discuss related works in volume- and point-based novel view synthesis, the two directions we combine in our work.

Neural Radiance Fields. Recently, implicitly representing 3D scenes within volumetric fields became popular, enabling novel view synthesis via volume rendering and without the need for proxy geometry. Mildenhall et al. [35] showed exceptional results by compressing a complete 3D scene into a large coordinate-based MLP, a concept called *Neural Radiance Field (NeRF)*. To render images, pixel-wise ray-marching is used with the volume rendering formulation for each pixel color C:

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

with $\alpha_i = 1 - e^{-\sigma_i \delta_i}$. Here, density σ_i and color $\mathbf{c}_i$ are the outputs of the MLP at each ray interval δ_i. Several successive works aim to resolve challenges bound to this concept by addressing input view distributions [11, 31, 58, 65] and computation times [3, 7, 11, 36, 37, 50, 53]. For the latter, discretizing the scene space using voxel grids [18], octrees [45, 64], tensor decomposition [8, 14, 41], or even distilling a faster model for inference with neurally textured triangle meshes [9] or mesh baking [42, 62] proved effective. In terms of training and rendering speed, *Instant-NGP* [36] presented exceptional results with a hash grid-based space partitioning scheme, allowing training within minutes and frame rates of up to 10 fps. Furthermore, close to our approach are hybrids of conventional and optimized ideas: Point-NeRF [60] uses an explicit point cloud with neural features, however, images are rendered with slow ray-marching and are restricted to reconstruct bounded scenes. Regarding quality, Barron et al. [3] propose anti-aliasing through conical frustum sampling and enable extension to unbounded scenes via space contraction [4]. The current state of the art in this field, *Zip-NeRF* [5], combines ideas from both quality and efficiency directions. It augments the underlying grid-based data structure with anti-aliasing through conical sampling, addition of scaling information, and refined empty space skipping. While Zip-NeRF is comparably fast in training (taking about 5 hours), the rendering speed for novel views is limited to ∼0.2 fps on consumer-grade hardware. This approach is close to our method, as we also recombine grid-based appearance information with implicit density formulation for scene reconstruction. In contrast to Zip-NeRF, however, our method enables rasterization-based rendering, making it faster in inference.

Point Rendering. Orthogonal to NeRFs, neurally rendering radiance fields via explicit points is an established and efficient methodology for novel view synthesis. Early work builds on established techniques [25] and associates point clouds obtained through MVS with optimizable colors or features [1]. Points are rendered as splats of varying sizes and a *convolutional neural network (CNN)* is used to in-

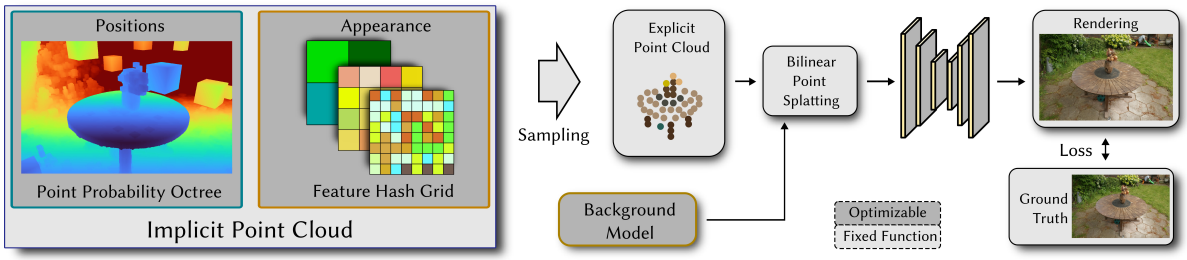

Figure 2. **Overview of our method:** We introduce the implicit point cloud, a combination of a point probability field stored in an octree and implicitly stored appearance features. To render an image for a given viewpoint, we sample the representation by estimating point positions and querying the multi-resolution hash grid for per-point features. This explicit point cloud – together with a small background MLP – is then rendered with a bilinear point splatting module and post-processed by a CNN. During optimization, the neural networks as well as the implicit point cloud are optimized, efficiently reconstructing the scene.

terpret features. Due to the unstructured and disconnected nature of point clouds, holes in image space where no point was projected to can appear, which the neural network can also resolve. Several follow-up works have been proposed eliminating training time [23, 40], adding per-view feature optimization [32], reflection warp-fields [33], or differentiable tone mapping [44]. Furthermore, differentiability with respect to point positions and camera parameters – via approximate [44] or linear gradients [17] – has been introduced, retaining rendering performance for large point clouds compared to other point representations [34, 57, 63]. Recent point-based radiance field renderers [16, 17, 28, 32, 33, 63] also adapt NeRFs rendering technique to α-blending of points. Instead of taking N samples along a ray as in Eq. (1), they blend together $\mathcal{N}$ sorted points with associated colors c and opacities α instead of computing α based on density. For approaches keeping true to MVS reconstructions, *Trilinear Point Splatting (TRIPS)* [17] is closest to our point rendering, as it also uses bilinear point splatting followed by a neural network for feature decoding. However its reconstruction quality is limited by the quality of the MVS reconstruction, as no point cloud augmentations are performed.

Overcoming the reliance on point cloud priors has been proposed in several ways. Recent improvements to MVS algorithms via CNNs [54, 61] as well as leveraging NeRFs [56] or transformers [13, 55] lessened the problem, while recent approaches also included the point cloud optimization process directly into the radiance field rendering pipeline. This is done either with additional points via error propagation [67], 3D error volumes [16], point growing via density estimation [60], or gradient-based densification [28]. Especially important here is *3D Gaussian Splatting (3DGS)* [28], which extends point rendering with anisotropic 3D Gaussians as a radiance field rendering paradigm. Since its publication, it has become the basis for a multitude of follow-up works [20, 26, 29, 39, 59, 66]. Apart from removing the need for a CNN to fill holes, the *3DGS* formulation allows starting with a sparser point cloud that is densified by repeatedly splitting large Gaussians during the optimization. In contrast, our approach captures more fine details, as we are able to optimize and render more detailed geometric and appearance information in our implicit point cloud. Furthermore, we are independent from an initial point cloud, thus increasing robustness.

3. Method

A minimal point cloud capable of novel view synthesis, is described by multiple data points consisting of a 3D position vector as well as color information, either though explicit RGB colors or decodable neural features. In our method, instead of storing these spatial and photometric properties in the same data structure, we split them and optimize both *implicitly*: Positions as an octree-based probability field $\mathcal{P}$ and appearance, i.e., colors/features and opacity, as a neural field $\mathcal{A}$. These two parts combined constitute what we call an *implicit neural point cloud*.

Input to our method is a collection of RGB images with poses and an estimated bounding box. From this, we initialize $\mathcal{P}$ as a voxel-based structure, which our algorithm iteratively refines into an octree to store probabilities for geometry (Sec. 3.1). This structure is then used as an estimator for point positions (Sec. 3.2). These positions are fed into $\mathcal{A}$, for which we use a multi-resolution hash grid [36], to retrieve opacity and spherical harmonics (SH) appearance features (Sec. 3.3). These parts are core to our proposed implicit point cloud structure, allowing us to optimize a radiance field efficiently, with fine geometric granularity, and great detail.

An overview of our method is depicted in Fig. 2. For a given viewpoint, we first obtain a set of point positions using $\mathcal{P}$ and query $\mathcal{A}$ to retrieve per-point appearance features. The resulting explicit point cloud is then rendered by first rasterizing with bilinear point splatting (Sec. 3.4) into a 2D feature image, then α-composited with the output of a background model, and post-processed by a rendering

network with a U-Net architecture [43] (Sec. 3.5). During training, our model is optimized end-to-end with losses and regularizers carefully selected for our method (Sec. 3.6). We proceed by describing all components in detail.

3.1. Sparse Point Probability Octree

Existing NeRF methods do not require explicit geometry as they place point samples along per-pixel camera rays [35]. While prior work demonstrated that this sampling scheme leads to aliasing artifacts [3], NeRF-based methods can leverage the favorable optimization properties of implicit volumetric models to achieve state-of-the-art image quality [5]. In contrast, point-based methods require a persistent set of explicit point positions during the optimization. Through optimization of point positions with approximated gradients [44] as well as handcrafted splitting, merging, and pruning operations [16, 28], these methods aim to refine an initial point cloud that is provided, e.g., as a byproduct of SfM algorithms. Importantly, the approximated position gradients as well as heuristics used for refining the point cloud, lead to a less robust optimization compared to that of implicit models.

One goal of this work is removing the need for a persistent set of point positions within a typical point-based neural rendering pipeline. In prior work, regular occupancy grids have proven to be a useful tool for skipping empty space [18, 36]. Notably, the tracked occupancy is closely related to the volume density used to represent scene geometry in volumetric NeRF-based models. Our key observation is that this value can be normalized across the whole scene and interpret as a probability for geometry. However, as the fixed voxel size of regular occupancy grids makes it difficult to accurately model complex geometry, we propose using a sparse octree to store a point probability p_i for each leaf node. During the optimization, nodes are updated, subdivided, or pruned, which allows for accurate reconstruction of fine geometric structures. By sampling the multinomial distribution represented by this octree, we can extract point clouds on demand (Sec. 3.2).

Initialization. We initialize the probability field with a uniform 3D voxel grid as the initial leaf nodes of the sparse octree with probabilities set to a uniform value to achieve equal sampling probability. *Optionally*, we incorporate a point cloud as a prior, which slightly improves initial convergence of the optimization (see supplement for details).

Probability Updates. Theoretically, the probability of each leaf node should represent how much of its volume is occupied by geometry *relative to other leaf nodes*. To this end, we employ an updating strategy inspired by occupancy grid updates in Instant-NGP [36] that combines exponential decay and knowledge from the current 3D model. During optimization, we update the probabilities of all leaf nodes (p_i) after each optimization step using the following formula:

$$p_i = \max\left(\lambda_u \cdot p_i, \max\left(\{b_0, ..., b_n\}\right)\right), \tag{2}$$

where $\{b_0, ..., b_n\}$ is the set of α-blending weights of all n points extracted from the i-th leaf node. Note that we use the transmitted blending weight $b_j = \alpha_j T_j$ (Eq. (1)) computed by our rasterizer for the updates. This imposes a visibility prior on the probabilities, as occluded points receive less transmittance. For the exponential decay, we found $\lambda_u = 0.9968$ to work well in practice and use it in all experiments. We also skip updates during the first 100 iterations for increased robustness, as $\mathcal{A}$ does not contain reliable information yet.

Subdivision. To refine the represented geometry, we want to iteratively subdivide and prune empty space in leaf nodes. Intuitively, we achieve this by identifying partially unoccupied voxels, which we locate by tracking the difference between the largest and smallest blending weight of all n points extracted from the i-th leaf node in each iteration:

$$q_i = \max\left(\lambda_u \cdot q_i, \max\left(\{b_0, ..., b_n\}\right) - \min\left(\{b_0, ..., b_n\}\right)\right), \tag{3}$$

where λ_u and $\{b_0, ..., b_n\}$ are the same as in Eq. (2). In all experiments, we subdivide leaves every 500 iterations during the optimization if q_i is above a threshold $\tau_s = 0.5$. Created leaves inherit their parents' probability and the initial value of q_i is set to 0. Note that, due to limitations of efficient multinomial sampling (see Sec. 3.2) with respect to numerical precision, we only subdivide if the resulting number of leaves is less than 256^3.

Pruning. By repeatedly applying Eq. (2), the probability of leaf nodes whose volume is empty will exponentially decay towards zero. As such we remove leaf nodes whose probability is less than $\tau_p = 0.01$ every 100 iterations. For stability, we only start pruning after the first 500 iterations.

3.2. Point Sampling Strategies

Fundamental to this work is the idea of sampling high-quality point clouds using the implicit octree-based point position estimator introduced in Sec. 3.1. Especially during optimization, we face the common issue of being limited in terms of GPU memory and are therefore restricted to a fixed budget for the number of sampled points.

This leads us to design two sampling strategies: A viewpoint-specific and a viewpoint-independent sampling scheme. For *training*, we want to generate a point cloud to render a specific viewpoint with which we can then compute the loss function in each iteration. In contrast, a global, viewpoint-independent point cloud increases temporal stability as well as rendering performance during *inference*, as no

per-frame sampling of the implicit point cloud is required. In the following, we detail the considerations that went into designing both sampling strategies.

Viewpoint-Specific Sampling. We identify three key properties of an effective re-weighting scheme for a specific viewpoint: (1) No samples should be placed outside of the view frustum, (2) regions further away from the camera require less samples, and (3) leaves with a higher subdivision level l, i.e., those representing a smaller volume, should be sampled less. Specifically, we compute the viewpoint-specific probability $\hat{p}_i$ as follows:

$$\hat{p}_i = \mathbb{1}_{visible} \cdot \frac{p_i}{d_i \cdot 2^{l_i \cdot \lambda_l}}, \tag{4}$$

where $\mathbb{1}_{visible}$ is an indicator function returning one for visible leaves and zero otherwise, λ_l controls how much less a smaller voxel should be sampled, and d_i represents the distance between leaf center and the image plane. Empirically, we found $\lambda_l = 0.5$, i.e., a small bias towards sampling smaller voxels more often, to work well as it allows the model to more accurately reconstruct regions that are well-captured in training images.

Using multinomial sampling with replacement, we convert the re-scaled probabilities to a list of leaf indices. For each element in this list, we uniformly sample the selected leaf's spatial extent to obtain the final 3D position. This improves quality, as appearance features (see Sec. 3.3) are regularized across a larger volume and not overwritten by hash-collisions.

For inference, we extend this scheme by sampling multiple point clouds, rasterizing them into 2D feature images (see Sec. 3.4), and averaging the features for each pixel. Empirically, we find this to be superior to simply increasing the number of points for inference, which can be explained by the transmittance-based rendering formulation used by our rasterizer.

Viewpoint-Independent Sampling. Motivated by the observation that sampling an implicit point cloud takes up a large chunk of the rendering time during inference, we pre-extract a global point cloud for all viewpoints. For this, we use Eq. (4) but omit the factors $\mathbb{1}_{visible}$ and d_i. Again, using multinomial sampling with replacement, we extract the number of samples for each leaf. To increase stability, we then use the 3D Halton sequence [22] for determining the final position of each sample. We consider this as an advantage of our method: Our implicit formulation can be used to extract large point clouds without having to store them on disk.

3.3. Appearance Representation

After sampling, we retrieve $M + 1$ appearance features from a multi-resolution hash grid [36] for each point. Before querying with each point's position, we apply the spherical contraction by Barron et al. [4] to increase the relative capacity of our model near the center of the scene, i.e., the best-observed region during optimization. Of the retrieved per-point features, the first one is converted to a valid opacity value $\alpha_h \in [0, 1]$ using a Sigmoid-like activation: $\alpha_h = 1 - e^{-e^x}$ [36]. The remaining M features are used as coefficients for SH evaluations to produce a view-dependent feature for each point. In practice, we use SH of degree 2 and 4D view-dependent output features, resulting in $M = 36$ SH coefficients per point. As our background model, we employ a small MLP that computes a 4D feature for each pixel using the corresponding SH-encoded viewing direction as input.

3.4. Differentiable Bilinear Point Splatting

Prior work on point rasterization demonstrated that point rendering for radiance fields can be very fast and yield great results [17, 28, 44]. However, using one-pixel point rendering (projecting and discretizing points to one pixel) leads to aliasing as well as the need for approximate gradients [44]. To avoid this, we opt to use a bilinear formulation, that is we splat each point to the closest 2×2 pixels after projection. Thus, for a point $p_w = (x, y, z)^T$, we project it to the image coordinates $p = (u, v, d)^T$ with

$$p = P \cdot V \cdot p_w, \tag{5}$$

where P and V are the intrinsic and extrinsic camera matrices respectively. For the 2×2 closest pixels' center points $p_{i \in \{0..3\}} = (u_i, v_i)^T$ we then compute the respective opacities with

$$\alpha = \alpha_h \cdot (1 - |u - u_i|) \cdot (1 - |v - v_i|). \tag{6}$$

This causes the point's contributions to be weighted correctly based on its projected position. We then use this while blending (see Eq. (1)) all points $\mathcal{N}$ in depth order to render the image.

This bilinear splatting approach has three advantages: (1) We obtain more robust gradients, (2) improve temporal stability of the rendering pipeline, and (3) the rasterized images contain less holes which simplifies the task of the hole-filling CNN. In TRIPS, Franke et al. [17] use a similar splatting approach but instead interpolate trilinearly into an image pyramid based on a learned per-point radius. Our implicit point cloud enables us to render high-quality feature images without the need for interpolation with respect to a third dimension.

3.5. Post-Processing

For decoding, we use a standard three-layer U-Net architecture with a single residual block based on *Fast Fourier Convolution (FFC)* [10]. We find that this enhances reconstruction regarding high-frequency details. For challenging,

Method	Mip-NeRF360 LPIPS↓	Mip-NeRF360 SSIM↑	Mip-NeRF360 PSNR↑	Tanks&Temples LPIPS↓	Tanks&Temples SSIM↑	Tanks&Temples PSNR↑	Train hrs↓	Render fps↑	Size GB↓
Instant-NGP [36]	0.380	0.698	25.61	0.438	0.737	21.82	0.08	5.7	0.1
ADOP [44]	0.259	0.723	23.54	0.236	0.802	21.69	7.00	67.6	0.53
TRIPS [17]	0.213	0.778	25.94	0.229	0.831	22.62	4.00	90.0	0.76
3DGS [28]	0.254	0.814	27.20	0.276	0.866	25.27	0.50	194.2	0.58
Zip-NeRF [5]	0.219	0.828	28.56	0.233	0.878	26.75	5.00	0.2	0.9
Ours	0.164	0.847	28.56	0.189	0.878	25.93	9.58	2.9	1.1
Ours (16M)	0.173	0.841	28.36	0.220	0.862	25.12	6.15	5.5	1.1
Ours (8M)	0.188	0.830	27.83	0.252	0.846	24.54	4.25	9.4	1.1
Ours (pre-ex.)	0.207	0.802	26.85	0.261	0.833	23.71	4.25	27.7	1.1

Table 1. **Quantitative comparisons** on the Mip-NeRF360 [4] and Tanks and Temples [30] datasets. The three best results are highlighted in **green** in descending order of saturation. Alongside our default configuration that uses 33M samples *(Ours)* we also provide metrics for our method when trained with less samples (*16M* and *8M*). Furthermore, we report inference results using 67M points extracted with our view-independent sampling algorithm *(pre-ex.)*.

e.g., auto-exposed outdoor scenes, we append the differentiable tone mapping module proposed by Rückert et al. [44] to our pipeline.

3.6. Optimization Loss

Inspired by prior works [21, 28], we combine a per-pixel loss and two established image-space loss functions:

$$\mathcal{L} = \mathcal{L}_{\text{R}} + \mathcal{L}_{\text{D-SSIM}} + \lambda_{\text{vgg}} \cdot \mathcal{L}_{\text{VGG}} + \lambda_{\text{decay}} \cdot \mathcal{L}_{\text{Reg}}. \quad (7)$$

Specifically, we use the robust loss $\mathcal{L}_{\text{R}}$ [2] with $\alpha = 0$ and $c = 0.2$ as our per-pixel loss as well as D-SSIM and VGG [27] losses which are commonly attributed with a closer resemblance of human perception. For regularization we follow Barron et al. [5] and impose a normalized weight decay on the parameters of the multi-resolution hash grid. We use $\lambda_{\text{vgg}} = 0.075$ and $\lambda_{\text{decay}} = 0.1$.

4. Experiments

We conduct multiple experiments to evaluate our method.

4.1. Datasets and Baselines

For evaluation, we use a total of 17 real scenes featuring a broad spectrum of challenges regarding both geometric and photometric aspects. The Mip-NeRF360 dataset [4] contains five outdoor and four indoor scenes captured with fixed exposure and white balance settings. We further use all eight scenes from the *intermediate* set of the Tanks and Temples dataset [30]. It was captured without fixed camera settings and presents challenges regarding photometric variation that complicate reconstruction, providing relevant insights for in-the-wild performance. We use the established 7:1 train/test split [4] for all scenes. We optimize for $50,000$ iterations and render an image for a single viewpoint in each of those. For further details regarding our implementation, we refer the reader to our supplemental document.

We compare our method against Instant-NGP [36], ADOP [44], TRIPS [17], 3D Gaussian Splatting [28], and the current state of the art in terms of image quality Zip-NeRF [5]. We use the images kindly provided by the authors of the respective publications when available – otherwise we use the official implementation to generate the images – and compute all image quality metrics under identical conditions. All methods use a RTX 4090 when memory was sufficient, otherwise an A100 was used.

4.2. Results

We show quantitative results for the scenes from Mip-NeRF360 and Tanks and Temples in Tab. 1, as well as qualitative comparisons in Fig. 1 and our supplemental. In terms of image quality metrics, our method clearly outperforms previous point-based techniques (TRIPS and 3DGS) and achieves similar quality as Zip-NeRF on both datasets. Visually, we observe that our method outperforms all baselines with respect to representing fine details, which is also represented in its excellent LPIPS scores. We complement our evaluation by conducting a perceptual experiment in which we compare INPC against Zip-NeRF, as the latter achieves the best quality metrics among the compared-against methods (see Tab. 1). We followed a fully randomized, within-participants experimental design with a 2AFC task. Our 17 participants saw the results of both methods side-by-side (one pair at a time, in random order and screen side, with a different order per participant) and were instructed to select the image they preferred. The 55 stimuli covered all 17 evaluated scenes and consisted of a minimum of 3 frames per scene. Our method was favored by the participants on an average of 69.41% of the cases, with all participants preferring our results with a ratio above the chance line. Our supplement contains full details on the experimental setup, stimuli selection, and evaluation.

We further show approximates for the training time, inference frame rate, and resulting model size in Tab. 1. Our method requires slightly longer training than the recent Zip-NeRF and TRIPS. Like other NeRF-based methods such as Zip-NeRF, our model always has the same size (1.1 GB),

Method	Sampling	Rendering	Post-Proc.	Total	#Points
Ours (8M)	51ms	37ms	26ms	114ms	4×8M
Ours (8M)†	16ms	10ms	26ms	52ms	8M
Ours (pre-ex.)	N/a	28ms	26ms	54ms	67M

Table 2. **Inference speed breakdown** for the *Playground* scene [30] rendered on an RTX 4090 GPU (2000×1085 pixels). The configuration marked with † is the same as in ablation F) shown in Tab. 3. The bottleneck of our rendering is the sorting step which requires 27 / 7 / 17 milliseconds (ms) respectively.

whereas point-based methods such as 3DGS require up to 2 GB of storage depending on the scene. Regarding inference frame rate, our method is roughly an order of magnitude faster than Zip-NeRF but currently outperformed by the explicit point-based approaches 3DGS, TRIPS, and ADOP.

In Tab. 2, we break down inference performance for different versions of our model that uses 8M samples during training. While this work focuses on image quality, the numbers confirm that even at FHD resolution, our model maintains its interactivity.

Ablations. In Tab. 3 and Fig. 3, we show ablations of our 8M model computed on the five outdoor scenes from the Mip-NeRF360 dataset. A) – C) dissect our loss function showing that each term has a meaningful contribution. For D), we omit octree subdivision during optimization, which greatly hinders reconstruction of the scene's foreground while having no visible impact on far away objects. E) and F) validate the effectiveness of our bilinear splatting approach as well as the point cloud multisampling during inference. Next, G) shows that our method does not depend on initial SfM points for sampling probability initialization to achieve its high visual fidelity. For H), we observe that leaving out the background model barely impacts metrics but results in sampling points in the sky (see alpha images in Fig. 3) which results in "floaters" upon visual inspection. Furthermore, I) indicates the residual FFC block in the U-Net enhances performance for high-frequency details, which is only partly captured by metrics. Lastly, we omit our post-processing, i.e., the U-Net and tone mapping module described in Sec. 3.5, in J). As this leaves our point-based method with no hole-filling technique, we double the number of samples during training to 16M and also disable λ_{vgg} due to the absence of the CNN. Somewhat surprisingly, we observe both quantitatively and qualitatively (see Fig. 3) that this configuration still provides good results, which confirms the potential of our implicit point cloud formulation.

Configuration	LPIPS↓	SSIM↑	PSNR↑
A) No D-SSIM Loss	0.204	0.729	25.27
B) No VGG Loss	0.238	0.754	25.29
C) No Weight Decay	0.210	0.744	25.22
D) No Subdivision	0.508	0.406	19.15
E) No Bilinear Splatting	0.243	0.708	24.36
F) No Multisampling	0.201	0.740	25.02
G) No SfM Prior	0.197	0.753	25.29
H) No Background Model	0.197	0.752	25.28
I) No FFC Block	0.207	0.749	25.32
J) No Post-Processing	0.277	0.718	24.32
Ours (8M)	0.192	0.761	25.53

Table 3. **Model ablations** with respect to image quality metrics computed on the outdoor scenes from the Mip-NeRF360 dataset.

5. Discussion

The experiments confirm our approach's effectiveness on common benchmarks, outperforming current state of the art regarding perceived image quality, while rendering at interactive frame rates. This is also reflected in the per-scene image quality metrics, which we include in our supplemental.

All our models use 4× multisampling for inference, which results in up to 528 million blended splats (4×33M points) per rendered feature image for our default configuration that uses 33M samples in each training iteration. We would like to highlight the results of our 8M configuration. It provides excellent quality, while training in just over 4 hours and blending 128 million splats (4×8M points) into each feature image at interactive frame rates of ∼9 fps – all on a single RTX 4090 GPU. Albeit at the cost of image quality, it allows for easy modification towards faster rendering by either disabling multisampling or by pre-extracting a global point cloud. Moreover, our view-independent sampling algorithm causes only a small drop in visual quality

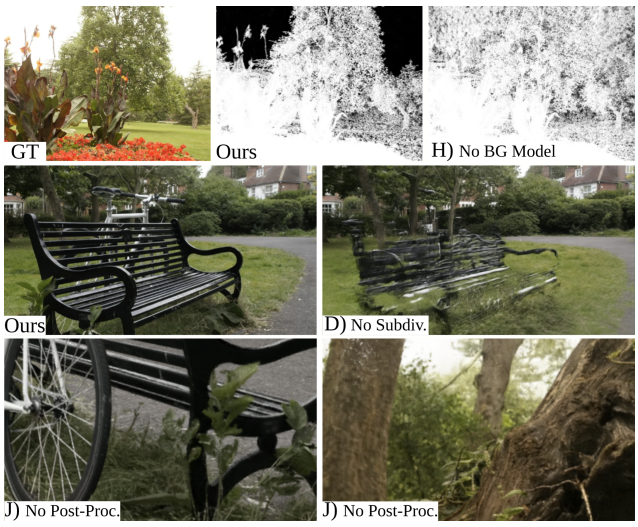

Figure 3. **Visual comparisons** for ablations H, D, and J in Tab. 3. Our background model prevents the sampling of points in the sky. Disabling octree subdivision causes foreground reconstruction to fail. Omitting post-processing (Sec. 3.5) leads to holes and high-frequency noise in renderings.

Figure 4. **Visual comparison** of INPC configurations. Our global pre-extraction slightly reduces visual quality in terms of fine detail, especially in FHD renderings of Tanks and Temples scenes. Without multisampling images are slightly sharper but our sampling sometimes misses thin structures. For Mip-NeRF360 scenes, the difference between our *8M* and default (*33M*) configuration is barely visible.

for the Mip-NeRF360 dataset, as evident by the LPIPS metric, where it still outperforms 3DGS. However, we observe that due to the higher image resolution (roughly $2\times$ more pixels) of the Tanks and Temples scenes, the budget of 67M pre-extracted global points is not always sufficient. We show visual comparisons for our default, 8M, and pre-extracted configuration in Fig. 4.

Limitations. Regarding limitations of our method, we observe that it is sometimes unable to reconstruct fine geometric detail close to the camera, a property shared with existing methods such as 3DGS and Zip-NeRF (see Fig. 5). Similar to previous works that used a CNN for post-processing rasterizer outputs [16, 44], we identify temporal stability as a minor issue during inference (see our supplemental video). While this work is mostly focused on image quality, we observe that the global sorting used in our implementation impacts its real-time rendering capability. This is a disadvantage compared to the recent 3DGS and TRIPS. However, we are confident that an optimized implementation can overcome some of the gap, as specialized rendering methods for explicit point clouds showed promising results [47].

As extracting a global point cloud greatly boosts frame rate during inference, the optimization pipeline could be adjusted to facilitate viewpoint-independent sampling, e.g., by encouraging binary opacity values as done by Reiser et al.

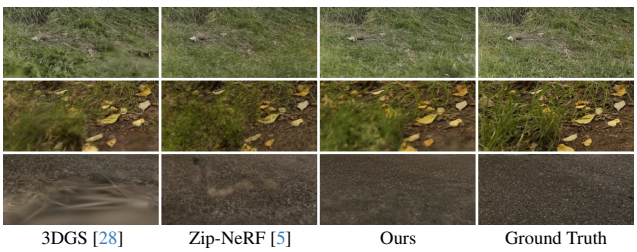

Figure 5. **Limitations.** Our method and state-of-the-art baselines sometimes fail to recover fine geometric detail near the camera.

[42]. We also believe the underlying octree-based data structure could further be improved. Lifting our implementation's limitation of 256^3 active leaf nodes in the octree, in combination with improved routines for subdivision, updating, and pruning, is likely to boost reconstruction quality.

6. Conclusion

In this work, we have introduced Implicit Neural Point Clouds, a concept fusing NeRF- and point-based radiance fields, utilizing the advantages of both. Our INPC retains favorable optimization properties of NeRF by representing a point cloud inside an octree-based probability field for point positions and an implicit appearance model. The evaluation shows that our method improves upon the previous state-of-the-art method in terms of perceptual image quality, while also enabling rendering at interactive frame rates on consumer-grade hardware. We believe that the implicit point cloud representation as well as other ideas presented here can enable future work towards further closing the gap between best-quality and real-time radiance field approaches.

https://fhahlbohm.github.io/inpc/

Acknowledgments. We would like to thank Peter Kramer for his help with the video, Timon Scholz for his help with the implementation of our viewer, and Fabian Friederichs and Leon Overkämping for their valuable suggestions. This work was partially funded by the DFG ("Real-Action VR", ID 523421583) and the L3S Research Center, Hanover, Germany. We thank the Erlangen National High Performance Computing Center (NHR@FAU) for the provided scientific support and HPC resources under the NHR project b162dc. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the DFG (ID 440719683). Linus Franke was supported by the Bavarian Research Foundation (AZ-1422-20) and the 5G innovation program of the German Federal Ministry for Digital and Transport under the funding code 165GU103B.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Eur. Conf. Comput. Vis.*, pages 696–712. Springer-Verlag, 2020. 1, 2

[2] Jonathan T. Barron. A general and adaptive robust loss function. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 4326–4334, 2019. 6

[3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Int. Conf. Comput. Vis.*, pages 5835–5844, 2021. 2, 4

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 5460–5469, 2022. 2, 5, 6

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *Int. Conf. Comput. Vis.*, pages 19640–19648, 2023. 1, 2, 4, 6, 8

[6] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32 (3), 2013. 2

[7] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *Int. Conf. Comput. Vis.*, pages 14104–14113, 2021. 2

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *Eur. Conf. Comput. Vis.*, pages 333–350. Springer-Verlag, 2022. 2

[9] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. MobileNeRF: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 16569–16578, 2023. 2

[10] Lu Chi, Borui Jiang, and Yadong Mu. Fast Fourier convolution. In *Adv. Neural Inform. Process. Syst.*, pages 4479–4488. Curran Associates, Inc., 2020. 5

[11] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo Radiance Fields (SRF): Learning view synthesis for sparse views of novel scenes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 7907–7916, 2021. 2

[12] Paul Debevec, Yizhou Yu, and George Boshokov. Efficient view-dependent IBR with projective texture-mapping. In *EG Rendering Workshop*, 1998. 2

[13] Yikang Ding, Wentao Yuan, Qingtian Zhu, Haotian Zhang, Xiangyue Liu, Yuanjiang Wang, and Xiao Liu. TransMVSNet: Global context-aware multi-view stereo network with transformers. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 8575–8584, 2022. 3

[14] Daniel Duckworth, Peter Hedman, Christian Reiser, Peter Zhizhin, Jean-François Thibert, Mario Lučić, Richard Szeliski, and Jonathan T. Barron. SMERF: Streamable memory efficient radiance fields for real-time large-scene exploration. *ACM Trans. Graph.*, 43(4), 2024. 2

[15] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating textures. *Comput. Graph. Forum*, 27(2):409–418, 2008. 2

[16] Linus Franke, Darius Rückert, Laura Fink, Matthias Innmann, and Marc Stamminger. VET: Visual error tomography for point cloud completion and high-quality neural rendering. In *SIGGRAPH Asia Conference Papers*. Association for Computing Machinery, 2023. 3, 4, 8

[17] Linus Franke, Darius Rückert, Laura Fink, and Marc Stamminger. TRIPS: Trilinear point splatting for real-time radiance field rendering. *Comput. Graph. Forum*, 43(2), 2024. 1, 3, 5, 6

[18] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 5491–5500, 2022. 1, 2, 4

[19] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In *SIGGRAPH*, pages 43–54. Association for Computing Machinery, 1996. 2

[20] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 5354–5363, 2024. 3

[21] Florian Hahlbohm, Moritz Kappel, Jan-Philipp Tauscher, Martin Eisemann, and Marcus Magnor. PlenopticPoints: Rasterizing neural feature points for high-quality novel view synthesis. In *Vision, Modeling, and Visualization*, pages 53–61. The Eurographics Association, 2023. 6

[22] John H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702, 1964. 5

[23] Mathias Harrer, Linus Franke, Laura Fink, Marc Stamminger, and Tim Weyrich. Inovis: Instant novel-view synthesis. In *SIGGRAPH Asia Conference Papers*. Association for Computing Machinery, 2023. 3

[24] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6), 2018. 2

[25] Alexander Hornung and Leif Kobbelt. Interactive pixel-accurate free viewpoint rendering from images with silhouette aware sampling. *Comput. Graph. Forum*, 28(8):2090–2103, 2009. 2

[26] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*. Association for Computing Machinery, 2024. 3

[27] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Eur. Conf. Comput. Vis.*, pages 694–711. Springer International Publishing, 2016. 6

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4), 2023. 1, 3, 4, 5, 6, 8

[29] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3D Gaussian representation for real-time rendering of very large datasets. *ACM Trans. Graph.*, 43(4), 2024. 3

[30] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4), 2017. 6, 7

[31] Georgios Kopanas and George Drettakis. Improving NeRF quality by progressive camera placement for free-viewpoint navigation. In *Vision, Modeling, and Visualization*, pages 11–20. The Eurographics Association, 2023. 2

[32] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Comput. Graph. Forum*, 40(4):29–43, 2021. 3

[33] Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. Neural point catacaustics for novel-view synthesis of reflections. *ACM Trans. Graph.*, 41(6), 2022. 3

[34] Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 1440–1449, 2021. 3

[35] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis.*, pages 405–421. Springer International Publishing, 2020. 1, 2, 4

[36] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), 2022. 1, 2, 3, 4, 5, 6

[37] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards real-time rendering of compact neural radiance fields using depth oracle networks. *Comput. Graph. Forum*, 40(4): 45–59, 2021. 2

[38] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. Graph.*, 36(6), 2017. 1

[39] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian splatting for view-consistent real-time rendering. *ACM Trans. Graph.*, 43(4), 2024. 3

[40] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. NPBG++: Accelerating neural point-based graphics. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 15948–15958, 2022. 3

[41] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Trans. Graph.*, 42 (4), 2023. 2

[42] Christian Reiser, Stephan Garbin, Pratul Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan Barron, Peter Hedman, and Andreas Geiger. Binary Opacity Grids: Capturing fine geometric detail for mesh-based view synthesis. *ACM Trans. Graph.*, 43(4), 2024. 2, 8

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Med. Image Comput. Comput.-Assist. Interv.*, pages 234–241. Springer International Publishing, 2015. 4

[44] Darius Rückert, Linus Franke, and Marc Stamminger. ADOP: Approximate differentiable one-pixel point rendering. *ACM Trans. Graph.*, 41(4), 2022. 3, 4, 5, 6, 8

[45] Darius Rückert, Yuanhao Wang, Rui Li, Ramzi Idoughi, and Wolfgang Heidrich. NeAT: Neural adaptive tomography. *ACM Trans. Graph.*, 41(4), 2022. 2

[46] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 4104–4113, 2016. 2

[47] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. Software rasterization of 2 billion points in real time. *Proc. ACM Comput. Graph. Interact. Tech.*, 5(3), 2022. 8

[48] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 519–528, 2006. 2

[49] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Vis. Commun. Image Process.*, pages 2–13. International Society for Optics and Photonics, SPIE, 2000. 1, 2

[50] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 2845–2854, 2021. 2

[51] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edith Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Nießner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhöfer, and Vladislav Golyanik. Advances in neural rendering. *Comput. Graph. Forum*, 41(2):703–735, 2022. 2

[52] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4), 2019. 2

[53] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-NeRF: Scalable construction of large-scale NeRFs for virtual fly-throughs. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 12912–12921, 2022. 2

[54] Vibhas K. Vats, Sripad Joshi, David J. Crandall, Md. Alimoor Reza, and Soon-Heung Jung. GC-MVSNet: Multi-view, multi-scale, geometrically-consistent multi-view stereo. In *IEEE/CVF Winter Conf. Appl. Comput. Vis.*, pages 3230–3240, 2024. 3

[55] Xiaofeng Wang, Zheng Zhu, Guan Huang, Fangbo Qin, Yun Ye, Yijia He, Xu Chi, and Xingang Wang. MVSTER: Epipolar transformer for efficient multi-view stereo. In *Eur. Conf. Comput. Vis.* Springer-Verlag, 2022. 3

[56] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. NerfingMVS: Guided optimization of neural radiance fields for indoor multi-view stereo. In *Int. Conf. Comput. Vis.*, pages 5590–5599, 2021. 3

[57] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 7465–7475, 2020. 3

[58] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, and Aleksander Hołyński. ReconFusion: 3D reconstruction with diffusion priors. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 21551–21561, 2024. 2

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

[60] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 5428–5438, 2022. 2, 3

[61] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *Eur. Conf. Comput. Vis.*, pages 785–801. Springer International Publishing, 2018. 3

[62] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. BakedSDF: Meshing neural SDFs for real-time view synthesis. In *ACM SIGGRAPH Conference Papers*. Association for Computing Machinery, 2023. 2

[63] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Trans. Graph.*, 38(6), 2019. 3

[64] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *Int. Conf. Comput. Vis.*, pages 5732–5741, 2021. 2

[65] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 4576–4585, 2021. 2

[66] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian splatting. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 19447–19456, 2024. 3

[67] Yiming Zuo and Jia Deng. View synthesis with sculpted neural points. In *Int. Conf. Learn. Represent.*, 2023. 3