Vision, Modeling, and Visualization (2023)
T. Grosch and M. Guthe (Eds.)

PlenopticPoints: Rasterizing Neural Feature Points for High-Quality Novel View Synthesis

Florian Hahlbohm Moritz Kappel Jan-Philipp Tauscher Martin Eisemann Marcus Magnor

Institut für Computergraphik, TU Braunschweig, Germany
{lastname}@cg.cs.tu-bs.de

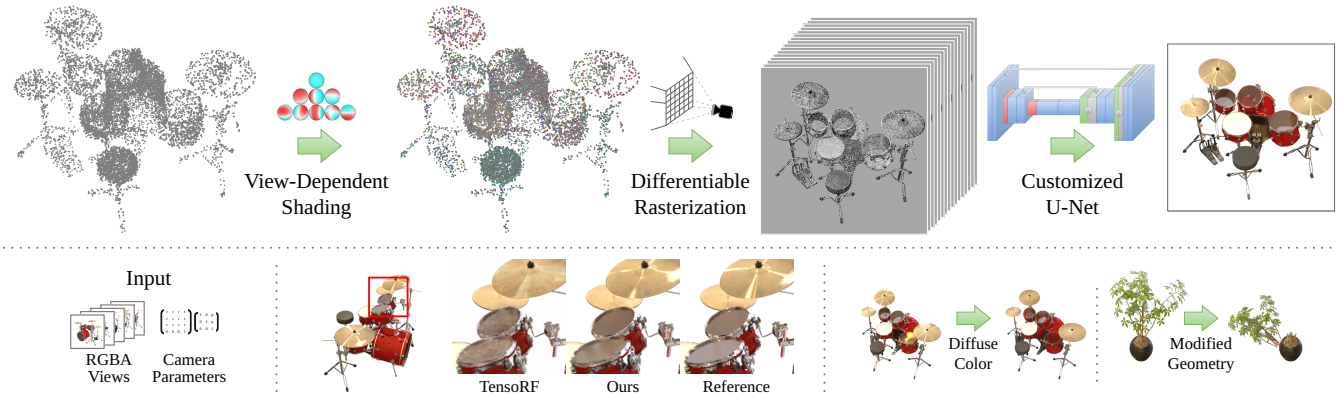

Figure 1: *Our method renders an image by rasterizing view-dependent feature points to a high-dimensional feature map that is post-processed by a rendering network. Given camera-calibrated RGBA images of an object of interest, we initialize and optimize a 3D point model. During inference, our model synthesizes novel views with improved quality and supports multiple ways of scene editing.*

Abstract

This paper presents a point-based, neural rendering approach for complex real-world objects from a set of photographs. Our method is specifically geared towards representing fine detail and reflective surface characteristics at improved quality over current state-of-the-art methods. From the photographs, we create a 3D point model based on optimized neural feature points located on a regular grid. For rendering, we employ view-dependent spherical harmonics shading, differentiable rasterization, and a deep neural rendering network. By combining a point-based approach and novel regularizers, our method is able to accurately represent local detail such as fine geometry and high-frequency texture while at the same time convincingly interpolating unseen viewpoints during inference. Our method achieves about 7 frames per second at 800×800 pixel output resolution on commodity hardware, putting it within reach for real-time rendering applications.

CCS Concepts

• ***Computing methodologies*** → ***Image-based rendering; Point-based models;***

1. Introduction

Leveraging the rapidly increasing performance of neural rendering methods, research on novel view synthesis (NVS) has reached unprecedented levels of image quality and general performance. NVS is closely related to image-based rendering and describes the task of generating images of a scene for unseen viewpoints given only a few images. Many recent methods enable this by precisely reconstructing the scene inside a complex model. The reconstructed scene can then be viewed from arbitrary viewing angles and positions alleviating the need for expensive manual 3D modeling. Two main branches of work exist. Firstly, proxy-based approaches employ discrete estimates of a scene's geometric structure using point clouds or triangle meshes. Secondly, neural field approaches model the scene in a continuous manner through large multi-layer perceptrons (MLPs) or trilinearly interpolated voxel grids. NeRFs use large MLPs to represent view-dependent density and color for

DOI: 10.2312/vmv.20231226

any 3D position and viewing direction [MST*20]. In conjunction with volume rendering this implicit representation produces high-quality images. A core problem of NeRF is the large amount of MLP queries required to compute the color of a pixel leading to multiple days of training time and slow inference rendering. Additionally, MLPs struggle with learning high-frequency functions causing NeRF to inaccurately represent fine details in geometry and texture. Follow-up works such as Plenoxels [FKYT*22] explore the possibility of trading memory usage for faster rendering through an explicit voxel grid representation. More recently, Instant-NGP [MESK22] and TensoRF [CXG*22] employ such voxel grids to achieve fast optimization, rendering, and unprecedented image quality.

On the other side, methods that employ geometric proxies are also able to avoid the aforementioned problems. Firstly, they can model fine details in geometry and texture, by allocating more primitives in regions where high levels of detail are required. Secondly, they can avoid the expensive volume rendering by simply replacing it with rasterization. We make use of point clouds in this work. Existing point-based methods are optimized for large real-world [RFS22] or forward-facing [ZD23] scenes. As a result, point-based methods usually lag behind implicit methods in reconstructing singular objects. Ideally, the entire hemisphere around an object should be photographed inwards to enable high-quality digitization. Additionally, the background can be removed beforehand. Our method extends prior work on point-based models by focusing on this exact scenario. While neural field approaches already produce impressive results for these scenes, we show that point-based models can achieve competitive or even superior visual quality. A common drawback of point-based models are holes in the rasterized images due to missing connectivity information. The density of the point cloud and the image resolution primarily drive the severity of this issue. Inspired by previous work on point-based NVS [ASK*20], we use a deep neural rendering network with a customized U-Net architecture [RFB15] to resolve this problem. Moreover, point-based models require additional regularization compared to neural fields modeled through MLPs. To address this, we combine an optimization procedure and multiple regularizers guiding the model to only use points located near object surfaces. Furthermore, previous point-based methods require an initial point cloud, which is usually acquired from multi-view stereo methods or depth images. In this work, we show a regular grid within the visual hull of the object is sufficient for obtaining high-quality results.

In summary, our core contributions are:

- A point-based, end-to-end method capable of state-of-the-art novel view synthesis for both synthetic and real-world single-object scenes.
- An optimization procedure for point-based models requiring no input point cloud but instead refining a high-quality point cloud from a regular grid.
- Multiple regularizers fine-tuned for the presented model.

Our source code is available at https://graphics.tu-bs.de/publications/hahlbohm2023plenopticpoints.

2. Related Work

The problem of novel view synthesis (NVS) has been very actively researched in recent years. The methods can be grouped into those based on neural fields (implicit and explicit volumetric models) as well as those based on geometric proxies. For methods based on geometric proxies, we specifically focus on point-based models. The recent state-of-the-art report by Tewari et al. contains further details on neural rendering in general [TTM*22].

Implicit Volumetric Representations. Early work, such as the Lumigraph from 1995, focuses on the reconstruction of a light field [GGSC96]. Local Light Field Fusion (LLFF) by Mildenhall et al. is a more recent method that leverages neural networks [MSOC*19]. In 2020, Mildenhall et al. published their seminal work on Neural Radiance Field (NeRF) reinvigorating interest in the field [MST*20]. NeRFs produce high-quality images using a large multi-layer perceptron (MLP) but have a slow rendering speed and difficulties reproducing high-frequency details such as fine geometry and specular effects. Barron et al. improve NeRF to represent fine details by sampling inside the conical frustum of each pixel [BMT*21]. It was further extended to work for unbound 360° scenes by introducing a non-linear scene parametrization, online distillation, and a distortion-based regularizer [BMV*22]. Ref-NeRF addresses NeRF's limitation of accurately representing glossy surfaces by explicitly modeling spatially-varying scene properties such as surface normals [VHM*22]. A plethora of other works improving image quality in various scenarios exist [MBRS*21, MHMB*22, TCY*22], while others address the large computational requirements of evaluating NeRF models [GKJ*21, YLT*21, HSM*21, CFHT23]. VaxNeRF reduces the number of training iterations required to train the model by constraining all samples along each pixel ray to be located inside the visual hull of the scene [KIT*21]. Our system leverages the visual hull in a similar manner, as we use it to construct the initial set of points. Point-NeRF models a neural field using a set of neural feature points [XXP*22]. In contrast to our work, Point-NeRF employs NeRF's volume rendering to generate images with its main advantage being faster per-scene training times through efficient initialization of the used point cloud.

Explicit Volumetric Representations. In contrast to the implicit MLP-based model of NeRF, multiple follow-up works have demonstrated that continuous volumetric radiance fields can be modeled explicitly using voxel grids. Plenoxels show that the combination of a sparse voxel grid, spherical harmonics shading, and trilinear interpolation can replace NeRF's MLP [FKYT*22]. Efficiently implemented in CUDA, Plenoxels reaches comparable image quality significantly faster than NeRF. Concurrent work to Plenoxels uses a similar voxel grid but models view-dependent color using a small MLP [SSC22]. More recently, Instant-NGP drastically reduces the computational requirements for both training and inference by employing a sophisticated hashing strategy to represent a feature grid inside the scenes volume [MESK22]. Chen et al. show further performance improvements can be achieved through the application of tensor decomposition methods [CXG*22]. As the decomposition of a voxel grid into vectors or matrices, is a very effective method for implicit regularization, their

method achieves fast optimization and matches the image quality of leading implicit volumetric representations. Concurrent work combines a sparse feature grid with 2D feature planes to obtain a memory-efficient representation that works well with unbounded scenes [RSV*23]. The renderings produced by the aforementioned methods sometimes contain grid-artifacts in areas with fine details. The U-Net within our rendering pipeline prevents similar artifacts often leading to more detailed and smoother renderings of surfaces with high-frequency textures.

Point-based Representations. Various works from recent years have used point-based geometric proxies for NVS. Aliev et al. extract a point cloud from RGB-D images and attach neural descriptors that are rendered by a deep convolutional neural network [ASK*20]. The method was further extended in 2022 by alleviating the need for per-scene optimization and improving the model's ability to represent view-dependent effects [RALB22]. In comparison, our method does not require any depth information as input and specializes in high-quality per-scene results instead of cross-scene generalization. In 2021, Lassner and Zollhöfer introduce Pulsar, which combines a sphere-based scene representation and a differentiable blending function [LZ21]. They show qualitatively that Pulsar is able to produce good results in the NVS setting we focus on in this work. We use the differentiable rasterization function presented in Pulsar in our method, but leverage only a small subset of Pulsar's capabilities. Instead, we introduce several additions to the image formation pipeline leading to visible improvements in image quality. Rückert et al. employ one-pixel point rasterization to achieve high performance in terms of both image quality and rendering speed [RFS22]. In contrast, our method does not require an input point cloud, no point normals, and employs view-dependent shading of points. Furthermore, point-based models have been used for NVS via per-view interpolation of neighboring images [KPLD21], in large driving scenarios [OLN*22], and accurate representation of specular effects in neural rendering [KLR*22]. A more recent method, proposes a point-based method for NVS that drastically reduces memory requirements and computational cost during training and inference [ZBRH22]. Our method makes use of foreground masks in a similar manner but produces images of higher quality at the cost of computation time. We achieve this by employing a vastly different optimization strategy and image formation pipeline. Recent work proposes another point-based model that uses Pulsar [LZ21] as the rasterization backbone [ZD23]. They initialize a point cloud using depth information predicted by a pre-trained multi-view stereo network and refine it iteratively, while our method only requires foreground masks. Furthermore, we design our method for 360° scenes, e. g. the synthetic scenes used in NeRF [MST*20], while their method performs best on forward-facing scenes. Concurrent work achieves improved results for 360° real-world scenes through splatting of 3D Gaussians [KKLD23].

3. Method

In this section, we introduce all components used for initialization, optimization and inference.

3.1. Initialization

Similar to any other explicit model, an initialization strategy is required to produce a model that can be optimized. The point cloud used by our pipeline is constructed in a multi-step procedure using the foreground masks of the input RGB images. We first determine a bounding box and its fill factor, i. e. an approximation of how much space inside the bounding box is occupied by geometry. To achieve this, we construct a point cloud P on the vertices of a regular grid with a resolution and size sufficient for accurate estimation of the aforementioned properties. By projecting every point $\mathbf{p} \in P$ into each foreground mask $m \in M$ using the projection C_m of the respective camera, we can eliminate points that are part of the background in any image:

$$\hat{P} = \{\mathbf{p} \mid \mathbf{p} \in P \wedge \forall m \in M : m(C_m(\mathbf{p})) > 0\} \tag{1}$$

Where all remaining points $\hat{P}$ are located inside the visual hull [Lau94] of the scene. A visualization of this is depicted in Figure 2. Note, that the result of this procedure depends on foreground mask quality. For real-world scenes, artifacts may occur due to the imperfect nature of state-of-the-art foreground segmentation techniques. To complete the initialization of our model, we then attach an opacity value $o \in [0,1]$ and shading features $\omega \in \mathbb{R}^{144}$ to each point defined by its position $p \in \mathbb{R}^3$. Therefore, we represent a scene consisting of N points as a set $S = \{(p_i, o_i, \omega_i)\}_i^N$. Initially, we set the opacity value of each point to 1 and all of its shading features to 0.

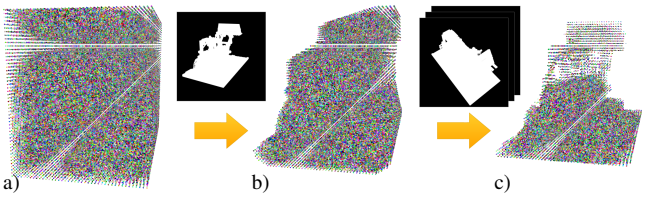

Figure 2: *a) We initialize feature points at the vertices of a regular grid. b) – c) Using segmentation masks we remove points outside the visual hull of the object (b) single mask, c) multiple masks). For better readability, we subsample the points and assign random colors.*

3.2. View-Dependent Shading

We model view-dependent appearance of the i-th point within its 144 shading features ω_i. Each $\omega_{ijk} \in \omega_i$ represents a weight for a basis function defined on the sphere. Spherical harmonics (SH) is a standard basis for functions defined on the surface of a sphere that is frequently used in computer graphics due to its flexibility and computational efficiency. When rendering a point $\mathbf{p}_i$ from a specific viewing direction $\mathbf{d} \in \mathbb{R}^3$, we use the first nine SH functions Y_j to convert its view-independent feature vector $\omega_i \in \mathbb{R}^{144}$ to a view-dependent feature vector $\hat{\omega}_i \in \mathbb{R}^{16}$:

$$\hat{\omega}_{ik} = \sum_{j=0}^{8} \omega_{i(9j+k)} \cdot Y_j(\mathbf{d}), \quad k \in [0, \ldots, 15] \tag{2}$$

We empirically determine using the first nine SH functions leads to the best results (see Section 4.3).

3.3. Differentiable Rasterization

We use a rendering scheme based on rasterization to convert the view-dependent scene representation $\hat{S} = \{(p_i, o_i, \hat{\omega}_i)\}_i^N$ to a feature map F. While traditional rasterization is not differentiable, several solutions for approximating the respective derivatives have been proposed. We use an algorithm called Pulsar that represents points as small spheres [LZ21]. To compute the values for each pixel, Pulsar employs a differentiable blending function. While Pulsar can provide gradients for most camera parameters as well as sphere positions and radii, we decide against optimizing these properties to reduce the computational complexity. We detail our configuration in Section 1 of our supplemental material and point the reader to the original publication by Lassner and Zollhöfer for further mathematical and implementation-specific details [LZ21].

3.4. Feature Map Post-Processing

A limitation of point-based models is the fact that rendered images may contain holes depending on the resolution of the rendered image relative to the number of points. While the uniform distribution of points induced by our initialization strategy minimizes the size of these holes, it does not eliminate them. Similar to previous work [ASK*20, RFS22, ZD23], we use a convolutional neural network with a U-Net [RFB15] architecture to convert the 16-channel feature map to the final RGB image. Depending on the resolution of the rendered image relative to the number of rasterized points, this rendering network is also responsible to fill any holes within the feature map. Empirically, we determine that the following modifications to the original U-Net architecture lead to improved results: Using three instead of five down-/upsampling layers, omitting batch-normalization layers, and using average instead of maximum pooling for downsampling (see Section 1 of our supplemental material for details). Similar effects with respect to these modifications were observed in previous work [RFS22, ZD23]. For a given feature map F, we obtain the final RGB prediction $\hat{I}$ by remapping the output of the U-Net according to the following equation:

$$\hat{I} = \frac{\mathcal{U}(F) + 1}{2} \tag{3}$$

This allows the U-Net to operate in $[-1, 1]$ while still predicting valid RGB values in $[0, 1]$. Empirically, we find this remapping operation produces better results compared to the commonly used Sigmoid function.

3.5. Losses and Regularization

In each training iteration we use our full model to predict an RGB image $\hat{I}$ for a randomly selected training viewpoint. The base of our loss function consists of two separate loss terms both computed using the respective ground truth training image I. We combine a plain L1 loss with a perceptual loss $\mathcal{L}_{VGG}$ based on VGG features [SZ15] defined as:

$$\mathcal{L}_{VGG} = \mathcal{F}_{VGG}(I, \hat{I}) \tag{4}$$

Where $\mathcal{F}_{VGG}(I, \hat{I})$ computes the distance between I and $\hat{I}$ given the network $\mathcal{F}_{VGG}$ as proposed by Johnson et al. [JAFF16] Importantly, this loss term is different from the LPIPS [ZIE*18] metric we use for evaluation. Although they use the same pre-trained VGG network, our ablation study indicates improved reconstruction quality, i. e. we do not just overfit the LPIPS metric through this loss term. The combination of a per-pixel and a perceptual loss is crucial to enable accurate representation of surfaces. Solely using a per-pixel loss causes noisy reconstruction results because for a given pixel the information of surrounding pixels is not considered. The perceptual loss is a feasible solution to alleviate this problem.

To achieve high image quality, using more points is desirable as it reduces the frequency and size of holes in the rasterized feature maps. Because our model's parameter count scales linearly with the number of points, this motivates the usage of regularization techniques. In contrast to previous work, we delay the learning rates for higher-frequency SH functions at the start of the optimization. This way, the model first learns the diffuse color of each point and then slowly incorporates the specular color later. We find that this regularization prevents overfitting leading to improved quality during inference. Please refer to Section 1 of our supplemental material for details on how we implement this delay. We introduce a small but effective regularization method we call point position noise (PPN), which is comparable to temporal anti-aliasing for static scenes [YLS20]. During every training iteration, we slightly offset each point from its actual position in a random direction. We found PPN to prevent overfitting of the higher-frequency SH functions. Analogous to dropout layers [SHK*14], we also randomly exclude 25% of points from the forward pass in each training iteration. Doing so encourages the model to learn what points are best suited to approximate the scene's geometry. Furthermore, it prevents artifacts during inference when points become visible which were occluded by other points for every training viewpoint and thus not optimized. In contrast to a similar approach [ZD23], our method uses no fixed subsets but a fully random selection in each training iteration while always using all points during inference. To encourage learning of spatially consistent opacities o and features ω, we further employ 3D total variation (TV) regularization [RO94]. Our implementation is similar to the one presented by Yu et al. [FKYT*22] For efficient indexing, we use the regular grid on which the points were placed during the initialization. During an iteration, we randomly select a subset of the regular grid and separately compute the loss term for point opacities and features with different weights. We observe computing a TV regularization loss on the 2D feature maps as done by previous point-based methods [ZBRH22, ZD23] to be inferior to the aforementioned 3D TV regularization. During the optimization, the model is supposed to reduce the opacity of points not located on surfaces as close to zero as possible. To encourage this, we compute a loss on the opacity values $\mathbf{o} \in \{0, 1\}^n$ of all n points in the point cloud as follows:

$$\mathcal{L}_{OPR} = \frac{1}{n} \sum_{i=1}^{n} [o_i + c \cdot (\log(o_i) + \log(1 - o_i))] \tag{5}$$

This way the model is encouraged to learn that the sum of all opacity values should be minimized while preferably using opacity values close to either zero or one [XXP*22]. We empirically determine $c = 0.1$ to be a good value and use it in all experiments. In combination, these regularization techniques enable us to confidently remove points whose opacity drops below a threshold during the optimization.

4. Experiments

We compare qualitative and quantitative performance on synthetic data as well as real-world data and present an extensive ablation study composed of multiple sub-experiments.

4.1. Setup

We detail the used datasets and baselines as well as the implementation of our method. For quantitative comparisons, we use the image quality metrics PSNR, SSIM [WBSS04], and LPIPS [ZIE*18].

Datasets. For the experiment on synthetic data, we use the Realistic Synthetic 360° dataset [MST*20]. It contains eight scenes of objects with fine geometry and high-frequency texture. For every scene, 100 training, 100 validation, and 200 test images are available, each with a resolution of 800×800 pixels. While training and validation viewpoints are randomly sampled from the hemisphere around each object, the viewpoints for testing are sampled from a continuous path around the object. Being a synthetic dataset, ground truth camera parameters and foreground segmentation masks are available.

We evaluate our method on real-world data using a small subset of the Common Objects in 3D (CO3D) dataset [RSH*21]. The dataset contains sequences captured with a smartphone by non-professionals walking on a circular path around an object of interest. The dataset authors extracted 202 frames from each video sequence and added annotations, e. g. , camera parameters and foreground segmentation masks. We select four scenes that have good image and segmentation mask quality, contain fine geometry or high-frequency texture, and have few frames where parts of the object are outside the image. Details can be found in Section 2 of our supplemental material. As proposed by the authors, we split the available images into sets of five and hold back every second set for testing. Consequently, each scene consists of 102 images for training and 100 images for testing, each with roughly a full HD resolution. We downscale the images by a factor of 0.5 to approximately match the dimensions of the synthetic scenes described above. Furthermore, we use the segmentation masks to remove the background in each scene and rescale the camera poses to lie inside a cube $[-2,2]^3$.

Baselines. We compare against a selection of computationally efficient, high quality, state-of-the-art NVS methods. Firstly, we compare against NeRF [MST*20], more specifically its JAX implementation JaxNeRF [DBS20]. Next, we select Plenoxels [FKYT*22] and TensoRF [CXG*22] to evaluate against leading approaches using voxel grids. For the synthetic experiment, we use results provided by the authors for the three aforementioned baselines. Furthermore, we compare against Instant-NGP [MESK22] where we evaluate images on white background opposed to the black background used by the authors. While our approach uses Pulsar [LZ21], we are unable to directly compare against the approach described by the authors as its implementation is not fully specified in the paper. For the sake of completeness, we provide a more extensive overview including results of multiple point-based methods in Section 2 of our supplemental material.

Implementation. We implement our method in Python using the PyTorch framework and use the Pulsar [LZ21] implementation provided by PyTorch3D [JRR*20]. For every scene, we compute the bounding box and visual hull of the object using the foreground masks and start the optimization with approximately four million points. We optimize for 20,000 iterations using the ADAM optimizer [KB15] with betas set to 0.9 and 0.95 respectively. The learning rates for the opacity and U-Net parameters are set to 1e-1 and 1e-4 and exponentially decayed to 1e-2 and 5e-5 respectively. As introduced in Section 3.5, the SH-based shading features ω are optimized on a per-degree basis. We detail this in Section 1 of our supplemental material. In every training iteration we use our model to generate the full image for a randomly selected training viewpoint and compute the loss as described in Section 3.5. We set λ_{VGG}, λ_{TV_o}, λ_{TV_ω}, and λ_{OPR} to 1e-3, 1e-3, 1e-5, and 1e-5 respectively. TV regularization of the shading features and opacities is disabled after 500 and 5,000 iterations respectively to allow modeling of fine details. After 500 iterations, we remove all points with an opacity value less than $\tau = 0.05$ and repeat this every 100 iterations thereafter. We use a single NVIDIA GeForce RTX 3090 with 24 GB VRAM for all experiments.

4.2. Results

Synthetic Scenes. We report quantitative results for the Realistic Synthetic 360° dataset in Table 1. As indicated by the PSNR metric, our method produces images with a higher per-pixel error compared to Instant-NGP and TensoRF. However, our method outperforms all baselines with respect to perceptual image metrics, especially LPIPS. Through qualitative evaluation on result images (see Figure 3), we confirm the results of the aforementioned quantitative analysis. We observe significantly improved representation of high-frequency texture and reflective surface characteristics, especially noticeable in the *Drums* and *Materials* scenes. In contrast to the smooth reflections produced by our method, JaxNeRF produces blurry reflections while Plenoxels, Instant-NGP, and TensoRF produce noisy reflections with many visible artifacts. Opposed to other methods, PlenopticPoints' U-Net is capable of 2D generalization, which is a major reason for the aforementioned improvements. In Table 1, we also include average optimization time, resulting model size, and rendering speed at inference time. While our method does not optimize as fast as TensoRF, Instant-NGP, and Plenoxels, it outperforms JaxNeRF in terms of optimization time by a large margin. During inference our method achieves frame rates of 7 FPS, which is two orders of magnitude faster than JaxNeRF and about as fast as TensoRF and Plenoxels. Of the compared methods, only Instant-NGP is able to achieve real-time rendering frame rates. Lastly, we observe our explicit, point-based representation has a similar model size as Plenoxels. The U-Net's hole-filling capabilities enables PlenopticPoints' rendering pipeline to work with a more sparse point cloud. Thus, we can match the memory footprint of Plenoxels even though we store 144 appearance features per point instead of the 27 used by a Plenoxel [FKYT*22].

Real-World Scenes. We compare real-world performance on 4 selected scenes from the CO3D dataset [RSH*21]. As we remove the background using the foreground segmentation masks provided with the scenes, each baseline's ability to deal with imperfections

Method	Realistic Synthetic 360° [MST*20] Opt. Time ↓	Model Size ↓	FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	CO3D [RSH*21] PSNR ↑	SSIM ↑	LPIPS ↓
JaxNeRF [DBS20]	>24 hrs	**14.3 MB**	0.05	31.68	0.954	0.068	32.83	0.965	0.057
Plenoxels [FKYT*22]	11.4 min	778.1 MB	15	31.70	0.958	0.050	29.12	0.944	0.093
Instant-NGP [MESK22]	**5.0 min**	66.4 MB	**>60**	32.88	**0.965**	0.046	30.16	0.949	0.110
TensoRF [CXG*22]	17.4 min	71.8 MB	1	**33.08**	0.964	0.051	31.94	0.960	0.066
Ours	2.0 hrs	862.5 MB	7	32.07	**0.965**	**0.038**	**32.85**	**0.970**	**0.035**

Table 1: *Average image quality metrics for the Realistic Synthetic 360° dataset [MST*20] and the CO3D dataset [RSH*21]. For the synthetic dataset, we also include average optimization time, resulting model size, and FPS during inference.*

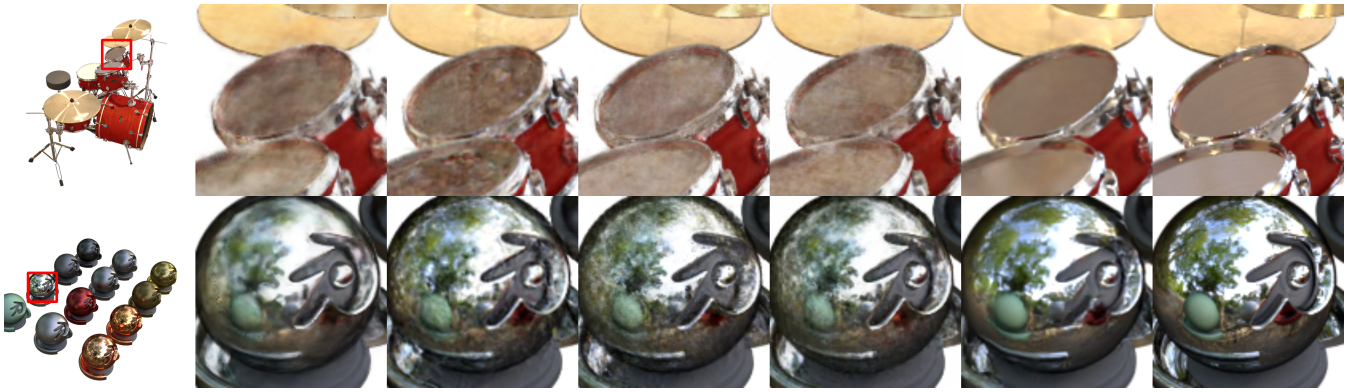

Figure 3: *Qualitative results for the Realistic Synthetic 360° dataset [MST*20]. We select two scenes and render a viewpoint from the respective test set with each method. Note the significantly improved drum heads as well as the less blurry reflections on the metal ball reconstructed by our method. Please see our supplemental material for the remaining scenes.*

within these masks is tested. This way, we analyze how applicable each method is in terms of reconstructing an object of interest from a set of casually captured images. The quantitative results (see Table 1) show that the performance of Plenoxels and Instant-NGP suffers due to the aforementioned imperfections in camera parameters and foreground segmentation masks. JaxNeRF performs comparatively well in this experiment. Our method outperforms all baselines in terms of image quality metrics. This is also visible in Figure 4, where we show images generated by each method. In comparison, our method is the only one that is able to achieve satisfactory reconstruction of the specular effects within the *Car* scene. Especially Plenoxels and TensoRF struggle to accurately represent the object's boundaries which causes severe artifacts in the *Plant* scene.

4.3. Ablation Experiments

Regularization. We identify key components of our pipeline by conducting ablation experiments (see Table 2). According to our tests, the easiest and most effective way of improving image quality is increasing the number of initial points leading to higher VRAM requirements and computation times. We present experiments with one 1 million (M), 2M, and 4M initial points (experiments A, C, and K). While even the model using 1M points generates high-quality images (see quality metrics), we maximize the used VRAM and choose 4M for our final model to demonstrate that the combination of differentiable rasterization and point-based models can be used to achieve leading image quality. Random point dropouts enable us to use more initial points as less data is used for each forward and backward pass of our model. Due to VRAM constraints, we are unable to evaluate without random dropouts when using 4M initial points. Therefore, we show results for different dropout rates when initializing with 2M points (experiments B, C, and D) and observe that using a 25% dropout rate compared 0% leads to no significant loss in image quality but significantly smaller final point clouds. This indicates doing random dropouts teaches our model to

Initial Points	Experiment	Remaining Points	PSNR↑	SSIM↑	LPIPS↓
1M	A) Fewer Points	151,980	31.18	0.958	0.048
2M	B) 50% Dropout	310,645	31.68	0.961	0.043
	C) 25% Dropout	308,561	31.70	0.962	0.043
	D) No Dropout	572,572	31.81	0.962	0.042
4M	E) No $\mathcal{L}_{L1}$	3,841,027	29.90	0.949	**0.038**
	F) No Position Noise	1,431,219	31.49	0.962	0.040
	G) No $\mathcal{L}_{TV_o}$, $\mathcal{L}_{TV_\omega}$	937,958	31.85	0.963	0.039
	H) No $\mathcal{L}_{VGG}$	1,409,952	32.02	**0.965**	0.043
	I) No $\mathcal{L}_{OPR}$	1,535,794	32.02	**0.965**	**0.038**
	J) No SH Delay	1,907,305	31.80	0.964	**0.038**
	K) Complete Model	1,381,877	**32.07**	**0.965**	**0.038**

Table 2: *Ablation study with respect to number of initial points, random point dropouts, losses, and regularization. We report average image quality metrics on the Realistic Synthetic 360° dataset [MST*20].*

Figure 4: *Qualitative results for the CO3D dataset [RSH*21]. We select two scenes and render a viewpoint from the respective test set with each method. Note the cleaner object boundaries of the plant as well as the less blurry reflections on the car. Please see our supplemental material for the remaining scenes.*

work with fewer points, a desirable property as less points lead to faster rendering, but observe no benefit of further increasing the dropout rate to 50% and therefore choose 25% as the default configuration. E) confirms $\mathcal{L}_{L1}$ is the most important loss for the optimization. F) and G) show that optimizing with point position noise and 3D TV regularization results in higher image quality. We find resulting point clouds model the object of interest more accurately when using TV regularization. In H), we observe $\mathcal{L}_{VGG}$ improves the LPIPS metric without overfitting it as the PSNR also increases. Lastly, I) shows opacity regularization reduces the number of remaining points while barely impacting image quality. To confirm the effectiveness of using customized learning rates for each SH degree, we disable it in J) and observe both reduced image quality as well as point cloud quality.

Shading Features. We determine the number of shading features attached to each point using two parameters. These are the number of channels in the rasterized feature map (CH) as well as the number of spherical harmonics (SH) functions used for the view-dependent shading. We show the influence of different values for these parameters in Table 3. For the number of feature channels, we choose relevant powers of two (8 and 16). The number of SH functions originates from the use of spherical harmonics functions up to a certain degree ℓ. Using functions up to degree ℓ means using a total amount of $(\ell+1)^2$ functions. Note that the higher the degree of a SH function, the higher the frequencies it can represent. As indicated by the results of the 8 CH experiments in Table 3, using more than the first nine spherical harmonics functions provides little benefit. This is in line with the findings of prior work, e. g. [YLT*21]. Further increasing either of the two aforementioned parameters is possible, but requires more than 24 GB of VRAM during the optimization. On top of that, the configurations with such high numbers of parameters are more likely to overfit. We therefore avoid those and choose 16 feature channels and use the first nine SH functions as they lead to the best results (see Table 3).

Highly-Reflective Scenes. Ref-NeRF addresses NeRF's limitations regarding the accurate representation of glossy surfaces by explicitly modeling spatially-varying scene properties such as surface normals [VHM*22]. Compared to our method, Ref-NeRF is significantly more expensive both in terms of optimization time (roughly 20× slower) and rendering speed during inference (roughly 100× slower). We think it is relevant to compare our method to Ref-NeRF, as we do not explicitly model properties such as surface normals, diffuse colors, and surface roughness and thus need to rely on an implicit representation for reflections. Its authors propose the synthetic Shiny Blender dataset, which consists of six highly specular scenes with simple geometry. In Table 4, we show results presented by the authors of Ref-NeRF [VHM*22] and results obtained using our method. The results show that our method performs on par with Mip-NeRF [BMT*21], the approach Ref-NeRF is built upon, but worse compared to Ref-NeRF for these highly specular scenes. Given that Ref-NeRF explicitly models scene properties that are key to compute accurate reflections, these results are to be expected. However, we think that future work could incorporate the integrated directional encoding used in Ref-NeRF [VHM*22] into the view-dependent shading of our rendering pipeline to achieve improved results on these scenes.

Configuration	PSNR↑	SSIM↑	LPIPS↓
8 CH, 4 SH	31.46	0.959	0.047
8 CH, 9 SH	31.93	0.963	0.042
8 CH, 16 SH	31.94	0.964	0.040
16 CH, 1 SH	29.80	0.947	0.066
16 CH, 4 SH	31.70	0.962	0.043
16 CH, 9 SH (default)	**32.07**	**0.965**	**0.038**

Table 3: *Average image quality metrics for the Realistic Synthetic 360° dataset [MST*20] when using different amounts of feature channels (CH) and spherical harmonics functions (SH).*

Method	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF [BMT*21]	29.76	0.942	0.092
Ref-NeRF [VHM*22]	**35.96**	**0.967**	**0.058**
Ours	29.87	0.946	0.107

Table 4: *Average image quality metrics of Mip-NeRF, Ref-NeRF, and our method on the Shiny Blender dataset [VHM*22]. Results for Mip-NeRF and Ref-NeRF taken from the Ref-NeRF publication [VHM*22].*

5. Discussion

Geometry Representation. We use a point cloud to represent a scene's geometry whereas other methods use either a large multi-layer perceptron (MLP), a voxel grid, or both. It is difficult to identify differences in quantitative and qualitative performance with respect to fine geometry for the synthetic scenes. For the real-world scenes however, our method represents fine details of the scene geometry at the object boundaries at much higher quality compared to other approaches. The results of the conducted experiments furthermore indicate that our point-based model can more accurately capture reflective surface characteristics. However, we mostly attribute this to the U-Net's ability to remove noise caused by imperfections of the geometric proxy, i. e. the point cloud. We explicitly analyze the shape of the point clouds produced by PlenopticPoints in Section 3 of our supplemental material.

Computational Requirements. While PlenopticPoints' average frame rate of 7 frames per second puts it within reach for real-time rendering applications, it is still slower compared to, e. g. , Instant-NGP. We measure a forward pass of our U-Net with an input shape of $16\times800\times800$ requires about 75 milliseconds. Pulsar [LZ21] requires between 35 and 135 milliseconds per execution, depending on the number of points. Previous work has already shown that the sphere-based scene representation of Pulsar is much slower compared to one-pixel point rasterization [RFS22]. However, the sphere-based representation allows for a better approximation of gradients during the optimization. Speeding up the rendering pipeline translates to faster inference and optimization. This is more significant compared to most previous approaches, as we use a full image during each training iteration whereas others, e. g. TensoRF and Instant-NGP, only use a small batch of rays.

Scene Editing. An advantage of our method in comparison to existing NVS approaches is the possibility for fine-grained scene editing without any changes to the implementation. The underlying point cloud can be modified freely, e. g. , by using arbitrary software capable of 3D point cloud modeling. While other models could be edited similarly, such editing would come with additional overhead. In Figure 5, we show examples of point clouds modified using Blender. Additionally, we can exclusively render specific SH frequencies, e. g. the zeroth SH degree for a view-independent color. However, we find the U-Net tries to recover view-dependent shading for some pixels as it recognizes the corresponding image region from remaining features, which limits this kind of editing.

Figure 5: *We show three examples for directly modifying the optimized point cloud. We chop off the ficus, stack the material balls, and the remove lego bulldozers's base plate.*

6. Conclusion

We presented PlenopticPoints, a point-based method for novel view synthesis given camera-calibrated input images. Instead of a requiring a complete point cloud as input, our model utilizes foreground segmentation masks to set up a regular grid of points. After optimization, a PlenopticPoints model can generate novel views of state-of-the-art quality, especially for scenes with fine geometry and specular reflections. Due to the underlying geometric proxy, i. e. the learned point cloud, the presented model supports scene-editing and integrates well with traditional rendering frameworks based on rasterization. While PlenopticPoints already produces high-quality images, some steps of its pipeline leave headroom for optimization. Future work could explore view-dependent shading techniques with better generalization capabilities as well as more efficient rasterization and hole-filling algorithms. Overall, our work shows that coarse-to-fine optimization as well as proper regularization allows point-based novel view synthesis methods to achieve highly-competitive results compared to neural field-based methods.

Acknowledgements

We would like to thank Peter Kramer for his help in preparing the supplemental video. The authors gratefully acknowledge funding by the German Science Foundation (DFG MA2555/15-1 "Immersive Digital Reality") and the L3S Research Center, Hanover, Germany.

References

[ASK*20] ALIEV K.-A., SEVASTOPOLSKY A., KOLOS M., ULYANOV D., LEMPITSKY V.: Neural point-based graphics. In *ECCV* (2020). doi:10.1007/978-3-030-58542-6_42. 2, 3, 4

[BMT*21] BARRON J. T., MILDENHALL B., TANCIK M., HEDMAN P., MARTIN-BRUALLA R., SRINIVASAN P. P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *ICCV* (2021). doi:10.1109/ICCV48922.2021.00580. 2, 7

[BMV*22] BARRON J. T., MILDENHALL B., VERBIN D., SRINIVASAN P. P., HEDMAN P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR* (2022). doi:10.1109/CVPR52688.2022.00539. 2

[CFHT23] CHEN Z., FUNKHOUSER T., HEDMAN P., TAGLIASACCHI A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR* (2023). doi:10.1109/CVPR52729.2023.01590. 2

[CXG*22] CHEN A., XU Z., GEIGER A., YU J., SU H.: TensoRF: Tensorial radiance fields. In *ECCV* (2022). doi:10.1007/978-3-031-19824-3_20. 2, 5, 6

[DBS20] DENG B., BARRON J. T., SRINIVASAN P. P.: JaxNeRF: an efficient JAX implementation of NeRF, 2020. URL: https://github.com/google-research/google-research/tree/master/jaxnerf. 5, 6

[FKYT*22] FRIDOVICH-KEIL S., YU A., TANCIK M., CHEN Q., RECHT B., KANAZAWA A.: Plenoxels: Radiance fields without neural networks. In *CVPR* (2022). doi:10.1109/CVPR52688.2022.00542. 2, 4, 5, 6

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The Lumigraph. In *CGIT* (1996). doi:10.1145/237170.237200. 2

[GKJ*21] GARBIN S. J., KOWALSKI M., JOHNSON M., SHOTTON J., VALENTIN J.: FastNeRF: High-fidelity neural rendering at 200fps. In *ICCV* (2021). doi:10.1109/ICCV48922.2021.01408. 2

[HSM*21] HEDMAN P., SRINIVASAN P. P., MILDENHALL B., BARRON J. T., DEBEVEC P.: Baking neural radiance fields for real-time view synthesis. *ICCV* (2021). doi:10.1109/ICCV48922.2021.00582. 2

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. In *ECCV* (2016). doi:10.1007/978-3-319-46475-6_43. 4

[JRR*20] JOHNSON J., RAVI N., REIZENSTEIN J., NOVOTNY D., TULSIANI S., LASSNER C., BRANSON S.: Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia* (2020). doi:10.1145/3415263.3419160. 5

[KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *ICLR* (2015). doi:10.48550/arXiv.1412.6980. 5

[KIT*21] KONDO N., IKEDA Y., TAGLIASACCHI A., MATSUO Y., OCHIAI Y., GU S. S.: Vaxnerf: Revisiting the classic for voxel-accelerated neural radiance field. doi:10.48550/arXiv.2111.13112. 2

[KKLD23] KERBL B., KOPANAS G., LEIMKÜHLER T., DRETTAKIS G.: 3d gaussian splatting for real-time radiance field rendering. *ACM TOG* (2023). doi:10.1145/3592433. 3

[KLR*22] KOPANAS G., LEIMKÜHLER T., RAINER G., JAMBON C., DRETTAKIS G.: Neural point catacaustics for novel-view synthesis of reflections. *ACM TOG* (2022). doi:10.1145/3550454.3555497. 3

[KPLD21] KOPANAS G., PHILIP J., LEIMKÜHLER T., DRETTAKIS G.: Point-based neural rendering with per-view optimization. *CGF* (2021). doi:10.1111/cgf.14339. 3

[Lau94] LAURENTINI A.: The visual hull concept for silhouette-based image understanding. *ITPIDJ* (1994). doi:10.1109/34.273735. 3

[LZ21] LASSNER C., ZOLLHOFER M.: Pulsar: Efficient sphere-based neural rendering. In *CVPR* (2021). doi:10.1109/CVPR46437.2021.00149. 3, 4, 5, 8

[MBRS*21] 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). doi:10.1109/CVPR46437.2021.00713. 2

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG* (2022). doi:10.1145/3528223.3530127. 2, 5, 6

[MHMB*22] MILDENHALL B., HEDMAN P., MARTIN-BRUALLA R., SRINIVASAN P. P., BARRON J. T.: NeRF in the dark: High dynamic range view synthesis from noisy raw images. *CVPR* (2022). doi:10.1109/CVPR52688.2022.01571. 2

[MSOC*19] MILDENHALL B., SRINIVASAN P. P., ORTIZ-CAYON R., KALANTARI N. K., RAMAMOORTHI R., NG R., KAR A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG* (2019). doi:10.1145/3306346.3322980. 2

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). doi:10.1145/3503250. 2, 3, 5, 6, 7

[OLN*22] OST J., LARADJI I., NEWELL A., BAHAT Y., HEIDE F.: Neural point light fields. *CVPR* (2022). doi:10.1109/CVPR52688.2022.01787. 3

[RALB22] RAKHIMOV R., ARDELEAN A.-T., LEMPITSKY V., BURNAEV E.: NPBG++: Accelerating neural point-based graphics. In *CVPR* (2022). doi:10.1109/CVPR52688.2022.01550. 3

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. In *MICCAI* (2015). doi:10.1007/978-3-319-24574-4_28. 2, 4

[RFS22] RÜCKERT D., FRANKE L., STAMMINGER M.: Adop: Approximate differentiable one-pixel point rendering. *ACM TOG* (2022). doi:10.1145/3528223.3530122. 2, 3, 4, 8

[RO94] RUDIN L., OSHER S.: Total variation based image restoration with free local constraints. In *ICIP* (1994). doi:10.1109/ICIP.1994.413269. 4

[RSH*21] REIZENSTEIN J., SHAPOVALOV R., HENZLER P., SBORDONE L., LABATUT P., NOVOTNY D.: Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV* (2021). doi:10.1109/ICCV48922.2021.01072. 5, 6, 7

[RSV*23] 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). doi:10.1145/3592426. 3

[SHK*14] SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R.: Dropout: a simple way to prevent neural networks from overfitting. *JMLR* (2014). doi:10.5555/2627435.2670313. 4

[SSC22] SUN C., SUN M., CHEN H.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR* (2022). doi:10.1109/CVPR52688.2022.00538. 2

[SZ15] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. In *ICLR* (2015). doi:10.48550/arXiv.1409.1556. 4

[TCY*22] TANCIK M., CASSER V., YAN X., PRADHAN S., MILDENHALL B. P., SRINIVASAN P., BARRON J. T., KRETZSCHMAR H.: Block-nerf: Scalable large scene neural view synthesis. In *CVPR* (2022). doi:10.1109/CVPR52688.2022.00807. 2

[TTM*22] TEWARI A., THIES J., MILDENHALL B., SRINIVASAN P., TRETSCHK E., YIFAN W., LASSNER C., SITZMANN V., MARTIN-BRUALLA R., LOMBARDI S., SIMON T., THEOBALT C., NIESSNER M., BARRON J. T., WETZSTEIN G., ZOLLHÖFER M., GOLYANIK V.: Advances in Neural Rendering. *EG STAR* (2022). doi:10.1111/cgf.14507. 2

[VHM*22] VERBIN D., HEDMAN P., MILDENHALL B., ZICKLER T., BARRON J. T., SRINIVASAN P. P.: Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR* (2022). doi:10.1109/CVPR52688.2022.00541. 2, 7

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: from error visibility to structural similarity. *IEEE TIP* (2004). doi:10.1109/TIP.2003.819861. 5

[XXP*22] XU Q., XU Z., PHILIP J., BI S., SHU Z., SUNKAVALLI K., NEUMANN U.: Point-nerf: Point-based neural radiance fields. In *CVPR* (2022). doi:10.1109/CVPR52688.2022.00536. 2, 4

[YLS20] YANG L., LIU S., SALVI M.: A survey of temporal antialiasing techniques. In *CGF* (2020). doi:10.1111/cgf.14018. 4

[YLT*21] YU A., LI R., TANCIK M., LI H., NG R., KANAZAWA A.: PlenOctrees for real-time rendering of neural radiance fields. In *ICCV* (2021). doi:10.1109/ICCV48922.2021.00570. 2, 7

[ZBRH22] ZHANG Q., BAEK S.-H., RUSINKIEWICZ S., HEIDE F.: Differentiable point-based radiance fields for efficient view synthesis. *SIGGRAPH Asia* (2022). doi:10.1145/3550469.3555413. 3, 4

[ZD23] ZUO Y., DENG J.: View synthesis with sculpted neural points. In *ICLR* (2023). doi:10.48550/arXiv.2205.05869. 2, 3, 4

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR* (2018). doi:10.1109/CVPR.2018.00068. 4, 5