PAPR Up-close: Close-up Neural Point Rendering without Holes

Yanshu Zhang Chirag Vashist Shichong Peng Ke Li
Simon Fraser University
{yanshu_zhang, chirag_vashist, shichong_peng, keli}@sfu.ca

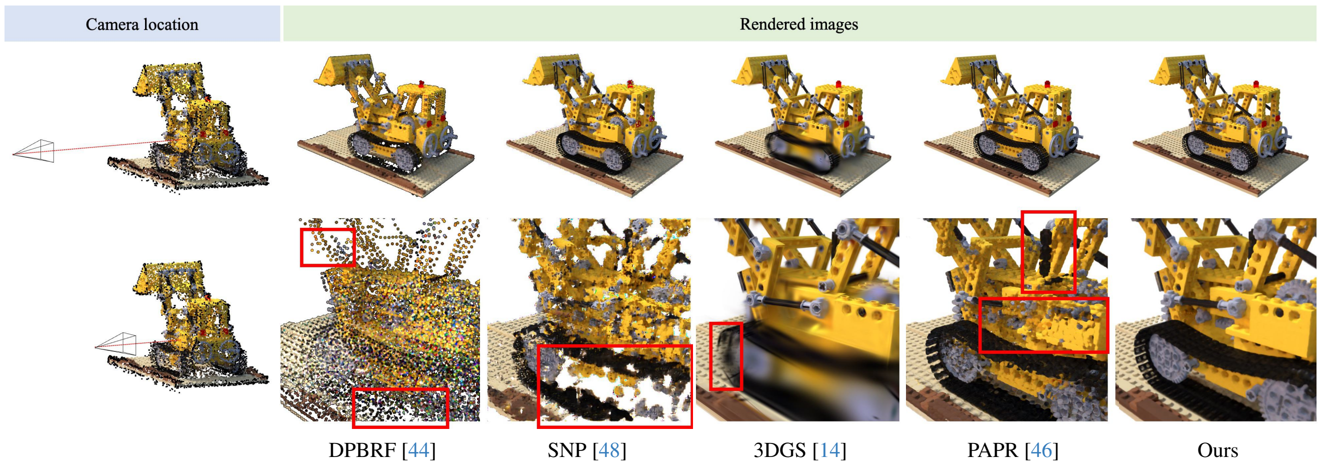


Figure 1. Recent splatting-based neural point renderers [14, 44, 48] can render promising novel view images from a parsimonious point cloud when the camera is distant from the object. However, due to the natural discreteness of the splats, these methods produce holes and/or other artifacts when we render novel views up-close. Different from splatting-based renderers, Proximity Attention Point Rendering (PAPR) [46] uses attention to predict the ray's intersection by interpolating the nearby points around a given ray. While in principle it can fill holes if the points are interpolated appropriately, PAPR still produces holes when incorrect intersections are predicted in close-up views. In this work, we propose an extension of PAPR which significantly reduces the prevalence of holes and other artifacts in close-up rendering, which outperforms recent neural point renderers.

Abstract

Point-based representations have recently gained popularity in neural rendering. While they offer many advantages, rendering them from close-up views often results in holes. In splatting-based neural point renderers, these are caused by gaps between different splats, which cause many rays to not intersect with any splat when viewed close-up. A different line of work uses attention to estimate each ray's intersection by interpolating between nearby points. Our work builds on one such method, known as Proximity Attention Point Rendering (PAPR), which learns parsimonious and geometrically accurate point representations. While in principle PAPR can fill holes by learning to interpolate between nearby points appropriately, PAPR also produces holes when rendering close-up, as the intersection point is often predicted incorrectly. We analyze this phenomenon and propose two novel solutions: a method for dynamically selecting nearby points to a ray for interpolation, and a robust attention method that better generalizes to local point configuration around unseen rays. These significantly reduce the prevalence of holes and other artifacts in close-up rendering compared to recent neural point renderers.

1. Introduction

Neural point-based representations have garnered substantial attention due to their explicit nature, ease of manipulation, and efficient handling of complex geometries [2, 14, 17, 26, 29, 44, 48], in comparison to implicit representations such as the neural radiance fields [21]. Evaluations of neural renderers are typically under constrained conditions, for example, by using a similar distance between the camera and the scene for both training and testing. However, in real-world scenarios, the camera-to-scene distance often changes significantly. Therefore, photorealistic rendering in the wild necessitates neural renderers that handle the small camera-to-scene distances inherent to close-up-views despite their absence from the training distribution.

Rendering close-up is different from zooming in: the former changes the camera position, while the latter adjusts the focal length. When the camera moves close, the angles at which rays intersect with the scene surface change, whereas they remain the same when solely adjusting the focal length. By combining these two changes, artists can create captivating effects like the dolly zoom. However, for neural renderers, rendering close-up is challenging as it not only changes the sampling rate of the rays hitting the same surface area, but also shifts the distribution of ray directions from that seen during training. While the change in sampling rate can lead to aliasing artifacts, which various approaches [3–5, 13, 42, 47] focus on, there is far less literature addressing artifacts arising from the shift in ray directions—we make this shift the focus of our paper. When a neural rendering model overfits the distribution of ray directions in the training set, it may fail to generalize to unseen rays to predict the correct intersection points between the scene surface and the rays, which results in holes or see-through artifacts.

For the splatting-based point renderers, where points are rendered with primitives known as *splats* whose extents are larger than a pixel, such as discs, ellipsoids, or Gaussian splats [14, 44], the discreteness of splats makes close-up artifacts more obvious when rendering. While the aliasing artifacts could be resolved by making the size of splats adapt to the sampling rate [20, 35, 42], there is no easy solution to fix the holes or see-through artifacts when there are gaps between splats. While the gaps may not be observable from the training ray directions, some rays that originally hit some splats may not hit any after the camera moves, thereby creating holes in the rendered image, as shown in Fig. 1 and illustrated in Fig 2a. While increasing the splat size could fill these holes, this can come at the cost of overly smoothed geometry and/or blurry texture.

The close-up artifacts in splatting-based point renderers are fundamentally caused by gaps between splats. If we had a continuous surface, we wouldn't have this issue. Attention-based point renderers aim to produce such continuous surface by interpolating between nearby points around each ray [7, 24, 46] using attention mechanisms. Because this interpolation produce a continuous surface, it can in principle avoid holes. Therefore, we build on one such method, known as Proximity Attention Point Rendering (PAPR) [46], which learns accurate and sparse point-based surface representations from scratch. It works by identifying a local neighborhood of points around each ray and then estimates the intersection of the ray with the scene surface from the neighborhood. It does so by interpolating between the points within the neighborhood, where the interpolation coefficients are given by attention weights. When the renderer is trained, the attention mechanism learns to generate correct interpolation coefficients from the local point configuration within the neighborhood. As the attention weights always sum to one, PAPR guarantees that every pixel will have points that substantially contribute to its rendering, even when the ray through the pixel is far from all points. This property is key: it enables PAPR to fill the gaps between points even with a sparse point cloud. While in principle PAPR is able to render a hole-free surface if it learns to produce the correct interpolation coefficients, as shown in Fig. 1, in practice PAPR persists in rendering holes as the camera moves close to the scene.

In this paper, we propose an extension of PAPR that renders hole-free surfaces from novel close-up views, named "PAPR Up-close". We begin by analyzing the reasons why PAPR produces holes when rendering close-up views. Our analysis reveals that this is due to incorrect predictions of the intersection points between rays and the scene for some rays. We break this phenomenon down further into two scenarios. In the first scenario, the true intersection point is outside the identified local neighborhood, making it impossible to be predicted correctly with an interpolation of points within the neighborhood. In the second scenario, the attention mechanism fails to predict the intersection point correctly from local point configurations around unseen rays, even when the true intersection lies inside the identified local neighborhood. To address the first issue, we introduce a Dynamic Point Selection (DPS) method that dynamically adapts the point selection strategy during training by prioritizing points that receive higher attention weights. To address the second issue, we enhance the robustness of the attention mechanism to variations in camera-to-object distance and shifts in the distribution of ray directions, thereby better handling unseen ray directions and distances to the camera. We demonstrate that PAPR Up-close produces substantially fewer holes and other artifacts in rendered close-up views compared to PAPR, yielding a relative improvement of 30.4%, 19.16% and 23.5% in terms of PSNR, SSIM and LPIPS respectively.

In summary, our main contributions are:

- We analyze why PAPR produces holes when rendering close-up views.
- We introduce a Dynamic Point Selection (DPS) method that adaptively adjusts the point selection strategy during training by prioritizing points with higher attention weights, which more frequently include the true intersection point within the local neighborhood and improve the accuracy of intersection predictions.
- We propose a Ray Perturbation Strategy (RPS) to enhance the robustness of the attention mechanism of PAPR to rays originating from unseen directions and distances.

2. Related Work

In this paper, we focus on close-up rendering of sparse point-based representations, without supervision from close-up captured views. Point-based rendering has long been an area of interest within computer graphics research due to its efficiency in rendering disconnected and unstructured geometric samples [11, 19]. However, it is prone to holes, aliasing artifacts, and discontinuities. Pioneering work on high-quality point-based rendering tackled these challenges by *splatting* point primitives with an extent larger than a pixel, such as circular or elliptical discs, ellipsoids, or surfels [1, 6, 16, 25, 27, 49]. While recent work extended these conventional methods to exploit modern GPU architectures [32, 33], our discussion focuses on point-based neural rendering techniques, which have been of recent interest.

2.1. Splatting-based Neural Point Rendering

Point-based neural rendering approaches typically incorporate neural networks or learnable components into a differentiable rendering pipeline. Splatting-based neural point renderers operate by projecting points onto the image plane as *splats*, which are then blended together to create a continuous image or feature map. The points could be augmented with learnable features that capture their opacity and radiance [14, 17, 26, 29, 44], and the rendering pipeline is often end-to-end differentiable with soft rasterizers [38, 39, 44]. Some works further refine the feature map using a convolutional neural network (CNN) to produce the final image [2, 26, 29, 48]. In splatting-based neural renderers, a splat's contribution to a pixel's value is a function of its spatial distance to the ray. Hard-boundary splatting techniques ignore splats whose distance to the ray is larger than the splat's radius [29, 38, 39]. In soft-boundary splatting, the contribution of a splat typically decreases rapidly as the distance to the ray increases, for example, the Gaussian splats [14, 44]. Consequently, this spatial discretization yields holes in the rendered images when many rays do not intersect with any splat. As these methods require a large number of points and relatively small splats to accurately model scene geometry and textures, this issue is especially serious with sparse point clouds. Close-up shots compound this issue, as many rays can be too far away from any splat to receive contributions from them, resulting in larger holes.

2.2. Attention-based Neural Point Rendering

Instead of explicitly modeling local geometry at each point with splat representations and determining ray intersections through rasterization, another flavor of neural point renderers employs an attention mechanism [37] to infer where the ray intersects the scene [7, 46]. Utilizing the fact that the intersection point between the ray and surface lies along the ray, these methods estimate the intersection by interpolating between the points in a local neighborhood around the ray using attention weights. Each point is represented by its position and additional ray-dependent point features, which serve as keys in the attention layer. Given a ray as a query, the attention layer learns to aggregate the point features to produce a feature map. A rendering model—usually an MLP or UNet [28]—is then used to generate the final image from the feature map. In contrast to splatting-based renderers, these methods ensure each pixel receives substantial contribution from some points, regardless of the distance between the pixel and the points, as the attention weights across points within the local neighborhood always sum to one. In principle this property makes it possible to render a sparse point-based representation without introducing holes. Among attention-based point renderers, Neural Point Light Field [24] focuses on rendering large scenes from captured point clouds but has limitations in synthesizing novel views of single objects [46]. Pointersect [11] aims at explicitly predicting surface intersections from given point clouds, but requires pre-training on ground truth meshes. Proximity Attention Point Rendering (PAPR) [46] is designed to learn a sparse and geometrically accurate point cloud from scratch. While it can generate high-fidelity renderings from novel views at similar distances as the captured views, PAPR produces holes in close-up renderings, as shown in Fig. 1. In this paper, we aim to improve the quality of its rendering of sparse point-based representations from novel close-up views that are much closer to the scene than the captured training views.

2.3. Different Artifacts of Close-up Rendering

Novel view synthesis is the process of generating new images from views that differ from those of the original captures [10, 18]. When we synthesize close-up views by moving cameras closer to objects, we observe two kinds of artifacts: first, the aliasing artifacts, which occur when the sampling rate on the image plane differs from the rate during training, and artifacts from the shift in ray direction distribution, which happens when a model overfits to the training ray directions and fails to generalize to ray directions that are out-of-distribution. As a fundamental issue in rendering algorithms of computer graphics, aliasing happens when a sample rate is below the Nyquist rate [23, 34]. Reducing aliasing artifacts can be done by supersampling [8] or pre-filtering [9, 12, 22, 36, 40, 50]. As supersampling is expensive in the neural rendering as its runtime generally scales linearly with the supersampling rate [3], recent approaches [3–5, 13, 42, 47] addressed the aliasing issue by integrating prefiltering-based techniques. In our method, instead of combating the aliasing artifact, we focus on the artifacts from the shift of ray direction distribution which manifest as holes when rendering close-up.

3. Method

3.1. Proximity Attention Point Rendering

Given multi-view RGB images along with their corresponding camera parameters, PAPR learns a point-based 3D scene representation $\mathcal{P}$ consisting of N points. Given a ray $\mathbf{r}_j = \mathbf{o}_j + m\mathbf{d}_j$ for $m \geq 0$, PAPR identifies a local neighborhood around the ray by selecting the K points $\mathcal{P}_j \subseteq \mathcal{P}$ that are nearest to the ray. It then predicts a feature vector $\mathbf{f}_j$ for the ray by aggregating ray-dependent features $\mathbf{v}_{ij}$ associated with the points in $\mathcal{P}_j$ using per-point weights $\{w_{ij}\}_{i=1}^{K}$ produced from an attention mechanism, by computing $\mathbf{f}_j = \sum_{i=1}^{K} w_{ij}\mathbf{v}_{ij}$. The feature vectors for all rays are then passed through an UNet [28] to produce the rendered image $\hat{\mathbf{I}}$. The model is learned end-to-end by minimizing a weighted combination of mean squared error (MSE) and the LPIPS metric [45] between the rendered image $\hat{\mathbf{I}}$ and the ground truth $\mathbf{I}_{\text{gt}}$.

$$\mathcal{L}_C(\hat{\mathbf{I}}, \mathbf{I}_{\text{gt}}) = \text{MSE}(\hat{\mathbf{I}}, \mathbf{I}_{\text{gt}}) + \lambda \cdot \text{LPIPS}(\hat{\mathbf{I}}, \mathbf{I}_{\text{gt}}) \tag{1}$$

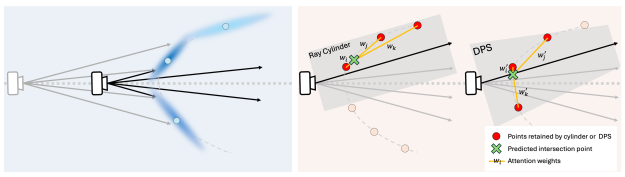
We now describe the attention mechanism in greater detail. The points $\{\mathbf{p}_i\}_{i=1}^{K}$ in the local neighborhood $\mathcal{P}_j$ for a ray $\mathbf{v}_j$ are represented by their positions $\mathbf{p}_i$ along with two ray-dependent point features, $\mathbf{s}_{ij}$ and $\mathbf{t}_{ij}$, which are computed as follows:

$$\mathbf{p}'_{ij} = \mathbf{o}_j + \langle \mathbf{p}_i - \mathbf{o}_j, \mathbf{d}_j \rangle \cdot \mathbf{d}_j, \tag{2}$$

$$\mathbf{s}_{ij} = \mathbf{p}'_{ij} - \mathbf{o}_j, \ \mathbf{t}_{ij} = \mathbf{p}_i - \mathbf{p}'_{ij}, \tag{3}$$

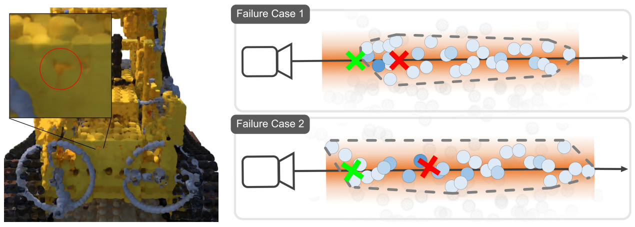
where $\mathbf{p}'_{ij}$ is the projection of point position $\mathbf{p}_i$ on ray $\mathbf{r}_j$. These features effectively capture the spatial point configuration of the local neighborhood around the ray by incorporating the features from all K points around the ray.

To find the attention weight w_{ij} for the i^{th} point in the local neighborhood $\mathcal{P}_j$, PAPR uses an embedded ray direction $\mathbf{q}_j = f_{\theta_Q}(\gamma(\mathbf{d}_j))$ as the query, and an embedded concatenation of the ray-dependent point features and point positions $\mathbf{k}_{ij} = f_{\theta_K}([\gamma(\mathbf{s}_{i,j}), \gamma(\mathbf{t}_{i,j}), \gamma(\mathbf{p}_i)])$ as the key. The function γ applies a positional encoding to its input, and f_{θ_Q} and f_{θ_K} are two embedding MLPs. We use the scaled dot-product attention [37] to compute $w_{ij} = \text{softmax}(\frac{\langle \mathbf{q}_j, \mathbf{k}_{ij} \rangle}{\sqrt{d_{\mathbf{k}}}})$, where $d_{\mathbf{k}}$ is the dimension of $\mathbf{k}_{ij}$. Although PAPR does not require explicitly predicting the ray intersections $\mathbf{x}_j$ for each ray, $\mathbf{x}_j$ can be calculated by interpolating between the positions of points in $\mathcal{P}_j$ as $\mathbf{x}_j = \sum_{i=1}^{K} w_{ij}\mathbf{p}_j$, which is within the convex hull of the points in $\mathcal{P}_j$. As shown by the authors [46], the predicted intersection points generally lie correctly on the scene surface, even though there is no explicit supervision on $\mathbf{x}_j$. Since the intersection point cannot be predicted correctly without estimating the local geometry, it is important to have an appropriate set of points in $\mathcal{P}_j$ to estimate the local geometry. To build the local neighborhood $\mathcal{P}_j$ around each ray, PAPR selects K nearest points


Ray Cylinder
DPS
Points retained by cylinder or DPS
Predicted intersection point
Attention weights

(a) Splatting-based renderers (b) Attention-based renderers

Figure 2. Illustration of splatting-based and attention-based renderers when rendering the same surface segment for close-up views. In Fig. a the center rays no longer hit any splats after the camera is moved closer. The left of Fig. b shows how PAPR interpolates points inside a *cylinder* around a ray, where points around the true intersection may not be retained to make the right prediction. On the right it shows how our method learns to retain retain all point around the true intersection with the dynamic point selection (DPS).



(a) (b) Failure Cases

Figure 3. Illustration of two causes of holes in PAPR's close-up rendering. Fig. a shows an example where the rays crossing the hole predict incorrect intersections. Fig. b visualizes two cases where PAPR fails to predict true intersection points. Point colors represent attention weights (darker means higher weight), dotted lines outline the convex hull formed by these points, green crosses mark true intersections, and red crosses indicate predicted intersections via attention-weighted interpolation of point positions.

solely based on $t_{\cdot j} = \|\mathbf{t}_{\cdot j}\|$, which is the orthogonal distance from each point to the ray.

3.2. Causes for Holes in PAPR

As shown in Fig. 1, PAPR struggles with rendering hole-free images when the camera moves close-up to the scene. In this section, we analyze the causes of holes appearing in images rendered from close-up views using PAPR. It's important to note that PAPR always predicts an intersection point for any ray, regardless of the distances between the ray and the points in the point cloud. Thus the existence of the holes is not because those rays do not intersect with the scene, as is often the case when holes appear in splatting-based methods. Instead, the issue arises because the intersection points identified by the rays are incorrect.

As shown in Fig. 3b, there are two cases when the issue happens. In the first case, the correct intersection point (denoted by the green cross) falls outside the convex hull formed by the points in the local neighborhood. As a result, there is no way for the intersection point to be predicted correctly by interpolating between points inside the neighborhood. In the second case, PAPR fails to learn attention weights that give the correct intersection, even when the correct intersection point is inside the convex hull of the points in the local neighborhood. This is because the attention mechanism fails to generalize to the point configuration within the local neighborhood around an unseen ray.

Holes become more apparent when rendering close-up views. As the camera moves close-up, the local point configurations around adjacent rays become highly correlated, since their corresponding local neighborhoods become very similar to each other. As a result, rays with incorrect intersections tend to be concentrated around some locations, thereby creating large holes. To tackle the first failure case, we propose a Dynamic Point Selection (DPS) method to select points that form the local neighborhood, as detailed in Sec. 3.3. To tackle the second failure case, we enhance the attention mechanism's robustness to local point configurations around unseen rays, as discussed in Sec. 3.4.

3.3. Dynamic Point Selection

PAPR selects the neighborhood points around a ray solely based on the orthogonal distance from a point to the ray, and such a selection strategy remains static during training. In the ideal scenario where the true intersection point of each ray is known in advance, we could select the minimal set of surrounding points that form a convex hull enclosing this intersection point. This subset effectively defines the ray's local neighborhood, allowing accurate intersection predictions with minimal computational cost. In the real world scenario, we do not know the ground truth intersections beforehand. However, as the model iteratively improves during training, the predictions of these intersection points become more accurate. Furthermore, the attention mechanism, which assigns interpolation weights to the selected points, learns to identify the most relevant points surrounding the true intersection. By giving higher weights to these relevant points and lower weights to less important ones in each ray's local neighborhood, the attention effectively narrows down the set of points crucial for accurate prediction. This insight motivates us to design a point selection method that adapts during training, to favor points that are more relevant to the true intersection. We refer to this approach as *Dynamic Point Selection (DPS)*.

At each iteration, DPS selects the points to form the local neighborhood around a ray $\mathbf{r}_j$ based on a weighted sum of two features, the orthogonal distance from a point i to the ray, denoted as $t_{ij} = \|\mathbf{t}_{ij}\|$, and the distance from a point $\mathbf{p}_i$ to the camera origin along the ray, denoted as $s_{ij} = \|\mathbf{s}_{ij}\|$:

$$d_{ij} = \alpha t_{ij} + \beta s_{ij}, \tag{4}$$

where α and β are learnable parameters. For each ray, the points with the smallest d_{ij} values are selected.

During training, α and β are optimized to ensure that the selected points align closely with those suggested by the attention mechanism. We achieve this by minimizing a binary cross-entropy (BCE) loss that measures the divergence between the points selected by DPS and those suggested by the attention mechanism. Specifically, given the attention weights for the K points in the local neighborhood around a ray $\mathbf{r}_j$, we classify the points with attention weights below the q^{th} 10-quantiles of the K weights as negative samples, and the remaining points as positive samples, where $q \in \{1, 2, \ldots, 10\}$. Each of the K point is assigned a label:

$$y_{ij} = \begin{cases} 1 & \text{if point } i \text{ a positive sample of } \mathbf{r}_j, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The BCE loss over a batch of M rays is given by:

$$\begin{aligned} \mathcal{L}_{BCE} = -\frac{1}{MK}\sum_{j=1}^{M}\sum_{i=1}^{K}&\rho_{ij}[y_{ij}\ln(1-\sigma(d_{ij})) \\ &+ (1-y_{ij})\ln\sigma(d_{ij})], \end{aligned} \tag{6}$$

where σ is the sigmoid function, and ρ_{ij} is a scaling factor that balances the positive and negative samples.

$$\rho_{ij} = \begin{cases} \frac{5}{(10-q)} & \text{if point } i \text{ a positive sample of } \mathbf{r}_j, \\ \frac{5}{q} & \text{otherwise} \end{cases} \tag{7}$$

The loss $\mathcal{L}_{BCE}$ is minimized when $\sigma(d_{ij})$ approaches 0 for positive samples and 1 for negative samples. By selecting the points with the smallest d values, DPS effectively prioritizes the more relevant points for accurate intersections according to the attention mechanism, thus refining the local neighborhood around each ray. This dynamic selection process continues throughout training, allowing the model to adjust the neighborhood points based on learned features, rather than relying on a fixed selection strategy. By the end of training, α and β have been adjusted to effectively capture the underlying patterns of point relevance in various spatial configurations observed during training. At test time, the optimized α and β are used to select points for unseen rays. As shown in Sec. 4.5, DPS enhances the hole-filling ability of the model when rendering close-up views.

3.4. Robust Attention

To tackle the second issue, as illustrated in Fig. 3b, where the attention mechanism fails to generalize to the local point configurations around unseen rays and consequently predicts incorrect intersection points, we improve the attention mechanism's robustness to these unseen configurations.

As discussed in Sec. 3.1, PAPR uses a vector $\mathbf{s}_{ij}$ as a ray-dependent point feature for the points within the local neighborhood to capture the local point configuration. This vector is defined from the camera center $\mathbf{o}_j$ to the point projection $\mathbf{p}'_{ij}$ on the view direction $\mathbf{d}_j$. As the camera moves closer to the object, the scale of $\mathbf{s}_{ij}$ changes significantly, resulting in local point configurations that are highly out-of-distribution, and the attention mechanism may fail to generalize across this variation. Therefore, we normalize $\mathbf{s}_{ij}$ by subtracting the vector with the $\mathbf{s}_{ij}$ that has the smallest length among the K vectors. This normalization ensures that the vector's scale remains consistent, thereby reducing the sensitivity of the attention to the distance to the camera.

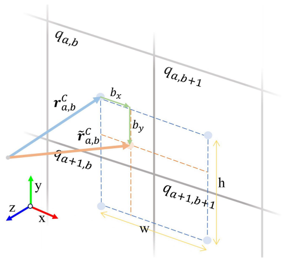


Figure 4. Illustration of the ray perturbation method described in Sec. 3.4 in the camera coordinate.

To further enhance the robustness of the attention mechanism to unseen local point configurations, we introduce a Ray Perturbation Strategy (RPS), by adding noise to the ray directions during training. The intuition is when we move the camera closer, the change of local point configurations is caused by the change of the ray directions, if $\mathbf{s}_{ij}$ has been normalized. By introducing augmented ray directions, we create a more diverse range of ray-dependent local point configurations during training. This allows the attention to better generalize to previously unseen point configurations during testing. As shown in Fig. 4, we denote $\mathbf{r}_{a,b}$ as the ray shooting at the center of pixel $q_{a,b}$ on the image plane, where $a \in \{1 \dots H\}$ and $b \in \{1 \dots W\}$, indexes from top to bottom and left to right. Specifically, for a given ray $\mathbf{r}_{a,b}$ in the world coordinate during training, we first transform it into the camera coordinate by:

$$\mathbf{r}^C_{a,b} = \mathbf{P}\mathbf{r}_{a,b}, \tag{8}$$

where $\mathbf{P}$ is the world-to-camera transformation matrix of this view. $\mathbf{r}^C_{a,b}$ is the coordinate of the pixel $q_{a,b}$ in the camera coordinate, corresponding to a ground truth RGB value $\mathbf{c}_{a,b}$. We perturb ray $\mathbf{r}^C_{a,b}$ by applying a translation:

$$\tilde{\mathbf{r}}^C_{a,b} = \mathbf{r}^C_{a,b} + \begin{bmatrix} b_x \\ b_y \\ 0 \end{bmatrix}, \tag{9}$$

where b_x and b_y are two random numbers from a normal distribution $\mathcal{N}(0, \sigma^2)$, clamped by the size of a pixel to make sure the perturbed ray remains within its pixel:

$$b_x \sim \min\left(\max\left(\mathcal{N}\left(0, \sigma^2\right), -\frac{w}{2}\right), \frac{w}{2}\right), \tag{10}$$

$$b_y \sim \min\left(\max\left(\mathcal{N}\left(0, \sigma^2\right), -\frac{h}{2}\right), \frac{h}{2}\right), \tag{11}$$

where w and h are the width and height of a pixel on the image plane in the camera coordinate. In practice, we choose $\sigma^2 = k\frac{(w+h)}{2}$ to be adaptive to the pixel size, where k is a hyperparameter that controls the magnitude of the perturbation. The RGB label $\tilde{\mathbf{c}}_{a,b}$ of ray $\tilde{\mathbf{r}}^C_{a,b}$ is then the bilinear interpolation of the colors of the nearby pixels, based on the coordinates of $\tilde{\mathbf{r}}^C_{a,b}$. For example, when $b_x > 0$ and $b_y < 0$,

$$\tilde{\mathbf{c}}_{a,b} = \frac{1}{wh}\begin{bmatrix} w-|b_x| & |b_x| \end{bmatrix}\begin{bmatrix} \mathbf{c}_{a+1,b} & \mathbf{c}_{a+1,b+1} \\ \mathbf{c}_{a,b} & \mathbf{c}_{a,b+1} \end{bmatrix}\begin{bmatrix} |b_y| \\ h-|b_y| \end{bmatrix}. \tag{12}$$

We then get $\tilde{\mathbf{r}}_{a,b} = \mathbf{P}^{-1}\tilde{\mathbf{r}}^C_{a,b}$ as the perturbed ray in the world coordinate, paired with $\tilde{\mathbf{c}}_{a,b}$ for training. We only perturb the rays that are not shooting at the pixels on the boundaries of an image. When rendering unseen images, we predict the color for each pixel by shooting only a single ray through the pixel center. Results presented in Sec. 4.5 demonstrate that our proposed robust attention mechanism significantly improves the model's ability to fill holes during the rendering of close-up views.

3.5. Training Details

In Dynamic Point Selection we initialize α and β in Eq. (4) to be 1 and 0. We fix α during training as it makes the learning process more stable. We use the second 10-quantile value for the binary partition mentioned in Section 3.3 during training. In the robust attention we add an $\varepsilon = 1\text{e}{-6}$ to the normalized $\mathbf{s}_{ij}$ to avoid gradient vanishing. For the ray perturbation strategy, we choose 1 or 5 as the value of k mentioned in Section 3.4. We train our model end to end using a weighted combination of the multi-view image loss $\mathcal{L}_C$ and $\mathcal{L}_{BCE}$,

$$\mathcal{L} = \mathcal{L}_C + \gamma\mathcal{L}_{BCE}, \tag{13}$$

where we choose $\gamma = 0.01$ for all experiments.

4. Experiments

4.1. Close-up Synthetic and Real Datasets

We evaluate our model on a new dataset that contains close-up views rendered from the synthetic scenes in NeRF-synthetic dataset [21]. For each training view in the NeRF-synthetic dataset, we translate the camera center along the optical axis to get the close-up camera and rendering in our new dataset, resulting in around 3× closer camera views. In order to demonstrate the effectiveness of the proposed method in real-world scenarios, we capture two real-world scenes, Bowser and Cup, from views surrounding the object at two distances. We use COLMAP [30, 31] to reconstruct the camera poses. Please refer to the supplementary materials for data generation details and images sampled from the close-up datasets. In all our experiments, we train our model and baselines on the training set with distant views and test on the close-up views.

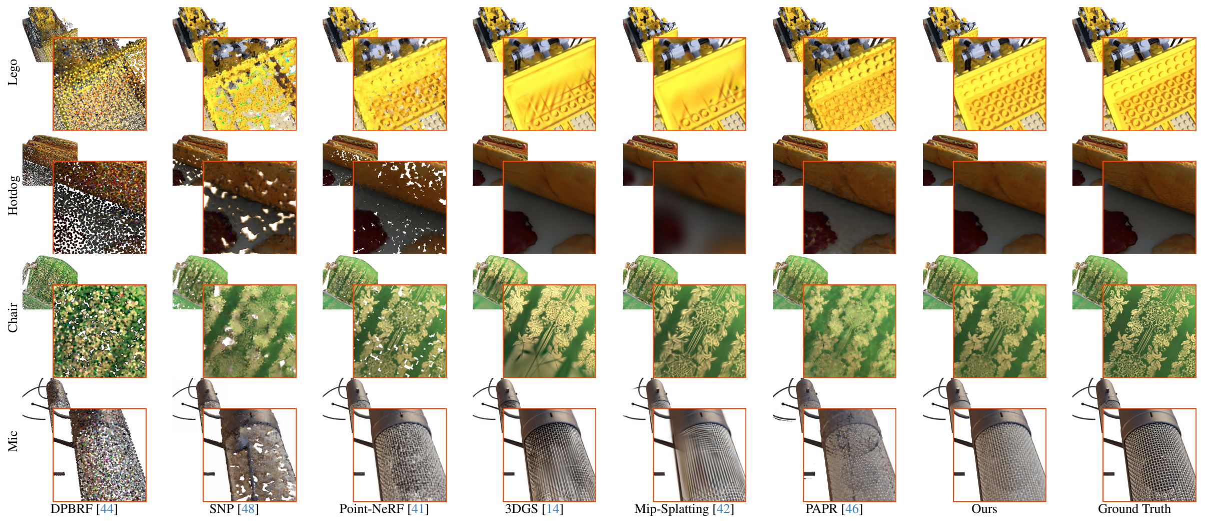


Figure 5. Qualitative comparison of novel view synthesis on the close-up synthetic dataset.

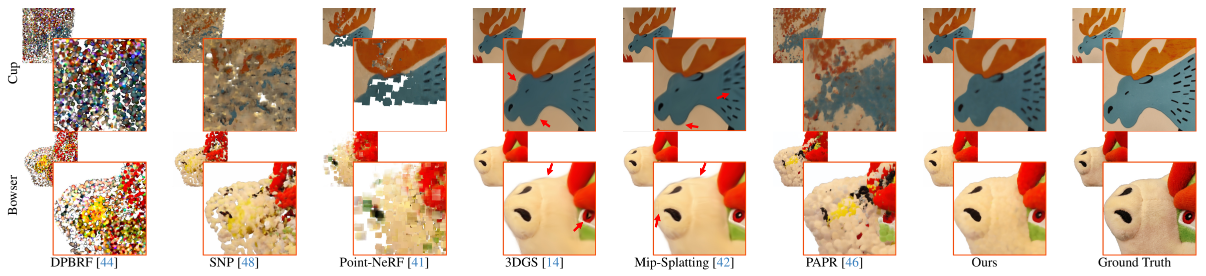


Figure 6. Qualitative comparison of novel view synthesis on the close-up real dataset.

4.2. Baselines

We compare our method with recent point-based neural renderers. Point-NeRF [41] renders a volumetric featured point cloud with volumetric ray marching. SNP [48] improves the rendering robustness to the holes in a featured point cloud with a point sculpting strategy, and renders the rasterized feature map with a UNet to further enhance the hole-filling capability. DPBRF [44] renders a point cloud via differentiable rasterization, where each splat is a 2D Gaussian on the image plane. 3D Gaussian Splatting [14] treats each point as a 3D Gaussian, rendered with a differentiable rasterizer by splatting the 3D Gaussians on the image plane. Mip-Splatting [42] enhances 3DGS by introducing a 3D smoothing filter and a 2D Mip filter, effectively eliminating aliasing artifacts. In contrast to the volumetric rendering and splatting-based rasterization methods, PAPR [46] learns and renders a surface point cloud using the attention mechanism, as described in Sec. 3.1. In all our experiments we evaluate the performance of these baselines and our method with a sparse point cloud of 30,000 points initialized using the methods proposed by respective authors.

4.3. Qualitative Results

Fig. 5 and 6 shows a qualitative comparison between our method, PAPR Up-close, and the baselines on the close-up synthetic and real datasets. Our method produces images with hole-free and smoother surfaces compared to the baselines, without losing the high-frequency details. Notably, our method captures fine texture details on the bulldozer's shovel, the details textures on the hotdog and chair, and the mesh on the mic. In contrast, the baselines either fail to render hole-free surfaces, or introduce artifacts. The comparison of our method and PAPR in Fig. 7 further shows that our method maintains high-quality renderings of distant views while significantly improving the model's hole-filling ability for the close-up renderings.

	Close-up Synthetic Dataset			Close-up Real Dataset		
	PSNR ↑	*SSIM* ↑	*LPIPS* ↓	*PSNR* ↑	*SSIM* ↑	*LPIPS* ↓
DPBRF [44]	13.06	0.352	0.525	7.301	0.076	0.779
SNP [48]	14.41	0.573	0.451	13.49	0.599	0.608
Point-NeRF [41]	19.48	0.746	0.396	13.90	0.706	0.551
3DGS [14]	21.45	0.763	0.291	18.15	0.782	0.423
Mip-Splatting [42]	22.96	0.786	0.286	18.97	0.794	0.404
PAPR [46]	18.55	0.720	0.357	16.41	0.728	0.499
Ours	**24.18**	**0.858**	**0.273**	**19.20**	**0.825**	**0.400**

Table 1. Comparison of image quality metrics (PSNR, SSIM and LPIPS [45]) on both the close-up synthetic and real datasets.

4.4. Quantitative Results

Table 1 shows the average image quality metric scores on the close-up synthetic and real datasets. Our method consis-

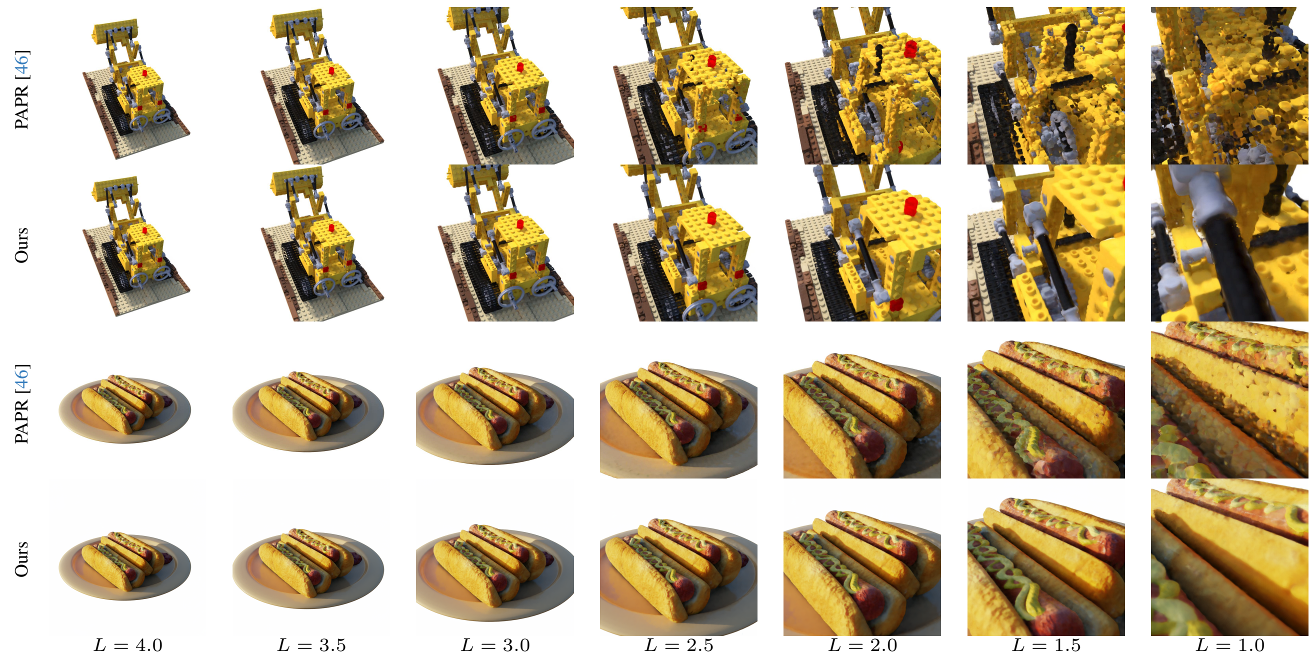


Figure 7. Qualitative comparison of our method and PAPR [46] on Lego and Hotdog scenes. The images are rendered from the cameras located at different distances L from the object along the same view direction.

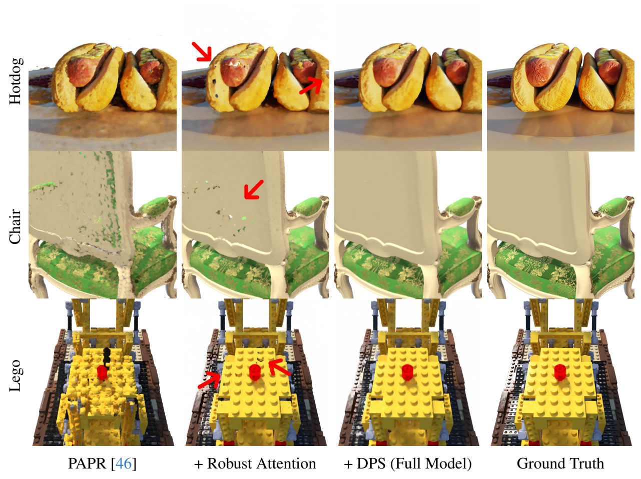


Figure 8. Ablation study on proposed components in Sec. 3.

tently outperforms the baselines across all metrics on both datasets. These results demonstrate significant improvement in the hole-filling ability of unseen close-up views.

4.5. Ablation Study

We evaluate the effectiveness of each proposed component in Fig. 8. Starting from the vanilla PAPR [46], we first add the robust attention proposed in Sec. 3.4, which significantly improves the model's hole-filling ability. We then add the Dynamic Point Selection selection method proposed in Sec. 3.3 and get our full model. The result shows that the Dynamic Point Selection selection method enables our model to further fill the remaining holes and refines the surface geometry in the rendered image.

5. Discussion and conclusion

Limitations As shown in Fig. 7, there is a slight quality degradation in the high-frequency details, which is a side effect of the ray perturbation augmentation. The amount of ray perturbation seems to introduce a tradeoff between the robustness of the attention mechanism and sharpness in the high-frequency texture. Incorporating a frequency regularizer, such as the one proposed in[43], could help mitigate this degradation. Our method also inherits the assumptions PAPR of a near-constant background color and object-centric scenes. We leave the exploration of possible solutions to the limitations for future work.

Conclusion In this paper, we extend Proximity Attention Point Rendering (PAPR) to address the challenge of rendering sparse point clouds from close-up views without holes. PAPR produces holes when rendering close-up views – we analyzed the cause and found it is due to some rays' intersection points being predicted incorrectly. To address this issue, we introduced a Dynamic Point Selection (DPS) that effectively leverages the point selection patterns learned from the attention mechanism to prioritize the most relevant points for accurate intersection prediction, and a Ray Perturbation Strategy (RPS) to improve the attention mechanism's robustness. Our experiments demonstrated a significant reduction in the prevalence of holes and other artifacts from close-up views compared to PAPR, and the results also outperform those of other recent neural point renderers.

References

[1] Marc Alexa, Markus H. Gross, Mark Pauly, Hanspeter Pfister, Marc Stamminger, and Matthias Zwicker. Point-based computer graphics. In *International Conference on Computer Graphics and Interactive Techniques*, 2004. 3

[2] Kara-Ali Aliev, Dmitry Ulyanov, and Victor S. Lempitsky. Neural point-based graphics. In *European Conference on Computer Vision*, 2019. 1, 3

[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. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, 2021. 2, 3

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ArXiv*, abs/2304.06706, 2023. 2, 3

[6] Mario Botsch, Alexander Sorkine-Hornung, Matthias Zwicker, and Leif P. Kobbelt. High-quality surface splatting on today's gpus. *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 17–141, 2005. 3

[7] Jen-Hao Rick Chang, Wei-Yu Chen, Anurag Ranjan, Kwang Moo Yi, and Oncel Tuzel. Pointersect: Neural rendering with cloud-ray intersection. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8359–8369, 2023. 2, 3

[8] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5:51–72, 1988. 3

[9] Franklin C. Crow. Summed-area tables for texture mapping. *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 1984. 3

[10] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 3

[11] Jeffrey P Grossman and William J Dally. Point sample rendering. In *Rendering Techniques' 98: Proceedings of the Eurographics Workshop in Vienna, Austria, June 29—July 1, 1998 9*, pages 181–192. Springer, 1998. 3

[12] Paul S. Heckbert. Fundamentals of texture mapping and image warping. 1989. 3

[13] Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19717–19726, 2023. 2, 3

[14] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42:1 – 14, 2023. 1, 2, 3, 7

[15] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 1, 2

[16] Leif P. Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Comput. Graph.*, 28:801–814, 2004. 3

[17] Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1440–1449, 2021. 1, 3

[18] Marc Levoy and Pat Hanrahan. Light field rendering. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 3

[19] Marc Levoy and Turner Whitted. The use of points as a display primitive. 2000. 3

[20] Zhihao Liang, Qi Zhang, Wenbo Hu, Ying Feng, Lei Zhu, and Kui Jia. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. *arXiv preprint arXiv:2403.11056*, 2024. 2

[21] 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. *Commun. ACM*, 65:99–106, 2020. 1, 6, 5

[22] Klaus Mueller, Student Member, Torsten Möller, J. Edward Swan, Roger Crawfis, Naeem Shareef, and Roni Yagel. Splatting errors and antialiasing. *IEEE Trans. Vis. Comput. Graph.*, 4:178–191, 1998. 3

[23] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47:617–644, 1928. 3

[24] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18419–18429, 2022. 2, 3

[25] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus H. Gross. Surfels: surface elements as rendering primitives. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000. 3

[26] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor S. Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15948–15958, 2022. 1, 3

[27] Liu Ren, Hanspeter Pfister, and Matthias Zwicker. Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering. *Computer Graphics Forum*, 21, 2002. 3

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015. 3, 4

[29] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Trans. Graph.*, 41:99:1–99:14, 2021. 1, 3

[30] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6, 1

[31] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 6, 1

[32] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. Rendering point clouds with compute shaders and vertex order optimization. *Computer Graphics Forum*, 40, 2021. 3

[33] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. Software rasterization of 2 billion points in real time. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5:1 – 17, 2022. 3

[34] Claude E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37:10–21, 1949. 3

[35] Xiaowei Song, Jv Zheng, Shiran Yuan, Huan-ang Gao, Jingwei Zhao, Xiang He, Weihao Gu, and Hao Zhao. Sags: Scale-adaptive gaussian splatting for training-free anti-aliasing. *arXiv preprint arXiv:2403.19615*, 2024. 2

[36] J. Edward Swan, Klaus Mueller, Torsten Möller, Naeem Shareef, Roger Crawfis, and Roni Yagel. An anti-aliasing technique for splatting. *Proceedings. Visualization '97 (Cat. No. 97CB36155)*, pages 197–204, 1997. 3

[37] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. 3, 4

[38] Yifan Wang, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38:1 – 14, 2019. 3

[39] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7465–7475, 2019. 3

[40] Lance Williams. Pyramidal parametrics. *Seminal graphics: pioneering efforts that shaped the field*, 1983. 3

[41] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5428–5438, 2022. 7, 1, 2

[42] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *arXiv:2311.16493*, 2023. 2, 3, 7

[43] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric P. Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21424–21433, 2024. 8

[44] Qian Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. *SIGGRAPH Asia 2022 Conference Papers*, 2022. 1, 2, 3, 7

[45] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 4, 7

[46] Yanshu Zhang, Shichong Peng, Seyed Alireza Moazenipourasil, and Ke Li. PAPR: Proximity attention point rendering. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 1, 2, 3, 4, 7, 8

[47] Yi Zhuang, Qi Zhang, Yingfa Feng, Hao Zhu, Yao Yao, Xiaoyu Li, Yan-Pei Cao, Ying Shan, and Xun Cao. Anti-aliased neural implicit surfaces with encoding level of detail. *SIGGRAPH Asia 2023 Conference Papers*, 2023. 2, 3

[48] Yiming Zuo and Jia Deng. View synthesis with sculpted neural points. *ArXiv*, abs/2205.05869, 2022. 1, 3, 7, 2

[49] Matthias Zwicker, Hans Rüdiger Pfister, Jeroen van Baar, and Markus H. Gross. Surface splatting. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001. 3

[50] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus H. Gross. Ewa splatting. *IEEE Trans. Vis. Comput. Graph.*, 8:223–238, 2002. 3