Differentiable Point-Based Radiance Fields for Efficient View Synthesis with Imperfect Masks

Jason C. Yuan
Department of Computer Science, Princeton University
jcyuan@princeton.edu

Abstract

Differentiable point-based radiance fields is an alternative algorithm to Neural Radiance Fields to perform novel view synthesis. Crucially, this algorithm relies on the use of object-background masks during training. In this work, we empirically demonstrate the negative effect of imperfect masks on this algorithm's render quality through a simulation of mask corruption. We then propose a two part modification to the algorithm to enable it to handle less-than-perfect masks. We demonstrate quantitatively and qualitatively that our proposed solution has high render quality even with imperfect masks.

1. Introduction

Novel view synthesis is the task of generating new views of a given object given a collection of training views. The canonical solution to this problem is Neural Radiance Fields (NeRF) [5], which models the scene as a continuous radiance field using a MLP. However, NeRF is extremely slow to train and render. The authors in [10] propose an alternative algorithm: differentiable point-based radiance fields. This approach represents a scene using a 3d point cloud with radiance parameters. This algorithm achieves near state of the art performance on the novel view synthesis task with orders of magnitude less training time, inference time, and memory requirement.

The initialization of the point cloud plays a crucial role in the differentiable point-based radiance field algorithm. A well-executed initialization greatly facilitates the optimization routine in generating a final point-cloud representation of superior quality. This algorithm takes an intuitive approach to achieve this by leveraging a foreground-background mask within the training data. Specifically, points are only initialized if they are consistent with all the masks.

Code available at `https://github.com/jasony123123/point-radiance`.

In real world applications, the assumption of an accurate mask may not hold. For example, many real world scenes will not even have a mask provided. Thus, in practice the mask will be generated using a segmentation network or using the depth data of RGBD images. In either case, such masks will have imperfections and not-necessarily be multi-view consistent.

In this work, we are interested in addressing the problem of decent-but-imperfect masks for the differentiable point-based radiance field algorithm. To summarize our contributions:

- We develop a method to simulate imperfect masks by adding a small amount of noise. We then empirically characterize the decrease in render quality of the algorithm on imperfect masks.
- We propose a two step solution to restore the performance of this algorithm even on imperfect masks. We quantitatively and qualitatively demonstrate the proposed solution enables differentiable-point-based-radiance-field to handle imperfect masks successfully.

2. Related Work

A wide variety of algorithms have been proposed for the novel view synthesis task, which have differing requirements in terms of masks. The algorithm that is the study of this paper was introduced in [10].

In [5], the authors introduce NeRF which uses a MLP to model the scene. It is orders of magnitude slower than [10] but masks are not needed. [7] present a modification to NeRF where a MLP is split into many smaller MLPs to accelerate rendering and also requires no masks. [3] propose an addition to NeRF using masks to reduce NERF network evaluations. [1] modifies NeRF to use a factorized expression, increasing the inference speed without requiring masks.

Plenoxels are a non-MLP approach introduced in [8] that are very fast but require significant memory and a mask to determine what volume to optimize. An alternative point-cloud based model is given in [6], but likewise requires

training data to be masked. [4] is another rendering technique that uses COLMAP to get a depth map instead of a mask.

Regularization of render outputs has also been used as a technique to reduce artifacts. [2] regularizes opacity near the camera by adding a loss term to maximize transmittance (*i.e*. transparency) in the near plane. [9] uses a penalty on the sum of densities near the camera to reduce floaters. Both techniques leverage the inductive bias that the space near the camera should generally be free of occlusions.

3. Method

3.1. Differentiable Point-Based Radiance Fields

The input of this algorithm is a viewing direction represented with camera intrinsic and extrinsic parameters R, t, M. The output is a 2d RGB image of the scene from that view. For a detailed explanation, refer to [10]. The pipeline is given in Figure 1.

The underlying model is represented by a point cloud in 3d space. Each point has a location parameter $P = (x, y, z)$ and a radiance parameter H. These parameters are learnt via stochastic gradient descent (Figure 1a)

The rendering algorithm consists of two phases. In the first phase, the 3d point cloud is projected into 2d space and a view-dependent color is computed for each points. The mapping from 3d to 2d is accomplished with traditional camera math using R, t, M, with depth information also retained. The radiance parameter H is the spherical harmonic parameters, which can be combined with R, t, M to compute a view-dependent RGB color (Figures 1b and 1c).

In the second phase, the 2d points (each with an associated depth and color) are splat rendered into the final output image. This involves using a Gaussian kernel to represent the density (α) parameter that each point has on each pixel, so that the the α value that a point has on a pixel is inversely proportional to the distance between the pixel and the point's 2d location. Then, each pixel's color is computed using traditional alpha blending, where we use each points color c, depth d and radiance α (Figures 1d, 1e, 1f). Since the entire pipeline is differentiable, stochastic gradient descent is applied to learn the point cloud parameters (x, y, z, H) for all the points.

One important aspect of this algorithm is the initialization of the point cloud. This is done using a rejection sampling technique. The training data is assumed to provide an object mask that describes what is the foreground (*i.e*. the object to be rendered) and the background (*i.e*. what we would like to ignore). Points (x, y, z) are sampled uniformly from 3d space, and only kept if they are consistent with all the masks in the training data. That is, each point is projected into 2d space of each training image, and only kept if it falls into the foreground mask of all the training images.

3.2. Mask Corruption Simulation

The use of the object masks is an important component of the differentiable point-based radiance field algorithm. It is also a limitation. The masks provided in the Blender dataset [5] are perfect, but in a real world application a mask will not be perfect.

We propose the following simulation of a less-than-perfect mask. Set a parameter $x \in [0\%, 100\%]$ (*i.e*. the "corruption level"). For each image mask, apply a random affine transformation consisting of: a random rotation between $[-90x, 90x]$, translation between $[0, 0.5x *$ (width or height)], scaling between $[1 - 0.5x, 1 + x]$, and shearing between $[0, 90x]$. Then, pass the imperfect masks as inputs to the original algorithm and measure the render quality.

3.3. Proposed Solution to Imperfect Masks

The corrupted masks changes the assumptions and provides a challenge to the existing algorithm. We propose the following solution to make differentiable point-based radiance fields more robust to imperfect masks.

First, relax the mask consistency constraint from a hard one to a soft constraint as suggested in [10]. Originally, we only keep points which fall in all masks during the initialization step. Instead, we keep a point if it falls into at least $x\%$ of the masks. x is a hyperparameter that must be chosen.

Second, we add a pretraining step consisting of 7 epochs of training only on the P_{position} parameters of the model. Additionally, in the pretraining step, we use a new loss function.

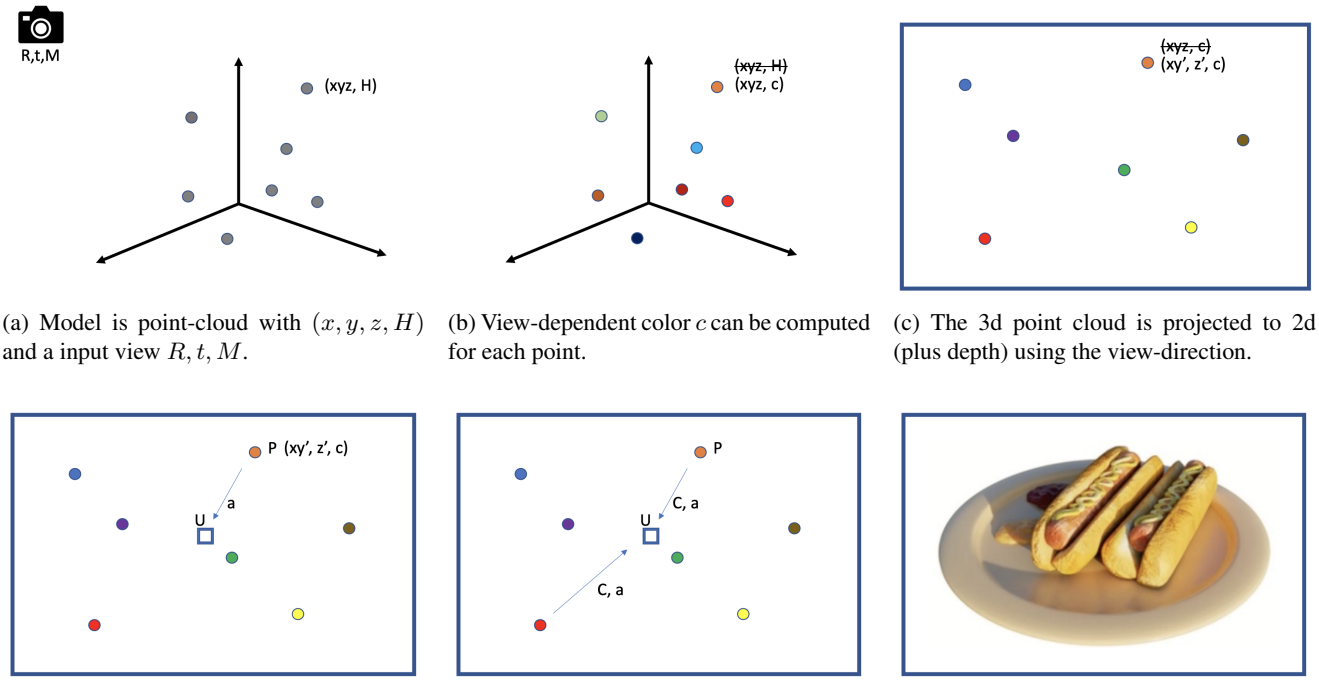
$$L = ||f(I_{\text{render}}) - f(I_{\text{ground-truth}})||_2^2 + \lambda_{\text{ridge}}||P_{\text{position}}||_2^2$$

$$f(I) = \tanh\left(\max_{\text{channel}}(I) \cdot 5\right)$$

4. Analysis

The natural solution to the problem of imperfect masks is to relax the mask constraint. However, this is just part of the solution proposed. In this section, we motivate the pretraining step of the proposed solution.

First, observe that it is difficult to know, apriori, what hyperparameter to choose when relaxing the mask constraint. If too small of a number is chosen, *i.e*. the mask constraint is relaxed too much, the initialized point cloud becomes highly inaccurate. For example, in Figure 2 observe that the initialized point cloud has degenerated into a ball around the object and there exists some outlier points which are not even attached to the main volume. This makes it challenging for the optimization.



(a) Model is point-cloud with (x, y, z, H) and a input view R, t, M.

(b) View-dependent color c can be computed for each point.

(c) The 3d point cloud is projected to 2d (plus depth) using the view-direction.

(d) The α density of a point P on a pixel U is given by a Gaussian kernel.

(e) Every point contributes a color c and density α to every pixel U

(f) The final output image after alpha blending (sorted by depth).

Figure 1: Overview of the rendering pipeline for the differentiable point-based radiance fields algorithm

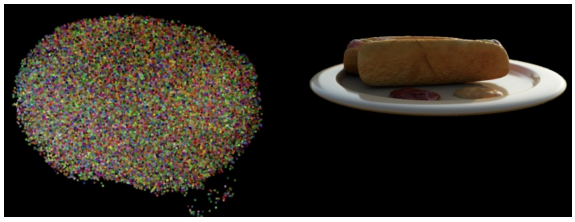


Figure 2: Corruption level 12%, mask consistency threshold 70%. The initial point cloud is on the left. The ground truth is on the right. Notice the outlier points in the bottom right of the initial point cloud.

Second, we observe that the dual optimization of radiance and position parameters makes it hard for a poorly initialized point cloud to converge to the right object volume positions. Figure 3 is an example of the model initialized with corrupted masks and relaxed mask constraints a few epochs into training. The initial point cloud is poorly initialized, such that many points are not where the object actually is. A few steps into training, the mass of points at the bottom (where there is no object in the ground truth) is being blacked out rather than their position being moved. Finally, in the final image there are artifacts below the object where no object should exist. From this, we hypothesize that the issue is that the dual optimization of position and color, when applied to a very inaccurate point cloud, will result in points being blacked out but not moved to their correct location, which leads to artifacts.

This motivates the pretraining step of the solution. In this phase, we train only the position parameters to avoid the "blacking-out" issue described above. Because we only care about the position and not the color, we pass both the ground truth and rendered image into the function f which essentially binarizes the pixels. Finally, to handle the fact that the initialized point cloud may have outliers and be very poorly initialized, we add a shrinkage term to the loss to bias point positions towards the center. The prior being that the object is likely centered. In this sense, we are leveraging a similar inductive bias as the regularization techniques used in [2] and [9].

5. Results

5.1. Mask Corruption Experiment

We apply the mask corruption detailed earlier for corruption percentages in 0% to 20%. Then, we run the original, unmodified algorithm. We find that as the corruption percentage increases, the model render quality declines significantly (Figure 4). Interestingly, the model size decreases at a steady linear-like rate with corruption percentage but the model render quality decreases at a slow rate until 8 per-

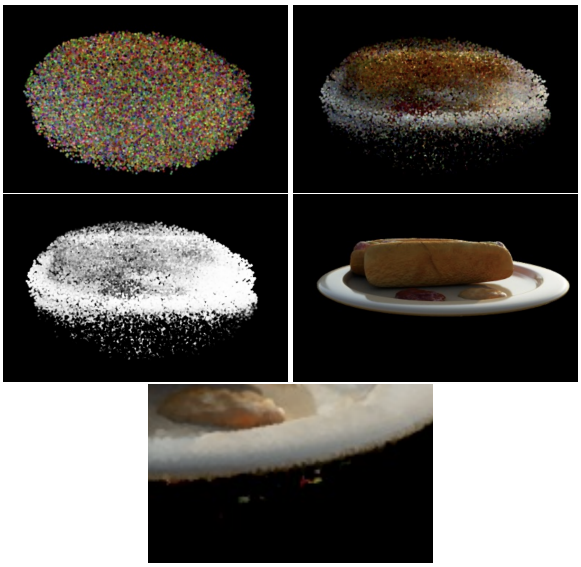


Figure 3: Corruption level 12%, mask consistency threshold 80%, no pretraining step. Initial point cloud (top left), after 5 epochs of training (top right and bottom left), ground truth (bottom right), artifacts in the final render (bottom center).

cent where it then decreases much more rapidly. This may suggest that the original algorithm is robust to small imperfections in the training object masks, but beyond a certain threshold, the optimization struggles to recover the original quality.

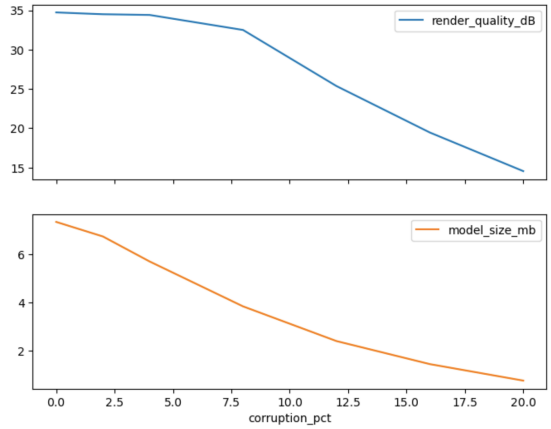


Figure 4: Varying the corruption level and running the unmodified algorithm.

Looking at the renders (Figure 5), we observe that as the corruption amount increases, the render becomes more and more sparse. By 12% corruption level, the rendered output has sparse dot-like artifacts. Examining the initialized point cloud in Figure 6 (*i.e.* the point cloud before any training is done), we see why: when the object mask corruption level increases, the volume in which the initialized point cloud is shrinks significantly and points are initialized in areas where the object is not present. This is in contrast to the initial point cloud generated from perfect masks: the object shape is almost perfectly captured before any training at all. So the imperfect masks presents two challenges. The first is that the masks are no longer multi-view consistent nor accurate so the hard rejection sampling mechanism is only able to initialize a small number of points in a small volume. Second, the initial point cloud from perfect masks is already close to the correct positions meaning there is not as much work for the optimization to do in terms of position, but for the imperfect masks the position optimization becomes very important.

Figure 5: Final renders of unmodified algorithm With corruption level of 0% (top left), 8% (top right), and 12% (bottom).

Figure 6: The inital point cloud with corruption level 0% (top row) and 8% (bottom row).

5.2. Evaluation of Proposed Solution

We fix the corruption percentage at 12% and investigate the efficacy of the first part of our proposed solution: relaxing the mask constraint from a hard one to a soft one. We run the algorithm and vary the mask consistency ratio from 100% (*i.e.* the original unmodified algorithm) to 70% (*i.e.* we keep an initial point if it falls in 70% of the training

masks). As the mask constraint is relaxed, the render quality improves dramatically to a level near the render quality of the original algorithm on perfect masks. However, this improvement only happens up to the 90% consistency requirement level. Once the mask constraint is relaxed further (*i.e.* decreased from 90%) the render quality starts to decline significantly (Figure 7). This observation is supported qualitatively. In Figure 8, we see that relaxing the mask constraint is only helpful to a certain point, after which the number of artifacts grows noticeably.

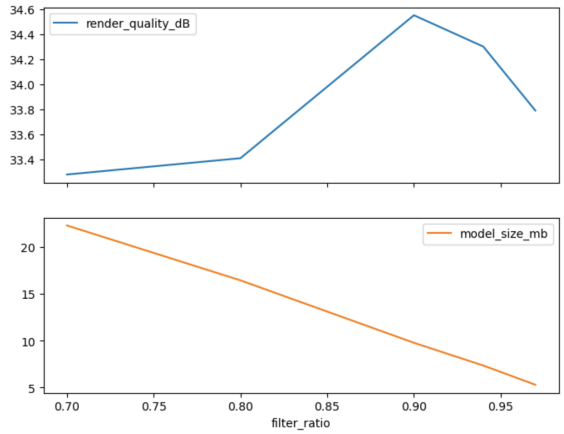


Figure 7: Experiment with fixed corruption level 12% and varying mask consistency threshold. No pretraining step.

Figure 8: Final renders with fixed corruption level of 12% and mask consistency threshold of 70% (top left), 90% (top right), 100% (bottom).

To understand further, we examine the initial point clouds before any training (Figure 9). We see that as the mask constraint is relaxed the initial volume grows (also clear in Figure 7). But the initial volume is highly inaccurate in both the high and low relaxation cases. It is more like expanding a spherical volume instead of actually matching the object volume.

We hypothesize that relaxing the mask constraint introduces more points but in a not-well-informed manner (unlike with perfect masks). So with a small relaxation, the additional points are still close enough to the object volume that it is helpful in terms of final render quality. But as the relaxation grows, the additional points start being added at more and more irrelevant positions and this poses a major challenge to the optimization algorithm that eventually causes the net effect on render quality to be negative.

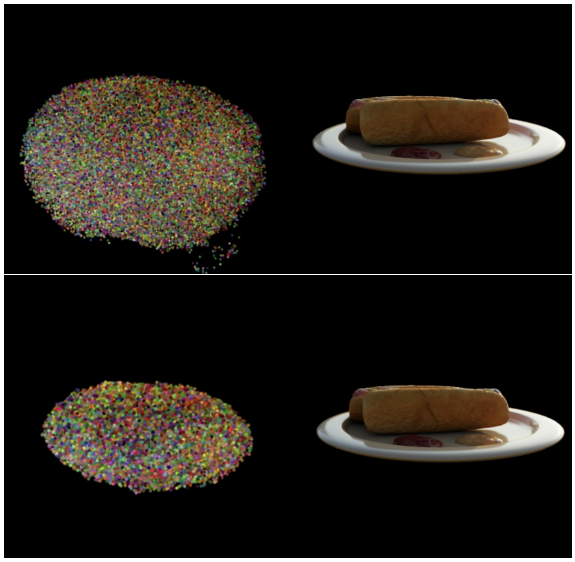


Figure 9: Initial point clouds with fixed corruption level of 12% and mask consistency threshold of 70% (left), 90% (right).

Finally, we investigate the results of applying both stages of the proposed solution. Fixing the corruption percentage at 12% and the mask relaxation constant at 80%, we apply our pretraining step. Quantitatively, we improve the render quality (measured in PSNR) from 33.41 dB without the pretraining step to 34.12 with the pretraining. And we decrease the model size from 16.44 mb to 16.08. Qualitatively, we observe fewer artifacts (Figure 10).

To understand further how the pretraining step is helping, we examine the point cloud after initialization and after the pretraining step (Figure 11). The initial point cloud occupies a volume that covers the real object volume but is highly inaccurate. After the pretraining step, we observe that the volume has shrunk significantly and begun to form sharper edges around the object volume, somewhat resembling the initial point cloud with perfect masks. We hypothesize the better shape of the initial point cloud after the pretraining step is responsible for the gains in render quality observed above.

Figure 10: Fixed 12% corruption level and 80% mask consistency threshold. Final renders with no pretraining step (left column) and with pretraining step (right column).

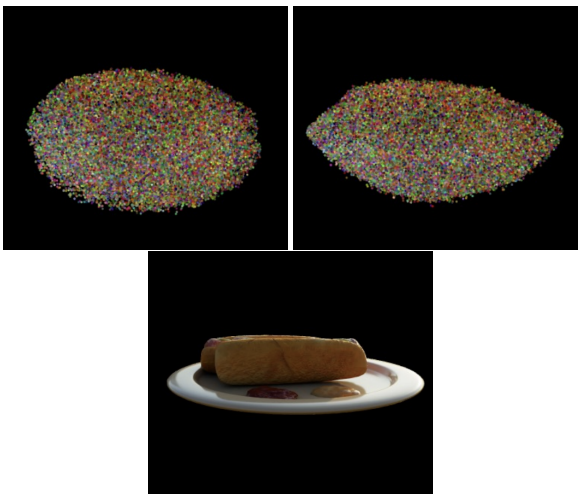


Figure 11: Fixed 12% corruption level and 80% mask consistency threshold. Point cloud at initialization (top left), after pretraining step (top right). Ground truth image (bottom).

6. Conclusion

In this work, we explore the differentiable point-based radiance fields algorithm in a modified setting: that of imperfect masks. We propose a technique to corrupt the masks, and demonstrate empirically the drastic decline in render quality. Then, we propose a two part solution which is to relax the mask constraint and add a small pretraining phase with a new loss function. We provide qualitative and quantitative results that demonstrate the success of the proposed solution.

The main limitation of this work was that it was done on only one scene of the Blender dataset. In addition, more robust results could be obtained by running the experiment across a broader range of corruption levels, mask consistency levels, and other hyperparameters.

References

[1] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *https://arxiv.org/abs/2103.10380*, 2021.

[2] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields, 2022.

[3] Naruya Kondo, Yuya Ikeda, Andrea Tagliasacchi, Yutaka Matsuo, Yoichi Ochiai, and Shixiang Shane Gu. Vaxnerf: Revisiting the classic for voxel-accelerated neural radiance field, 2021.

[4] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1440–1449, June 2021.

[5] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[6] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15969–15979, June 2022.

[7] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021.

[8] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

[9] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization, 2023.

[10] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. In *SIGGRAPH Asia 2022 Conference Papers*, SA '22, New York, NY, USA, 2022. Association for Computing Machinery.