# NeRFs in presence of Phase Optics

CHINMAY TALEGAONKAR, UCSD

This project mainly focuses on implementing NeRF from scratch, and developing a data generation pipeline using mitsuba3. Novel view reconstructions for 4 different scenes, with quantitative metrics are reported. The effect of positional encoding on the reconstructions has also been analyzed in one of the experiments. As an extension, the project also explored the idea of novel view synthesis from a NeRF in presence of a phase optic with known geometry, a uniform microlens array in this case. The performance is compared with standard NeRF for different training dataset sizes. While the Phase optic reduces spatial resolution of the captured data, it significantly increases angular resolution of the captured images. This motivates exploring the use of phase optic to capture data for view synthesis when the number of camera poses while training is very few. This claim is proved qualitatively in the results section.

Additional Key Words and Phrases: Neural Radiance Fields, View Synthesis, Mitsuba3

**ACM Reference Format:**
Chinmay Talegaonkar. 2024. NeRFs in presence of Phase Optics. *ACM Trans. Graph.* 1, 1 (October 2024), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Neural radiance fields (NeRFs) [Mildenhall et al. 2021] and related differentiable volumetric rendering-based methods [Barron et al. 2021; Verbin et al. 2022] are powerful tools for view synthesis. The main focus of this project is implementing NeRF from scratch to better understand the basic mathematical and computational toolkit used for NeRF and NeRF-like methods.

One drawback of these methods is that they need a lot of training views to accurately learn the underlying scene representation. A solution might be to acquire training data in presence of a phase optic to increase the angular resolution of the acquired data per image. Images acquired from a phase optic would have higher angular resolution than standard images, and can hence potentially decrease the number of training images needed for NeRF. For this project, I perform experiments with a uniform microlens array, however, my implementation can be extended to simulate ray tracing through non-uniform phase optic patterns as well (with different microlens radii). Refer to figure 7 for data rendered through different types of phase optic elements.

However, there's a trade-off here. Phase optic increases angular resolution, but causes a significant loss in spatial resolution. There are some potential ways around this trade-off, such as training a NeRF with a mixed data set, with and without phase optic. These will be explored in the future, but as a first step, this project focuses on expanding the self-implemented NeRF codebase to support training in presence of a phase optic and compare view synthesis results qualitatively and quantitatively against standard NeRF.

Author's address: Chinmay Talegaonkar, UCSD.


0730-0301/2024/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 2 OUTLINE

The basic implementation of NeRF from scratch is described first, with some visualizations of training poses. Qualitative and quantitative Results on 3 of the datasets shown in [Mildenhall et al. 2021], i.e. Lego, Drums, and Hotdog are demonstrated. Next, the effect of positional encoding on the final reconstructions is analyzed. This is followed by a brief description of the data generation pipeline developed using Mitsuba3. NeRF results are shown on custom data for spaceship scene generated using this pipeline. The rest of the report focuses on the phase optic part of the project. First, a theoretical parameterization of the phase optic is outlined. This is followed by a brief description of the ray tracing algorithm and assumptions used for propagating light rays through the phase optic. The next section outlines the experiments and results for training a NeRF in presence of the phase optic. The results show that when there are only 3 training images available, using a phase optic improves novel view reconstruction over standard NeRF on camera poses close to the training data poses.

## 3 NERF IMPLEMENTATION

All the components of NeRF were implemented from scratch, including the code to generate 360 degree videos[1]. This implementation doesn't support hierarchical sampling, as the ablation studies in [Mildenhall et al. 2021] show that there's not much relative benefit to hierarchical sampling. 256 samples per ray were used to compensate for the lack of hierarchical sampling. This implementation supports stratified sampling and can be trained on synthetic data generated from blender (as used in the NeRF paper). As a part of the project, scripts to generate training data were developed using mitsuba3 scenes. For faster training (and GPU constraints), the results are generated by training NeRF on 100 images of size $200 \times 200$, for 100,000 iterations. For qualitative evaluation, NeRF renderings from intermediate iterations are shown. PSNR and SSIM are used as quantitative metrics. The canonical rays are generated assuming that the camera looks at negative $z$ direction. Near and far planes ($t_n$ and $t_f$) are set to 2 and 6 for training and inference on the datasets provided in the original NeRF paper.

### 3.1 Implementation structure and procedure

The implementation procedure specific to this project is outlined here. For more details on the method and theory, refer to [Mildenhall et al. 2021]. First data processing scripts were written to extract the images and corresponding poses in the training set. A batch of rays originating from a pinhole camera was then generated. A ray starting from camera origin and incident at a pixel $(x, y)$ has the direction $x/f, -y/f, -1$. Then, for every image in the training set,

[1]The implementation is https://github.com/UCSD-Comp-Imaging/Nerf-Simple

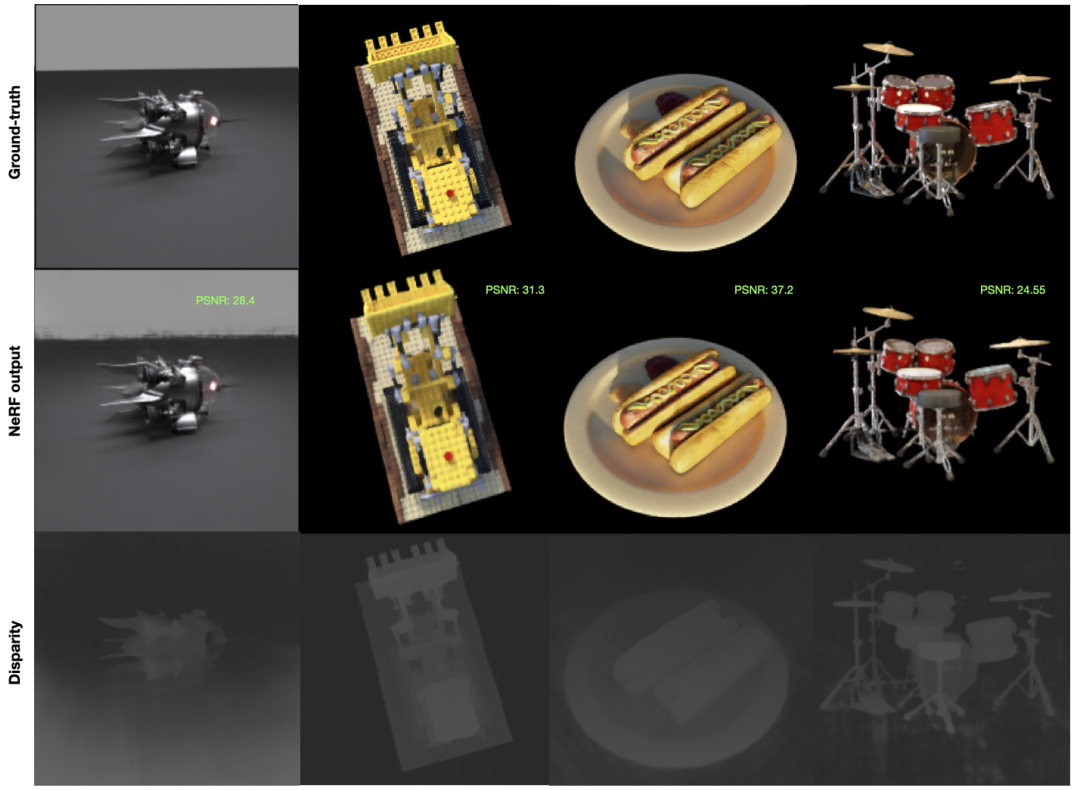


Fig. 1. **NeRF outputs for all 4 scenes compared against groundtruth**

camera rays are transformed according to the corresponding camera pose and are cached to be used for training. For an image size of $200 \times 200$, a total of 40,000 rays are generated. And since the training set has 100 images in different poses, that results in 4 million rays. While training, a batch of 1024 rays is selected randomly from the set of 4 million rays. The MLP in NeRF is evaluated at 256 points along each ray. At each sampled point $t_i$ along a ray $\mathbf{r}$, the MLP returns a volume density $\sigma_i$ and a color $\mathbf{c}_i$. The 256 points along each ray are selected using stratified sampling. The MLP evaluations along each ray are used to generate color for the given ray using the volume rendering equation

$$C(\mathbf{r}) = \sum_{i=1}^{256} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \text{where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

where $\delta_i = t_{i+1} - t_i$, i.e. distance between adjacent samples. The evaluated color is then regressed against the ground truth color for $\mathbf{r}$ using mean squared error loss.

To render images from a trained NeRF model, $C(\mathbf{r})$ is evaluated for all rays corresponding to the camera pose of the image, and the generated colors are filled into respective pixel locations. I used a higher batch size of 8000 while rendering images from a trained NeRF model to speed up the rendering process. It takes around 2.5 seconds to render the image at a resolution of $200 \times 200$ using the NeRF model implemented for this project. Training time for 100,000 iterations is approximately 4.5 hours at an image resolution of $200 \times 200$.

## 4 RESULTS

The implemented model was trained on 3 synthetic scenes provided in the original NeRF paper dataset. All images were downsampled to $200 \times 200$ for faster training. The learning rate $\beta$ was set to decay exponentially from $5 \times 10^{-4}$ to $5 \times 10^{-5}$. $\beta_t = \beta_0 \times 0.1^{\frac{t}{100N}}$. $\beta_0$ is the initial learning rate, and $N$ is the number of iterations, different for each scene. For each scene, the model was trained till the point that realistic-looking images started to emerge. Table 1 shows the average PSNR and SSIM values for NeRF models trained on different scenes. It is interesting to note that the number of iterations is somewhat correlated with the amount of high-frequency visual details in the scene. On the HOTDOG, just 30000 iterations give a good result, while on DRUMS scene a much higher number of iterations is needed to get a decent reconstruction quality.

Figure 1 shows RGB reconstructions and disparity maps by NeRF trained on each of the 3 scenes, for a given camera pose.

### 4.1 Reconstructions over iterations

Figure 2 shows the reconstructions by the model on validation images over time during the training process for all three synthetic scenes provided in the NeRF dataset.

| Scene | PSNR | SSIM | Num Iters |
|---|---|---|---|
| LEGO | 30.84 | 0.958 | 60000 |
| HOTDOG | 34.1 | 0.963 | 30000 |
| DRUMS | 25.95 | 0.927 | 90000 |
| SPACESHIP | 25.8 | 0.91 | 100000 |

Table 1. Average PSNR and SSIM for different scenes. The average PSNR and SSIM are computed by taking an average of PSNR and SSIM values for each image in the test set.

Since videos are the best way to appreciate view synthesis results, the 360-degree view synthesis videos of these scenes are being linked here. These videos were shown in the class presentation as well.

### 4.2 Effect of Positional Encoding

As described in the main paper, positional encoding has a significant effect on the performance of NeRF. I replicated that experiment on the lego scene, by training a model without positional encoding. Figure 3 shows the obtained result. Without positional encoding, PSNR and SSIM obtained were 26.7 and 0.9 respectively for the LEGO scene. The positional encoding parameters were the same as that in the NeRF paper.

## 5 DATA GENERATION USING MITSUBA3

Data generation scripts using Mitsuba3 were developed for evaluating other diverse kinds of scenes, and to also render data from a camera + microlens setup for the phase optic part of the project. This script is helpful beyond this project as well, since it was difficult to find any ready-to-use tools for generating training data for NeRF or NeRF-like models. Some of the important considerations while rendering data using mitsuba3 are

- The scenes in Mitsuba3 gallery have meshes in Y-up coordinate system, which needs to be taken into account while sampling poses on a hemisphere around the scene for rendering.
- Training data for NeRF is generated by moving the camera on a hemisphere, with the camera always looking close to the center of the scene. The scene files in mitsuba3 don't always have the scene centroid as the origin. Hence one needs to parse the scene files, and manually compute the centroid of the scene by traversing through all the meshes.

Figure 4 below shows the visualized camera poses and ray directions used for the training set of 100 images for the spaceship scene in the Mitsuba3 gallery.

### 5.1 Results on spaceship scene

For training NeRF on the spaceship dataset, $t_n$ and $t_f$ were set to 2 and 6. The model was trained for 40000 iterations. The average PSNR and SSIM on the validation set were 25.8 and 0.911 respectively. Figure 5 shows comparisons of reconstructions from some of the views of the spaceship.

The performance on the spaceship might further benefit from increasing the far plane distance and using more training, as currently, it seems like the model seems to learn the spaceship well, but is struggling to learn the background smoothly. The decent PSNR values obtained for the spaceship scene validate the data generation pipeline developed using Mitsuba3. A video of novel view renderings of the spaceship was shown in the class presentation.

## 6 PHASE OPTIC PARAMETERIZATION

This section describes the phase optic model implemented for this project. A closed-form parameterization of a phase optic element was used for better control of the surface geometry, and to speed up the ray-tracing code developed in python. The phase optic element is parameterized using spheres (as ray intersections through spheres has a closed form). The parameterization is as follows. Consider $N$ spheres $[S_1, S_2 \dots S_N]$ of refractive index $\mu$ distributed around a plane (the image sensor plane), as shown in Figure 6. Denote the phase optic surface as $\phi(x, y)$. Assume here that the camera faces negative z (image plane is at $z = -1$) for convenience and the camera origin at $(0, 0, 0)$. Let $S_i^z(x, y)$ denote the $z$ coordinate of the intersection of a ray drawn from $(x, y, -1)$ perpendicular to the image plane, directed away from the camera origin, i.e. in the negative z direction. Hence, $S_i^z(x, y) < -1$ for any valid intersection, as every ray travels in the negative z direction and starts at $z = -1$. If there's no intersection $S_i^z(x, y) = \text{Inf}$. The described ray tracing can be extended to the case where the camera faces the positive z-axis by changing directions and signs appropriately. Therefore, the phase optic surface $\phi(x.y)$ can be written as

$$\phi(x, y) = \min_i S_i^z(x, y)$$

Figure 6 below describes the parameterization intuitively. The green line denotes the phase optic surface profile.

### 6.1 Ray Tracing Through Phase Optic

Ray tracing through this setup for a general ray originating from the camera origin can be done as follows - Consider a ray originating from $(0, 0, 0)$ i.e. the camera origin. This ray can be written as $\mathbf{r} = t\mathbf{d}$. First, compute the intersection of the image sensor plane with $\mathbf{r}$, and then compute the refracted ray using the vector form of Snell's law, i.e.

$$\mathbf{n} \times \mathbf{r} = \mu \mathbf{n} \times \mathbf{r}'$$

Here $\mathbf{n}$ denotes the normal vector to the surface at the point of intersection of the ray to the surface. $\mu$ denotes the refractive index of the medium (glass) w.r.t air. The new refracted ray is denoted as $\mathbf{r}' = \mathbf{o}' + t\mathbf{d}'$ .

Now, compute the intersection of $\mathbf{r}'$ with each sphere $S_i$. Let $t_i$ be the value of $t$ on solving the quadratic equation for the intersection of a ray with the sphere. One assumption here is that the phase surface is continuous, and connected (i.e., there are no spheres floating in the air). Hence, the intersection point of the ray $\mathbf{r}'$ with the front surface of phase optic is the sphere $S_i$ with a maximum value of $t_i \; \forall i \in \{1 \dots N\}$. Let $i^* = \max_i t_i$. The origin of the final ray exiting the surface of phase optic corresponding $\mathbf{r}$ is $\mathbf{o}'' = \mathbf{o}' + t_{i^*}\mathbf{d}'$. The direction of this ray can be computed using the vector form of Snell's law. The normal vector at $\mathbf{o}''$ can be computed as $\mathbf{n} = \mathbf{o}'' - C_{max}$

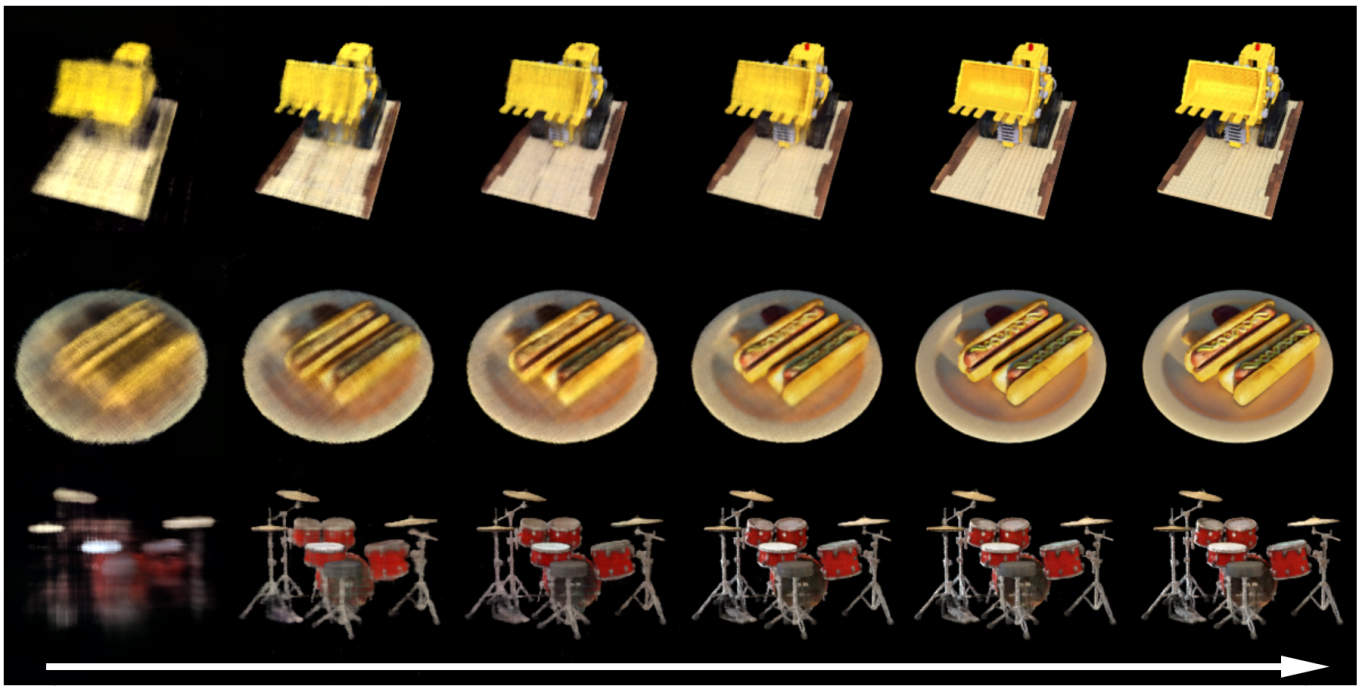

Fig. 2. Reconstructions for all 3 scenes at increasing iterations from left to right.

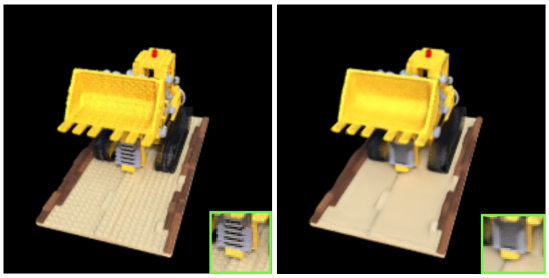

Fig. 3. **Effect of Positional Encoding** Left: Image reconstruction from a NeRF model trained with positional encoding. You can see in the bottom left inset that the model is able to recover high-frequency details. Right: Reconstruction from a model trained without positional encoding. Without positional encoding, the model heavily blurs out high-frequency details.

where $C_{max}$ denotes the center of the sphere corresponding to $t_{i^*}$. The final ray can be denoted as $\mathbf{r}'' = \mathbf{o}'' + t\mathbf{d}''$ . Hence for every ray originating from the camera, there is a corresponding ray that exits the phase optic surface. This correspondence map between rays can be cached and used to render scenes from a pre-trained NeRF model in presence of a phase optic, such as a microlens array. The effect of total internal reflection (TIR) is currently being ignored. If a ray shot from the camera undergoes TIR at the surface of the phase optic, the ray is allowed to pass through for simplicity. The ray tracing will be offloaded to Mitsuba3 in the future to alleviate this issue and for higher ray tracing speed. For the Phase Optic patterns used in my experiments, less than 15% of the total rays shot through the camera undergo TIR. Using this parameterization it is possible to design

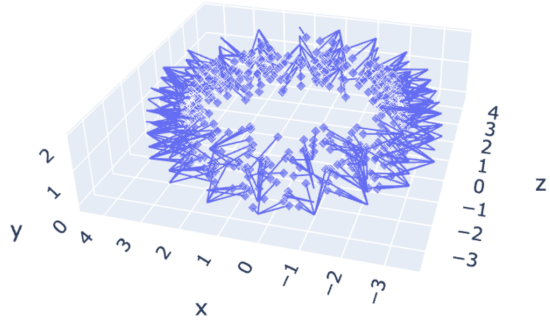


Fig. 4. **Poses of training images for spaceship data** The azimuth angle for the pose $\phi$ is sampled uniformly between 0 and 360 degrees. The zenith angle $\theta$ is sampled uniformly between 70 and 90 degrees. The value of $\theta$ was motivated by the spaceship being an elongated elliptical object. Note that since mitsuba scenes are in Y-up coordinate systems, movement along azimuth is rotation along the Y-axis. The validation set and test set poses for this experiment were sampled randomly in the same $\theta$ and $\phi$ range. The hemisphere radius is set to 4. This visualization is generated using the visu3d library.

phase optics with very little (less than 5%) TIR as well. At this stage TIR doesn't really affect the results that adversely, and since it's possible to design elements with minimal TIR, ignoring refraction for rays that undergo TIR shouldn't change the conclusions obtained from subsequent experiments. Note that the NeRF implementation assumes access to the ray directions through the phase optic while training and inference.

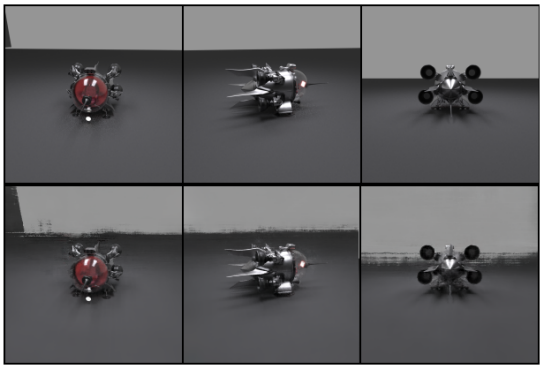

Fig. 5. Comparing spaceship reconstructions (bottom row) with corresponding ground truths (top row)

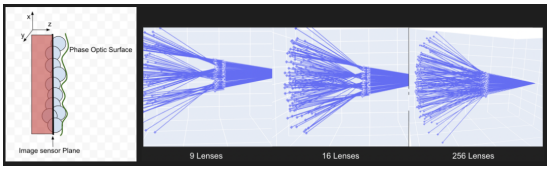


Fig. 6. (a) Phase Optic Parameterization using surface of spheres. (b) Resulting sampling pattern from microlens array on a varying number of lenslets.

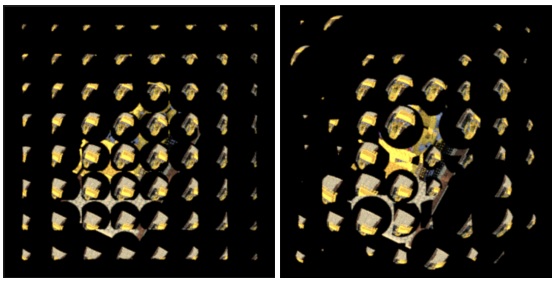

Fig. 7. **Images rendered in mitsuba in presence of a phase optic** (a) Uniform microlens array with 100 lens elements (b) microlens array with 100 elements, randomized lens radii and lens centers

## 6.2 Rendering through Phase Optic

Data in presence of a phase optic was rendered using Mitsuba3 renderer. The renderer can be given custom rays as inputs. Using the ray tracing procedure described above, for each ray shot through a camera pixel, a corresponding ray that exits the phase optic can be computed. For each pixel, the corresponding ray that exits the phase optic is computed, and then given as input to the renderer to evaluate color at that pixel.

Figure 7 shows some sample images generated using a uniform phase optic, as well as a random phase optic.

# 7 EXPERIMENTS

This section outlines the set of experiments done with phase optic data and discusses some results and potential issues.

| Dataset | PSNR | SSIM |
|---|---|---|
| No Phase Optic | 31.06 | 0.957 |
| 64 lenslets | 23.40 | 0.60 |
| 9 lenslets | 25.56 | 0.68 |

Table 2. Average PSNR and SSIM on datasets with and without phase optic.

## 7.1 Training NeRF in presence of Phase Optic

First, a performance comparison is done between NeRF trained with and without a phase optic. 100 training poses were randomly sampled on a hemisphere with a radius of 2. On the same hemisphere, 20 validation and 20 test poses were sampled for testing. Using this fixed set of poses, three datasets were generated -

- D1: Rendered all the poses at $400 \times 400$ resolution without a Phase Optic. This dataset was used to train a standard NeRF model. The model was trained for 20000 iterations.
- D2: The poses were rendered at $400 \times 400$ resolution in presence of a uniform microlens array, that has 100 microlens elements equally spaced across the image plane. The model was trained on this data for 20000 iterations.
- D3: This is the same as D2, but instead of using 100 microlenses, 25 microlenses were used.

Figure 10 below shows sample images from the datasets mentioned above.

The NeRF models were trained for 20000 iterations on all three datasets. To evaluate the models, PNSR and SSIM were computed for 10 novel views in the test set shown in table 2.

This experiment demonstrated that NeRF is able to learn a volumetric representation of a scene in presence of a phase optic. However, due to the loss of resolution in phase optic, the reconstructions suffer a loss in details, hence the PSNRs are lower when Phase Optic is used. The PSNR is lower when 100 lenses are used compared to using 25 lenses, as the spatial resolution of the input data reduces with the number of lenslets used. This experiment suggests that if sufficient diversity in the training data poses is ensured, there's not much benefit to using a Phase optic, as it would lead to a loss in spatial resolution of learned volumetric representation.

## 7.2 Training with limited views

This experiment focuses on exploring the potential benefits of using a phase optic with very few training views. The increased angular resolution in images captured using a phase optic should improve interpolation between views. Standard NeRF with very few images should overfit and hence would have very poor reconstruction performance on novel views (the novel views are assumed to be around the poses of the input images). 3 datasets were generated for this experiment, at $400 \times 400$ image resolution. The models were trained for 20000 iterations (due to lack of time and GPU resources). First, 3 random viewpoints (camera poses) are selected on a hemisphere. This set of poses is the same for all three datasets. For the first dataset (D0), the three poses are rendered using mitsuba3 without any phase optic. For the second and third datasets (D100, D25) the

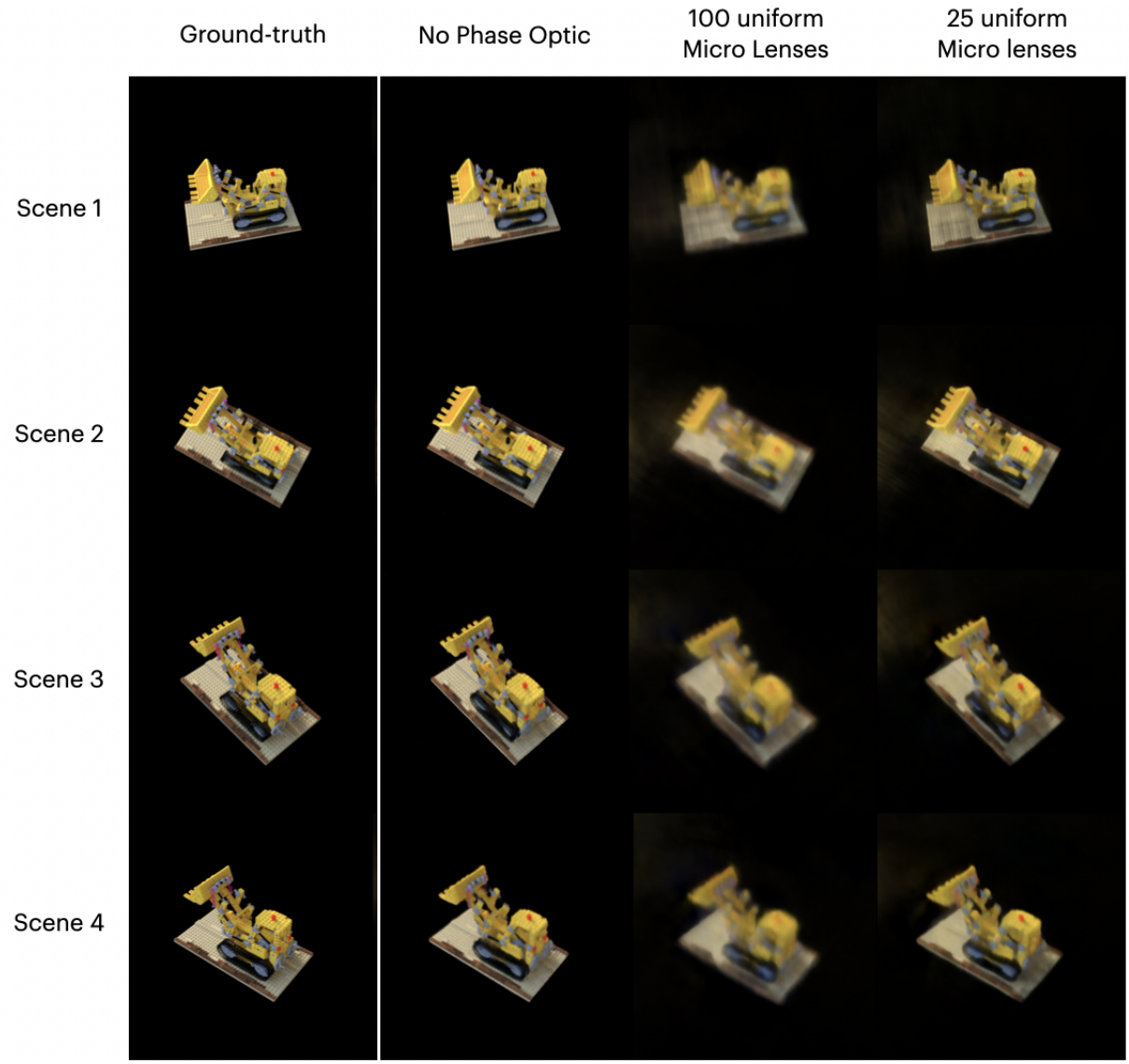


Fig. 8. **Novel views rendered using NeRF trained with and without phase optic** NeRF is able to render novel views when trained with phase optic data, but loses spatial resolution. The spatial resolution of input data is inversely proportional to number of microlenses. This explains why the model trained on images rendered through 25 microlenses produces sharper renderings than the model that used 100 microlenses data.

poses are rendered through a phase optic with 100 microlenses, and 25 microlenses respectively.

At inference time, the reconstruction performance is evaluated on novel viewpoints generated around the training viewpoints. To generate novel viewpoints close to the training images, the camera is moved on a circle formed by the intersection of the hemisphere, and a cone whose apex coincides with the center of the hemisphere, and the axis is the line joining the center of the hemisphere and the camera viewpoint location on the hemisphere. The distance between the training pose and the inference poses is controlled by the half angle of the cone. More formally, assume that the radius hemisphere is $r$. Consider a cone with half angle $\alpha$, apex at origin, and axis to be $z$ axis. A point on the intersection of the hemisphere and the cone can then be parameterized as $\mathbf{p} = (r \sin\alpha \sin u, r \sin\alpha \cos u, r \cos\alpha)$. Sampling values of $u \in [0, 2\pi]$ can be used to sample the points on the intersectional circle of the cone and hemisphere. The points sampled on the intersectional circle can be transformed using the training poses to generate novel viewpoints for inference around the training poses. For generating test videos, 30 points along a cone with $\alpha = 15$ degrees were sampled.

Figure 9 shows images of one of the (novel) viewpoints on this cone, rendered using NeRF models trained with and without phase optic data. Figure 9 qualitatively shows that D25 (center) is the best-looking image. It is much better than not using any phase optic, i.e.

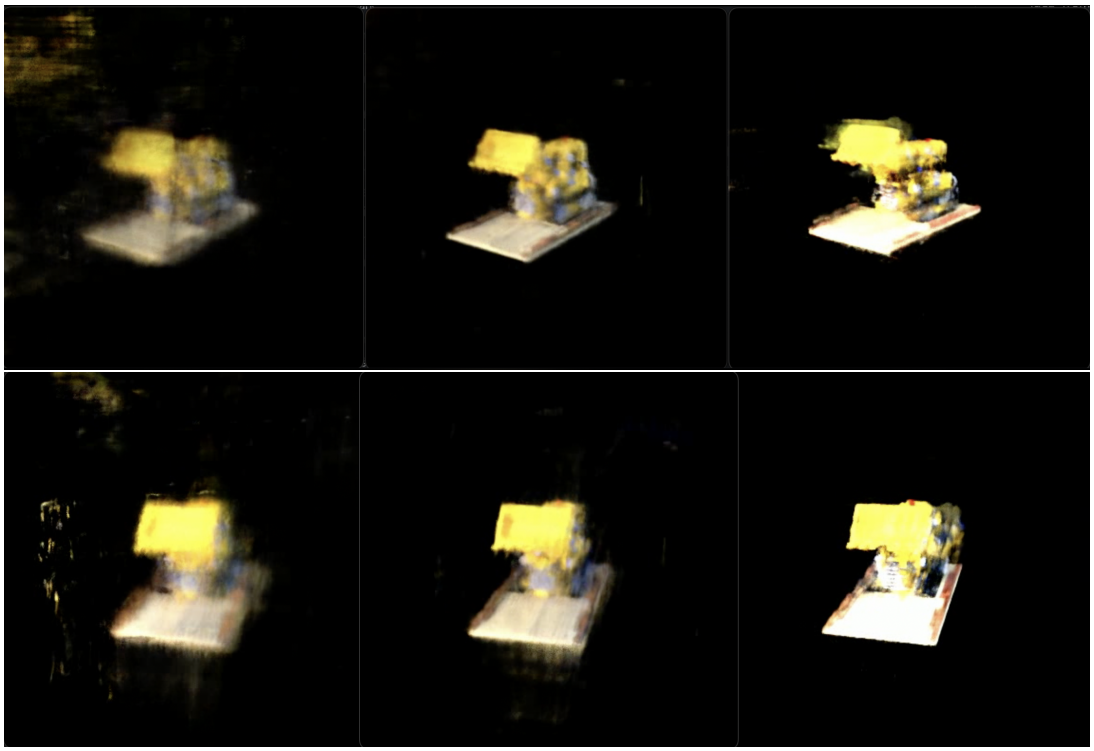

Fig. 9. **Novel Views rendered by NeRF trained on only 3 images** Novel view renderings using the trained NeRF models on the datasets mentioned in 7.2. **Left**: NeRF trained on images rendered with 100 microlenses phase optic. **Center**: NeRF trained on images rendered with 25 microlenses. **Right**: Standard NeRF without any phase optic.

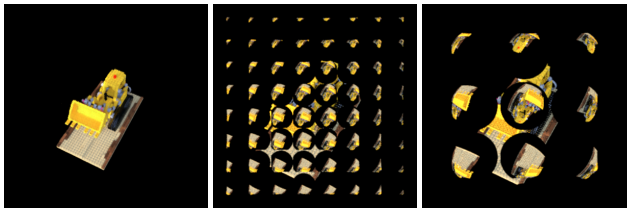

Fig. 10. **Sample images from datasets for phase optic experiments** (a) D1 (b) D2 (c) D3. Note that since D3 has fewer lenses, and larger lens radii, the images from microlenses in D3 have better spatial resolution than that in D2.

D0 (right). This validates the initial claim that using Phase optics for situations with extremely low training images might lead to better performance than standard NeRF. The right-most image has some missing details and holes near the central region of the bulldozer, while the $D25$ image (with 25 lenses) is able to fill in those details due to more angular information provided during training.

D100 (left), while having a higher angular resolution in the training data, looks worse than D25. This can be attributed to the spatio-angular resolution trade-off. Using 100 lenslets affects the spatial resolution much more adversely than using only 25 lenslets.

## 8 CONCLUSION AND FUTURE WORK

In conclusion, the correctness of the NeRF implementation was demonstrated through novel view reconstructions on a wide variety of scenes. A data rendering suite was also developed using mitsuba3, to render custom datasets for NeRF training. Ray tracing through a phase optic was integrated into the data generation script, to render datasets as seen through a phase optic. The NeRF implementation was extended by incorporating phase optic ray tracing, and a NeRF model was trained successfully on data rendered through a phase optic. This was followed by experiments to show the benefit of using phase optics for scenarios with very few input images. In fact, since 3 different views were used, the standard NeRF still had enough information to learn parts of the basic low-frequency scene geometry. The same experiment performed with a single viewpoint would be even better to show the advantage of phase optics when few training images are available.

This report explored a very limited subset of all possible phase optics one could use. Evaluating the effect of random phase optics would be interesting. The second phase optic experiment done with 3 images should be repeated on a single image, since that would enhance the relative benefit of phase optic even more. This approach of incorporating phase optic in training should also be tried with other NeRF-like methods as well, such as MIP-NeRF[Barron et al. 2021], Plenoxels [Yu et al. 2021] etc. On using lesser images for training with phase optic, the learned volumetric representation has

a lot of cloud-like artifacts. This needs to be further investigated and mitigated for better reconstruction quality.

## REFERENCES

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5855–5864.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. *Nerf: Representing scenes as neural radiance fields for view synthesis*. Vol. 65. ACM New York, NY, USA. 99–106 pages.

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. 2022. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 5481–5490.

Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2021. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131* (2021).