NEF: Neural Error Fields for Follow-up Training with Fewer Rays

Alessandro Luchetti[1]*[a], Kenta Ito[2]*[b], Dieter Schmalstieg[3][c],
Denis Kalkofen[4][d], and Shohei Mori[3,2][e]

[1]*University of Trento, Trento, Trentino-Alto Adige, Italy*
[2]*Keio University, Yokohama, Kanagawa, Japan*
[3]*University of Stuttgart, Vaihigen, Baden-Württemberg, Germany*
[4]*Graz University of Technology, Graz, Styria, Austria*
**Equal contribution, alessandro.luchetti@unitn.it, kenta.ito@keio.jp, kalkofen@tugraz.at,*
{dieter.schmalstieg, shohei.mori}@visus.uni-stuttgart.de

Keywords: Neural radiance fields, error visualization, novel view synthesis, follow-up training.

Abstract: A Neural Radiance Field (NeRF) is capable of representing scenes by capturing view-dependent properties from a specific set of images through neural network training. The lack of a significant initial image set implies that additional photographing session and training are required to improve the final view synthesis. For this purpose, we introduce a new variant of NeRF training analysis, termed the Neural Error Field (NEF). NEF visualizes and identifies view-dependent errors to reduce the number of ray samples used in the follow-up training. NEF does not require modifications to the NeRF core and training process. We evaluate and verify the accuracy of the results achieved with NEF on several public datasets, including real and synthetic images, and bounded and unbounded scenes.

1 INTRODUCTION

A Neural Radiance Field (NeRF) is an implicit scene representation using a neural network to sample view-dependent properties such as colors and densities through volumetric rendering (Mildenhall et al., 2020). The network is trained so that the rendered views match the imaged view-dependent attributes. Significant progress has been made in improving efficiency, reducing training times to minutes and rendering times to milliseconds (Müller et al., 2022). Nonetheless, lacking a substantial image dataset can require a subsequent photographing session and additional training to enhance the final view synthesis.

Given an additional image dataset, one might use all images to train a new NeRF model. However, leveraging the initial information from the first dataset should allow for more efficient ways to classify ray samples that contribute to the final results, enhancing efficiency and effectiveness. For example, uncertainty modeling and visualization in trained NeRFs may filter out unnecessary rays (Goli et al., 2024; Shen et al., 2021; Sünderhauf et al., 2023). Progressive screen-space analysis focuses only on impactful pixels (Zhang et al., 2023). However, these are designed to filter out non-contributing pixels during or after training for quality rendering, and thus, the knowledge may not transfer to the follow-up training purpose right away.

We aim to analyze the initial dataset and investigate how to filter out pixels with less impact for efficient and effective follow-up training. To this end, we introduce a NeRF variant, Neural Error Fields (NEF), a novel volumetric error representation for view-dependent and occlusion-aware error projection, enabling ray filtering. NEF enables predicting error at unseen viewpoints, which can be leveraged during the follow-up photographing session to identify undersampled or erroneous areas. It can also estimate ray filtering from new viewpoints.

We validate the design choices that make NEF training and rendering effective by comparing variants of error encodings, NEF training strategies, and post-image filtering. We demonstrate that utilizing NEF for the follow-up training can reduce the number of ray samples and maintain final rendering qual-

[a] https://orcid.org/0000-0003-0960-0996
[b] https://orcid.org/0009-0002-1469-6662
[c] https://orcid.org/0000-0003-2813-2235
[d] https://orcid.org/0000-0002-0359-206X
[e] https://orcid.org/0000-0003-0540-7312

ity. Furthermore, to support seamless integration with existing NeRF approaches, we design NEF as a post-hoc process, which follows common NeRF training (Tancik et al., 2023).

2 BACKGROUND

2.1 Neural Radiance Fields Training

Neural networks have shaped computer vision and graphics problems over the past few years, and NeRF is one of the notable examples (Mildenhall et al., 2020). NeRF represents scenes as a multi-layer perceptron. Pixel colors are rendered by sampling colors and densities along view rays. NeRF training optimizes the network weights so that the photometric differences between the rendering and corresponding pixels are minimized. Thus, NeRF learns the view-dependent properties of the scene.

For brevity, we abstract the training process and rendering process as follows:

$$\mathbf{W}_{n+1} := \mathcal{T}(\mathbf{D}, \mathbf{W}_n),\ \hat{\mathbf{c}} := \mathcal{R}(\mathbf{x}, \mathbf{W}_{n+1}).$$

Here, $\mathbf{x}$ denotes the 5D spatial input, $\mathbf{c}$, colors, and $\mathbf{W}$, trained weights of NeRF. An element $\mathbf{d}_j \in \mathbf{D}$ is given as $\mathbf{d}_j = \mathbf{x}_j, \mathbf{c}_j$, where $\mathbf{D}$ is a dataset.

2.2 Uncertainty Analysis of NeRF

The best practice to improve the performance of a NeRF model is to provide a comprehensive set of input images that potentially offer superior spatial coverage. However, no existing work adequately clarifies how minimal inputs can be formed. NeRF is a neural scene representation that implicitly identifies depth points via differentiable volume rendering. This characteristic prevents us from applying the plenoptic sampling theory of conventional light field data (Chai et al., 2000; Mildenhall et al., 2019; Ishikawa et al., 2023).

NeRF can perform an uncertainty analysis during or after training. Analysis during training requires modification of the network and loss designs (Martin-Brualla et al., 2021). Besides, considering the rapid emergence of new NeRF variants (Tewari et al., 2022), post-hoc approaches are a reasonable design choice (Goli et al., 2024). The NeRF ensemble approach involves multiple pieces of NeRF training with different initializations (Sünderhauf et al., 2023). Additional voxel grids are often used to describe uncertainties in NeRF (Goli et al., 2024; Warburg et al., 2023). The additional voxel samples are typically combined with the rendering samples to filter out artifacts or *floaters* (Philip and Deschaintre, 2023). Visual error tomography (Franke et al., 2023) explores the visualization of errors in a similar way. However, it does not aim at efficient training with additional datasets, nor does it model view dependency like we do.

Our approach requires neither NeRF core modifications nor additional voxel grids. We require only the input data for an existing NeRF framework, making no changes to the existing solution. Instead of attempting to find an uncertainty that correlates with the mean squared error (MSE) of the pixels (Jin et al., 2023), we directly estimate the error images.

Table 1: Follow-up training results on the Bonsai dataset (Barron et al., 2022) using different training strategies in the case of a 20× increase in dataset size for follow-up training (i.e., simulating this using a full dataset by a 5 : 95 split of initial $\mathbf{D}$ and additional $\mathbf{D}'$ datasets). The first two perform fine-tuning, and the third starts from scratch.

	PSNR↑	SSIM↑	LPIPS↓
$\mathcal{T}(\mathbf{D}', \mathbf{W}_1) \rightarrow \mathbf{W}_2$	23.866	0.814	0.103
$\mathcal{T}(\mathbf{D} \oplus \mathbf{D}', \mathbf{W}_1) \rightarrow \mathbf{W}_2$	25.195	0.852	0.098
$\mathcal{T}(\mathbf{D} \oplus \mathbf{D}', \mathbf{W}_0) \rightarrow \mathbf{W}_1$	26.160	0.890	0.018

2.3 Strategic Data Selection Considerations

If training with an initial dataset $\mathbf{D}$ is unsatisfactory, unpleasant visual artifacts can be filtered out from the NeRF rendering (Goli et al., 2024; Warburg et al., 2023) (Sect. 2.2) or substitute a follow-up training dataset, $\mathbf{D}'$, for improvements: $\mathbf{W}_{n+1} := \mathcal{T}(\mathbf{D} \oplus \mathbf{D}', \mathbf{W}_n)$ where $\oplus$ represents a data concatenation, allowing a follow-up training from scratch with a given complete dataset for $n = 0$ or can be trained with fine-tuning for $n = 1$, or simply use only the follow-up training dataset: $\mathbf{W}_{n+1} := \mathcal{T}(\mathbf{D}', \mathbf{W}_n)$.

NeRF variants utilize coarse initializations and denser sampling around estimates (Mildenhall et al., 2020; Tancik et al., 2023; Müller et al., 2022). We observed that the fine-tuning approach results in erroneous results due to incorrect initial guesses. Especially if $\mathbf{D}$ is small, the trained weights can be sub-optimal, and starting with these weights can lead to a worse result. The example in Table 1 demonstrates on the Bonsai dataset of Mip-NeRF 360 (Barron et al., 2021; Barron et al., 2022) that a follow-up training from scratch achieves the best performance. Based on this result, all subsequent follow-up training discussed will adopt the training from scratch approach, $\mathcal{T}(\mathbf{D} \oplus \mathbf{D}', \mathbf{W}_0) \rightarrow \mathbf{W}_1$.

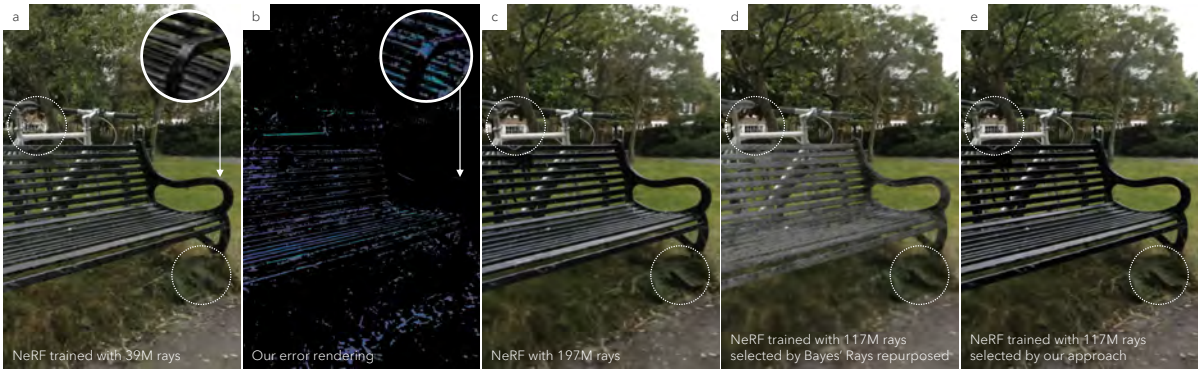


Figure 1: NeRF rendering with different amounts of total number of ray samples (a, c–e) and NEF (b), focusing on post-hoc ray pruning methods. While NeRF can render learned scene colors (a), NEF can render potential errors trained using the same dataset. Similarly to NeRF, which can synthesize view-dependent colors, NEF can represent view-dependent photometric errors and occlusions (insets from a different angle). This property allows us to identify the pixels or rays that contribute to follow-up training for updating the original NeRF model with a smaller number of samples (c vs. e). The same number of samples from the literature (Goli et al., 2024) fails to capture such pixels.

3 Neural Error Fields

To achieve a post-hoc approach, we modify only the inputs to bypass the repeated NeRF training without touching their core components. The pipeline consists of the following steps: (1) Initial NeRF training, (2) error map generation, (3) NEF training continued from step 1, (4) subset filtering via NEF rendering, and (5) follow-up NeRF training with the subset.

Initial NeRF training. The extrinsic and intrinsic parameters are calculated from the initial training data, $\mathbf{D}$. These parameters are independent of the NeRF algorithm and are calculated via a SLAM-enabled camera (e.g., Polycam[1]) or a structure-from-motion approach (e.g., COLMAP (Schönberger and Frahm, 2016)). Any NeRF algorithm $\mathbf{W}_1 := \mathcal{T}(\mathbf{D}, \mathbf{W}_0)$ can be selected for NEF optimization, as we keep their core modules untouched.

Error map generation. We generate error maps at the input viewpoints, $\mathbf{e} = |\mathbf{c} - \mathcal{R}(\mathbf{x}, \mathbf{W}_1)|_1$, by calculating the absolute difference between the trained NeRF renderings and the input images. The error maps are processed to obtain $\mathbf{e}' = f_\mathrm{e}(\mathbf{e})$, a visualization of the error map in 3D with NEF. We evaluate how different designs of error color encoding, f_e, give different results in Section 4.1.

NEF training continued from the initial NeRF training. A copy of the pre-trained NeRF model is used to train NEF. The color images of the initial training, $\mathbf{c}'$, are replaced with the error images, $\mathbf{e}'$ for NEF training. In other words, we recolorize the original NeRF model with $\mathbf{W}_1$ using the error map colors and obtain $\mathbf{W}_{\mathbf{e}'} := \mathcal{T}(\{\mathbf{x}_j, \mathbf{e}'_j\}, \mathbf{W}_1)$. This training process is intended as transfer learning to achieve quick convergence, exploiting the 3D space already identified during the first training.

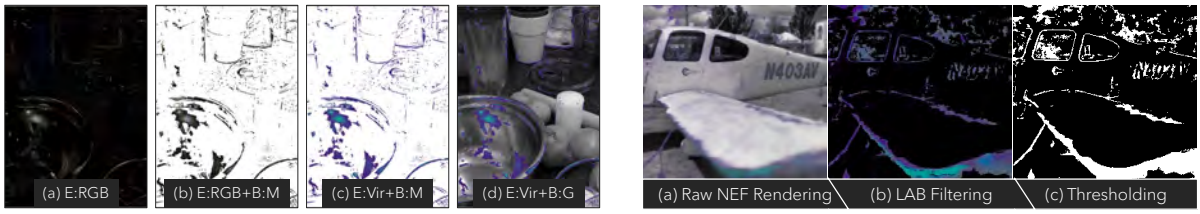
Pixel filtering with NEF. Assume that we obtain an additional image dataset in another photographing session and want to select only effective pixels in it. The NEF model rendering generates a 3D photometric error map $\hat{\mathbf{e}}' := \mathcal{R}(\mathbf{x}', \mathbf{W}_{\mathbf{e}'})$ by indicating the pixel locations of the original NeRF model that would result in an erroneous rendering at novel views, $\mathbf{x}'$. As discussed in Section 4.1, our error maps combine grayscale backgrounds with errors highlighted in pseudo-colors for occlusion-aware NEF training (Fig. 2.1d). Only pixels identified as erroneous affect the follow-up NeRF training. Such pixels $e_j \in \hat{\mathbf{e}}'$ can be selected as having a certain level of error, i.e., e_j must be larger than a threshold (Fig. 2.2c).

Follow-up NeRF training with additional viewpoints. Given the filtered dataset, $\mathbf{D}'$, there are three choices in the subsequent training of the NeRF model (Table 1). We compute with the training strategy $\mathbf{W}_2 := \mathcal{T}(\mathbf{D} \oplus \mathbf{D}', \mathbf{W}_0)$ for the best performance.

4 EVALUATION

Overview. We implemented the method with Nerfstudio v0.3.3 and a defacto-standard NeRF model, Nerfacto (Tancik et al., 2023). We evaluate how different colormap encoding strategies influence the er-

[1]Polycam: `https://poly.cam/`



(1) Validated input error maps for NEF. (2) Post filtering process for (d) `E:Vir+B:G`.

Figure 2: Error maps encoding for NEF and its filtering.

ror map visualization in NEF of the NeRF representation. We also compare the best choice with a full dataset training and another post-hoc approach (Goli et al., 2024) to show how well our approach performs the follow-up NeRF training.

4.1 Validating Error Encoding for NEF

Color maps. We empirically design f_{e} to reasonably describe 3D photometric errors via NEF using color maps. In total, we evaluate four maps to find the best choice (Fig. 2.1).

- `E:RGB`. We calculate L1-norm, $\mathbf{e}_i = |\hat{\mathbf{c}}_i - \mathbf{c}_i|_1$. Since NeRF is trained to minimize pixel-wise differences via the volume rendering, many regions appear very dark except for the edges and specular highlights (Verbin et al., 2022) (Fig. 2.1a).
- `E:RGB+B:M`. We assign transparency values, α_i, to dark pixels for non-existing points where the pixel errors are below the top 10% error values (Fig. 2.1b). Too many pixels could be masked out, leading to under-constrained NEF training.
- `E:Vir+B:M`. We apply pseudo-colors to the error values $\mathbf{e}_i$ to crank up the missing contrast in `E:RGB+B:M`. We chose the Viridis color space $f_{\mathrm{vir}}(|\mathbf{e}_i|_2)$ for high color variations (Fig. 2.1c) as a locally optimal solution that we found during our initial tests (Table 2).
- `E:Vir+B:G`. For better reasoning of occlusions and error rendering at novel views, we replace 'black' non-erroneous pixels with the original pixel colors in grayscale g_i (Fig. 2.1d). This representation lets NEF consider occlusions in the 3D scene, while still being able to separate gray pixels from colored pixels. Upon NEF rendering, we filter grayscale pixels to retrieve the encoded photometric error pixels only as in Fig. 2.2b. This is used to filter pixels by threshold (Fig. 2.2c).

To robustly distinguish Viridis and grayscale pixels, we first convert rendered colors into LAB colors. The LAB space is chosen due to its robust decoupling of chromaticity (a,b) from lightness (L). The color transfer is then performed by transforming the pixel values of the trained NeRF renderings to match the color distribution of the input images, $(L,a,b) \rightarrow (L,a',b')$ as $a' = \bar{\mu}_{a_{tgt}} + \bar{\sigma}_{a_{tgt}}(a-\mu_a)/\sigma_a$. The same applies to b'. The averages of the standard deviations $\bar{\sigma}_{tgt}$ and the mean values $\bar{\mu}_{tgt}$ are calculated from the input images. The luminance values are then set to zero to remove grayscale backgrounds, $(L,a',b') \rightarrow (0,a',b')$. The characteristic of clear color separation in the color space motivates using the Viridis colors. To recover the appropriate luminance, we find the best matching colors between the rendered LAB colors and the reference LAB colors in discrete steps, (L_k,a_k,b_k), as (L_k,a',b') for $\arg\min_k (1-(a',b')\cdot(a_k,b_k))$.

Dataset. We evaluated the four different input strategies using public datasets, Mip-NeRF 360 (Barron et al., 2022), Nerfstudio (Tancik et al., 2023), and Blender (Mildenhall et al., 2020) with 90:10 of train and evaluation splits. We created our ground truth by training NeRF for each scene with all available images and performed the calculations for the individual error map types in Sect. 4.1. Therefore, every type of error map has different ground-truth images.

Metrics. NEF renderings are evaluated at the viewpoints of the evaluation images using learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018) for rendered error maps and F-measure (Hand et al., 2021) for binarized images. We opt for these metrics because standard image metrics, such as peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM), are problematic for NEF. Given that many pixels are black, minor differences can lead to disproportionately large variations in these metrics, which do not accurately represent the behavior we observe. NEF's goal is not to visualize raw error, but to render a perceptually significant error volume in 3D.

Results. Table 2 shows that the second-best results are dataset dependent, though we conclude that

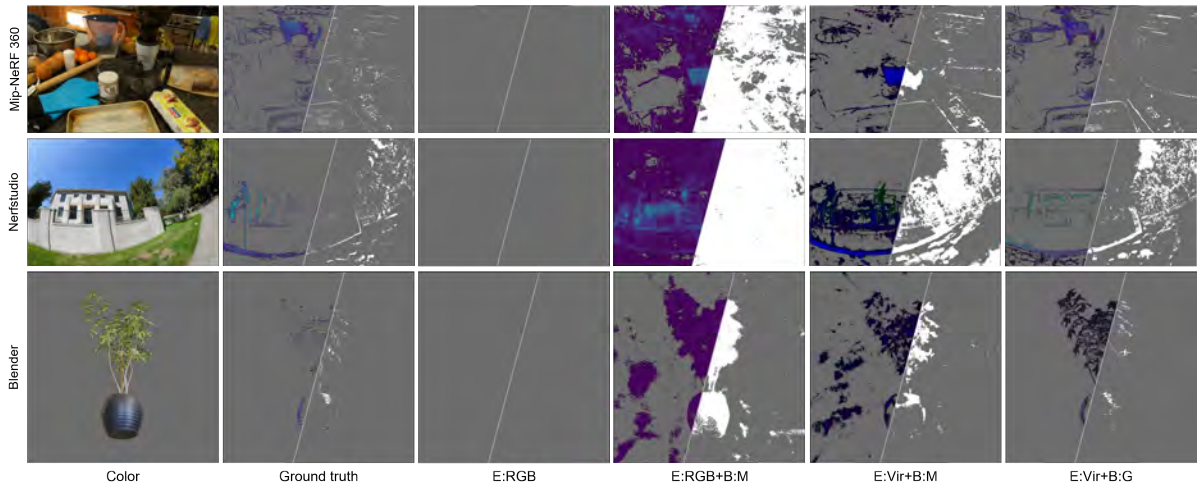


Figure 3: Qualitative comparisons of NEF rendering quality between different error pixel encoding strategies. `E:RGB` completely fails to reproduce any colors. Both `E:RGB+B:M` and `E:Vir+B:M`, which use transparency to exclude black pixels, reveal shapes and many inconsistent, erroneous pixels. The Viridis color space aids NeRF in curving better than RGB colors. Black pixels are replaced with gray pixels for better visibility.

Table 2: Quantitative comparison of NEF rendering quality between different error pixel encoding strategies in F-measure (↑) and LPIPS (↓)

F-measure	Mip-NeRF 360	Nerfstudio	Blender	LPIPS	Mip-NeRF 360	Nerfstudio	Blender
`E:RGB`	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	`E:RGB`	0.72 ± 0.11	0.77 ± 0.08	$\underline{0.25 \pm 0.05}$
`E:RGB+B:M`	0.19 ± 0.15	$\underline{0.32 \pm 0.20}$	0.08 ± 0.05	`E:RGB+B:M`	0.65 ± 0.04	0.62 ± 0.05	0.56 ± 0.11
`E:Vir+B:M`	$\underline{0.22 \pm 0.09}$	0.15 ± 0.11	$\underline{0.15 \pm 0.11}$	`E:Vir+B:M`	$\underline{0.55 \pm 0.07}$	$\underline{0.56 \pm 0.14}$	0.26 ± 0.05
`E:Vir+B:G`	$\mathbf{0.34 \pm 0.11}$	$\mathbf{0.32 \pm 0.14}$	$\mathbf{0.22 \pm 0.15}$	`E:Vir+B:G`	$\mathbf{0.41 \pm 0.07}$	$\mathbf{0.42 \pm 0.05}$	$\mathbf{0.14 \pm 0.03}$

`E:Vir+B:G` is the most robust and accurate in error map rendering. Fig. 3 qualitatively validates that NEF with `E:Vir+B:G` appears the most plausible. The grayscale background colors seem to provide more features on where to locate colors for errors and better represent occlusions. The Viridis colors lead to better training due to their vividness and their support for distinguishing the photometric error after NEF training partly mixes colors with grayscale backgrounds. Represented view dependency is best demonstrated in the supplemental video[2].

4.2 Validating Follow-up Training

We evaluate how the identified pixels can contribute to the final NeRF training. We use the Mip-NeRF 360 dataset (Barron et al., 2022) consisting of bounded and unbounded real scenes. We use images downscaled to $1/4$ of the original resolution (corresponding to "images_4"). Here, we evaluate how many ray samples we can reduce over different sizes of initial and additional datasets and how well the final NeRF training maintains the quality.

[2] `https://youtu.be/6hsq9Pn-OM4`

Datasets. To simulate the two-stage training, we split a scene dataset of N images into N_1, N_2, and N_{eval} for the initial training, follow-up training, and evaluation images. We follow the default Nerfstudio train-evaluation split N_{eval} at 10% of the total images. The remaining 90% of the images are divided into N_1 and N_2. Given this, we (1) train a NeRF model using N_1 color images, (2) train NEF with calculated N_1 error images, (3) identify erroneous pixels in N_2 color images, (4) perform a follow-up training for the NeRF model using the identified pixels, and (5) render the NeRF model at N_{eval} viewpoints and calculate image metrics. We test on three distinct configurations for the split between N_1 and N_2 with the remaining train images: $N_1 = 50\%$ and $N_2 = 50\%$; $N_1 = 20\%$ and $N_2 = 80\%$; and $N_1 = 5\%$ and $N_2 = 95\%$. This simulates 2x, 5x, and 20x increases in the effective dataset size for follow-up training with NEF.

Metrics. We calculate the image metrics, PSNR, SSIM, and LPIPS, for the NeRF models of steps 1 and 5 to assess improvements in the final NeRF training. We also measured the number of ray samples during the follow-up training, M.

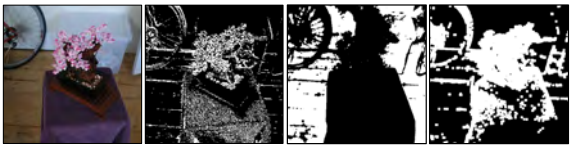
Table 3: Overview of Comparison Methods.

Method	Pruning Type	Error Prediction	Descriptions
Full Training	None	Not capable	Train on all rays, $\mathbf{D} \oplus \mathbf{D}'$. Baseline.
Direct Error	Static Mask	Capable with $\mathbf{D}'$	Predict pixel-wise errors only when the initial NeRF is precisely aligned with views.
FL-NeRF	Dynamic Mask	Not capable	Recursively select rays based on pure errors.
Bayes' Rays	Static Mask	Capable only with $\mathbf{D}$	Predict pixel-wise uncertainty at new viewpoints before visiting the site for $\mathbf{D}'$.
NEF	Static Mask	Capable only with $\mathbf{D}$	Predict pixel-wise errors at new viewpoints before visiting the site for $\mathbf{D}'$.

Methods. We compare the following five approaches. Table 3 summarizes their characteristics:

- Full training: The most naïve approach to performing the follow-up training is to use all pixels within the field of view, or to omit steps 2 and 3. Therefore, $M = P(N_1 + N_2)/N$, where P is the number of pixels in an image.
- Pixel pruning with Direct Error: This serves as a non-volumetric ground-truth-based post-hoc baseline. We calculate the L1 photometric error directly between the NeRF rendering ($\mathcal{R}(\mathbf{x}, \mathbf{W}_1)$) and the corresponding ground-truth image for all pixels in N_2. Pixels are selected based purely on the magnitude of this raw, image-space error, independent of any neural error field modeling.
- Pixel pruning with FL-NeRF (Zhang et al., 2023): This method differs from static mask approaches (NEF, Bayes' Rays) as it prunes rays progressively during every epoch to concentrate computation on erroneous areas. This serves as a baseline with in-process ray pruning.
- Pixel pruning with Bayes' Rays (Goli et al., 2024): It is the only post-hoc approach for uncertainty quantification like NEF. This approach defines a 3D voxel space in a scene and optimizes the voxels for uncertainty. We repurpose their uncertainty rendering for our pixel pruning purpose.
- Pixel pruning with NEF: NEF enables us to select pixels that would contribute the most to the follow-up training. We select pixels with top $t\%$ errors ($t = 10$ normalized error values by default). We perform Gaussian smoothing with a standard deviation $\sigma = 2$ over NEF renderings to account for pixel-level misalignment in rendered views and then binarize the image with the given threshold. We use binary images as additional mask images for our base system (Tancik et al., 2023), to identify which pixels to use for training. Therefore, $M = P(N_1 + rN_2)/N$, where $r \in [0,1]$ depends on t.

We use the same r for NEF and Bayes' Rays to ensure a fair comparison with the same number of pixels and rays for the follow-up training. We extract the uncertainty images from Bayes' Rays with Inferno's default sequential color map with monotonic lightness values. The lightness threshold was then calculated so that the number of pixels for N_2 was equal to that of the NEF masks. For all approaches, we iterated 3.0×10^5 times (the default value of Nerfstudio) for the first and follow-up NeRF training, and 5.0×10^2 times for NEF training. Fig. 4 is a visual comparison of selection masks.



(a) (b) (c) (d)

Figure 4: Comparison of selection masks at a $\mathbf{D}'$ viewpoint. (a) The original color image. (b) Mask image obtained from direct pixel error is more fine-grained. Note that (b) is obtained only when the view and the pixel-aligned NeRF rendering remain aligned during additional on-site dataset collection. (c) An integration of the Bayes' Rays (Goli et al., 2024) into our pipeline selects pixels mainly in flat areas, where depth information can be ambiguous. (d) In contrast, NEF, based on color-coded photometric errors, identifies visually wrong pixels, which are often around textured areas.

Results. Fig. 5 shows that every approach improves the rendering quality from the initial training, and all results are very competitive. While we found slightly larger variances that suggest stronger scene dependence in the NEF approach, the overall metric remains close to the full training in all metrics and all datasets. Follow-up training with NEF ray pruning performed comparably to FL-NeRF and to the ray pruning based on Direct Error.

This demonstrates that NEF's volumetric error encoding and ray pruning approach are effective at predicting and prioritizing error regions prior to obtaining new viewpoints, confirming that NEF successfully learns the true photometric error magnitude required for impactful ray selection. However, SSIM is generally lower in NEF and Bayes' Rays compared to Full Training, FL-NeRF, and Direct Error. It suggests they may introduce slight spatial misalignment or blurring in complex textural regions, likely due to relying on a pre-computed volumetric error rather than dynamic, pixel-accurate sampling. Bayes'

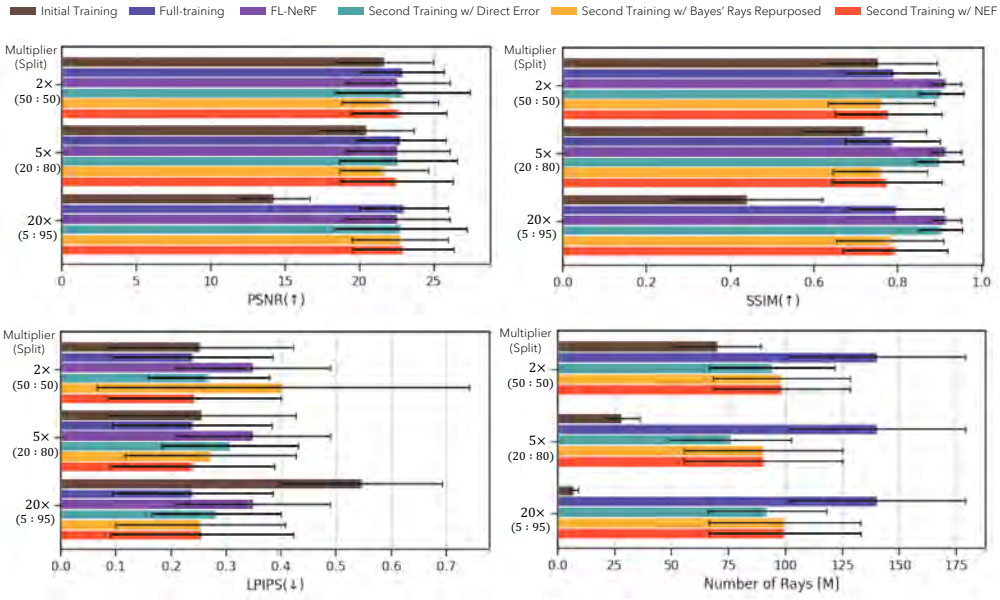


Figure 5: View synthesis quality (PSNR, SSIM, and LPIPS) and the number of rays for NeRF model training.

Rays obtain the worst average scores in PSNR and SSIM across all conditions compared to NEF, except in LPIPS for the dataset with 20× increase.

Fig. 6 shows qualitative comparisons. The follow-up training with NEF results in a visually convincing view synthesis, especially for specular highlights. The Bayes' Rays approach identifies pixels in flat areas as the source of greater uncertainty, while NEF finds erroneous pixels at edges and corners. NeRF models trained on only 5% of the initial data exhibit many fluctuating clusters, leading to predominantly noise-driven predicted masks.

While the number of rays selected with NEF is lower than that for the Full Training, the view synthesis quality with NEF is comparable. Furthermore, the NeRF model trained with NEF results in higher view synthesis quality than Bayes' Rays when using the same number of rays.

5 Conclusion

We presented neural error fields (NEF), a post-hoc approach to represent errors in NeRF training. We validated our choice of error encoding by comparing four possible approaches for encoding the error, and we evaluated its impact on the rendering quality and the ray filtering capability. NEF can represent view-dependent and occlusions by color coding errors. This allows us to predict and filter out unnecessary ray samples in subsequent training cycles. We demonstrate that follow-up training with NEF generates high-quality results with fewer rays than full training with all rays.

Future work may include estimating the next best view and visualization using NEF in the additional photographing session. NEF is a view-dependent and occlusion-aware error map representation. Therefore, it can calculate potential errors at given views by summing up the rendered error values.

ACKNOWLEDGEMENTS

This work was supported by the Alexander von Humboldt Foundation funded by the German Federal Ministry of Research, Technology and Space, German Research Foundation *DFG* (grant 528364066), and the JST BOOST, Japan Grant Number JPMJBS2409.

REFERENCES

Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., and Srinivasan, P. P. (2021). Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*.

Chai, J.-X., Tong, X., Chan, S.-C., and Shum, H.-Y. (2000). Plenoptic sampling. In *SIGGRAPH*, pages 307–318.

Franke, L., Rückert, D., Fink, L., Innmann, M., and Stamminger, M. (2023). VET: Visual error tomography for

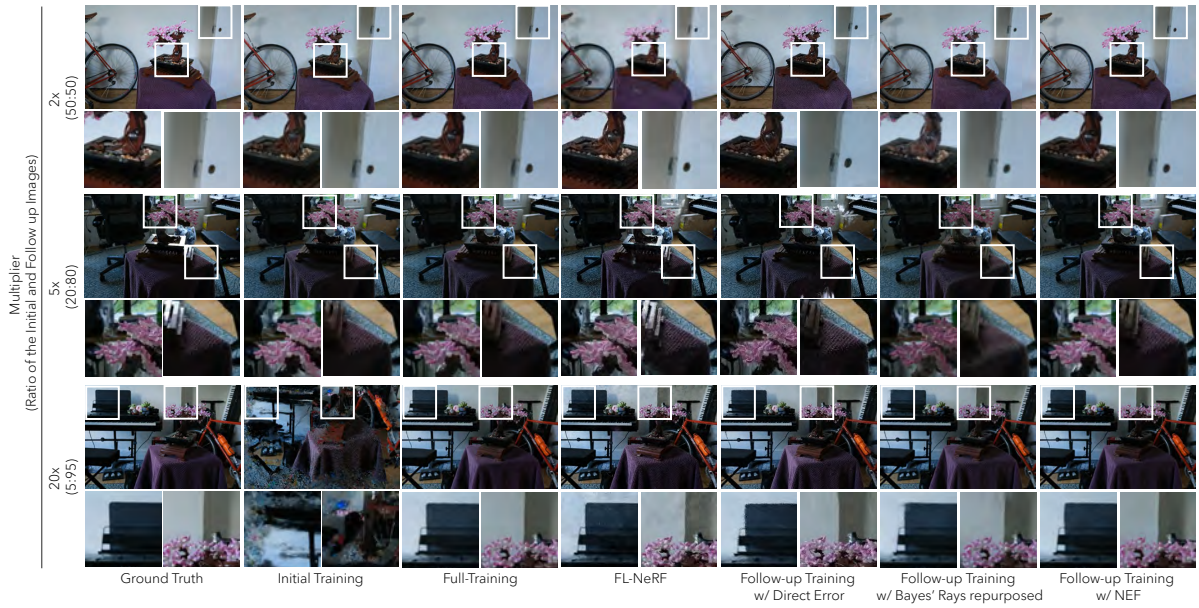


Figure 6: Qualitative results. 50% (top row), 20% (middle), and 5% (bottom) for initial datasets from three different views.

point cloud completion and high-quality neural rendering. In *SIGGRAPHA*.

Goli, L., Reading, C., Sellán, S., Jacobson, A., and Tagliasacchi, A. (2024). Bayes' Rays: Uncertainty quantification in neural radiance fields. In *CVPR*.

Hand, D. J., Christen, P., and Kirielle, N. (2021). F*: an interpretable transformation of the f-measure. *Machine Learning*, 110(3):451–456.

Ishikawa, R., Saito, H., Kalkofen, D., and Mori, S. (2023). Multi-layer scene representation from composed focal stacks. *IEEE TVCG*, 29(11):4719–4729.

Jin, L., Chen, X., Rückin, J., and Popović, M. (2023). NeU-NBV: Next best view planning using uncertainty estimation in image-based neural rendering. In *IROS*, pages 11305–11312.

Martin-Brualla, R., Radwan, N., Sajjadi, M. S., Barron, J. T., Dosovitskiy, A., and Duckworth, D. (2021). NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*.

Mildenhall, B., Srinivasan, P. P., Ortiz-Cayon, R., Kalantari, N. K., Ramamoorthi, R., Ng, R., and Kar, A. (2019). Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG*, 38(4):1–14.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*.

Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):1–15.

Philip, J. and Deschaintre, V. (2023). Floaters No More: Radiance Field Gradient Scaling for Improved Near-Camera Training. In Ritschel, T. and Weidlich, A., editors, *Eurographics Symposium on Rendering*. The Eurographics Association.

Schönberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *CVPR*.

Shen, J., Ruiz, A., Agudo, A., and Moreno-Noguer, F. (2021). Stochastic neural radiance fields: Quantifying uncertainty in implicit 3d representations. In *3DV*, pages 972–981.

Sünderhauf, N., Abou-Chakra, J., and Miller, D. (2023). Density-aware NeRF ensembles: Quantifying predictive uncertainty in neural radiance fields. In *ICRA*, pages 9370–9376.

Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., et al. (2023). Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH Conf.*, pages 1–12.

Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Yifan, W., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., et al. (2022). Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735.

Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J. T., and Srinivasan, P. P. (2022). Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *CVPR*, pages 5481–5490.

Warburg, F., Weber, E., Tancik, M., Hołyński, A., and Kanazawa, A. (2023). Nerfbusters: Removing ghostly artifacts from casually captured nerfs. In *ICCV*.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*.

Zhang, W., Xing, R., Zeng, Y., Liu, Y.-S., Shi, K., and Han, Z. (2023). Fast learning radiance fields by shooting much fewer rays. *IEEE TIP*, 32:2703–2718.