Daniel Rodrigues Perazzo

DirectVoxGO++: Fast Object Reconstruction Using Neural Radiance Fields

Universidade Federal de Pernambuco
www.cin.ufpe.br/~graduacao

Recife
2022

Daniel Rodrigues Perazzo

DirectVoxGO++: Fast Object Reconstruction Using Neural Radiance Fields

A B.Sc. Dissertation presented to the Centro de Informática of Universidade Federal de Pernambuco in partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering.

Area: *Visual Computing, Machine Learning*
Advisor: *Prof. Veronica Teichrieb (UFPE)*
Co-Advisor: *Prof. Luiz Velho (IMPA)*
Co-Advisor: *Prof João Paulo Lima (UFRPE)*

Recife
2022

FICHA

BANCA

I dedicate this work to my "besten Freunde" Nati and Line. I also dedicate it to my "brother" Giu, my family, and my love Malu.

ACKNOWLEDGEMENTS

Se fosse para ser realista, eu iria ter que fazer uma dissertação inteira falando sobre a quem eu dedico esse trabalho.

Primeiramente, gostaria de agradecer a minha família, especialmente minha mãe Sandra e meu pai Francisco, minha madrinha Rosa e minha afilhada Eloazinha. A convivência com eles e seus ensinamentos me transformaram na pessoa, profissional e pesquisador que sou hoje. Gostaria de agradecer também a meus tios, Sirleide e tio Sérgio, meus primos Silas e Samuel e minha prima Silvinha, vocês foram muito especiais para mim.

Aos grandes professores, tanto da escola quanto da faculdade, que passaram pela minha vida, deixo menção a apenas alguns porque a lista seria enorme: Jefferson Goés, Eduardo França, Heron Andrade, André Paegle, Lula Couto, Armando Cavalcantti, José Américo, Divanilson, Sílvio Melo, ACM, CABM, Daniel, Diego Nehab e muitos outros. Com certeza esses professores me ajudaram muito na minha trajetória.

Gostaria de agradecer também aos amigos e companheiros que eu fiz nessa minha trajetória, da epóca do colégio eu agradeço a Caio, Anna, Ravena, Leandro e muitos outros. Agradeço muito aos meus amigos da faculdade, em especial ao grupo formado por Lucas Ambrósio, Pedro Coutinho, Victor Miguel, Kinho, Zilde, Victor Hugo, Victor Ximenes, Zenio e muitos outros. Esse grupo me enriqueceu tanto nas conversas descontraídas quanto nas discussões sobre a complexidade de uma busca em árvore binária. Agradeço também aos meus amigos do grupo Maracatronics: Juju, Leo Trevisan, Caio e muitos outros.

Para finalizar, agradeço a meus amigos Gabo, Gigante, Leonardo, Rocha e Júnior, sempre vou lembrar das noites ouvindo música enquanto discutimos geopolítica mundial.

Esse trabalho não poderia ser feito, também, sem o grande apoio dos meus melhores amigos: Nati, Aline e Giu. Essas três pessoas, embora não tenham ajudado a escrever o trabalho em si, contribuíram imensamente na minha formação. Dedico esse trabalho especialmente a cada um de vocês. Amo muito vocês. Dedico também a minha namorada, Malu, alguém que eu conheci tão recentemente e que já deixou uma marca tão grande em mim, eu te amo.

"Last but not least", agradeço a todos os cientistas e colegas com quem eu tive o prazer de trabalhar. VT e toda a equipe do Voxar Labs por terem me dado uma casa onde eu pude pesquisar de uma maneira muito divertida, ao João Paulo (ou Jonga) por ter me orientado durante quase toda minha estadia de 3 anos no Voxar e também neste trabalho, e ao Luiz Velho, por ter respondido um email de um estudante localizado à 2,500 km de distância, ensinar os fundamentos de CG e visão computacional e me orientar e apoiar nessa caminhada. Agradeço também aos outros orientadores que eu tive nessa vida: Alana, Joma e Rafael. Vocês foram muito mas muito importantes mesmo. Com certeza essas orientações me ajudaram a trilhar meu caminho como pesquisador.

E com isso, sem mais delongas, vamos começar esse trabalho.

ABSTRACT

One of the most classic problems in computer vision, 3D reconstruction aims to build a 3D model of an object or scene given a series of views. In recent years, new reconstruction techniques based on Neural Radiance Fields (NeRFs) have created new forms to model objects instead of the traditional mesh and point cloud-based representations, allowing for more photorealistic rendering. However, these techniques were too slow to be used in practical settings, taking in the range of hours in high-end GPUs. Due to these limitations, new techniques have been created for fast reconstruction of scenes, such as DirectVoxGO. Alongside this limitation, one issue with NeRFs is that they were initially unable to separate the foreground from the background and had problems with 360° until the emergence of new techniques such as NeRF++. Our method extends DirectVoxGO, which is limited to bounded scenes, with ideas from NeRF++ and incorporates elements from a neural hashing approach employed by other works. Our technique improved photorealism compared with DirectVoxGO and Plenoxels on a subset of the LF dataset on average in at least 2%, 8% and 8% for PSNR, SSIM, and LPIPS metrics respectively, while also being an order of magnitude faster than NeRF++.

Keywords: Neural Radiance Fields, Computer Vision, Computer Graphics.

RESUMO

Um dos problemas mais clássicos em visão computacional, reconstrução 3D tem o objetivo de construir um modelo 3D de um objeto ou cena dado uma série de vistas. Nos últimos anos, novas técnicas de reconstrução baseado em Neural Radiance Fields (NeRFs) conseguiram criar novas formas de modelar objetos, substituindo formas tradicionais de modelagem 3D como nuvem de pontos e malhas. Essa nova representação permite uma renderização mais fotorealista. Contudo, essas técnicas demoram muito tempo, na faixa de horas, para aprender uma cena mesmo em GPUs potentes. Por conta dessas limitações, novas técnicas foram criadas para reconstruções rápidas, como o DirectVoxGO. Além dessa limitação de tempo, um problema com NeRFs é que, inicialmente, essas técnicas não separam o "foreground" do "background" além de possuírem problemas em cenas 360°. Esse tipo de problema foi, em parte, solucionado com o desenolvimento da técnica NeRF++. Dado este cenário, nós desenvolvemos um aprimoramento do DirectVoxGO com base nas idéias introduzidas no NeRF++ além de incorporar idéias de neural hashing proposta em outros trabalhos. A nossa técnica, batizada de DirectVoxGO++, conseguiu melhorar o fotorealismo em comparação com o DirectVoxGO e o Plenoxels em um subconjunto do LF dataset, em média e no mínimo em 2%, 8% e 8% para as métricas PSNR, SSIM, e LPIPS respectivamente e, ainda, sendo mais rápido do que o NeRF++ por uma ordem de magnitude.

Palavras-chave: Neural Radiance Fields, Visão Computacional, Computação Gráfica.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1

INTRODUCTION

One of the biggest challenges in computer vision and computer graphics is a seemingly simple one: “how can we reconstruct a 3D object?” [19]. Finding efficient ways would allow many different applications in industries as diverse as medicine [30], civil engineering [29], and even architecture and historical preservation [23]. As a recent example of an innovative way these types of technology can be used, a team of Brazilian researchers recently performed a 3D reconstruction of a mummy severely damaged in the National Museum of Brazil fires. Now, the mummy can be visualized in virtual reality (VR), as in Figure 1, helping to preserve it for future generations [1].

Figure 1: Virtual mummy visualized in VR, image from the IMPA website [1]

However, a more exciting and constantly pursued goal by researchers is to perform 3D reconstruction, and realistic rendering with the exclusive use of RGB images [2]. The use of cheap RGB cameras for reconstruction could significantly decrease costs and make it easily accessible. They are present in many smartphones since special sensors, such as RGB-D cameras, are more expensive.

Researchers started using classical computer vision and optimization methods and techniques to perform this task, using projective geometry to model the world. For example, one of the most famous techniques in this field is COLMAP [47], which enables significant reconstruc-

tion with a few views. 3D models can be obtained in a wide variety of presentions and formats, such as meshes, point clouds, and even volumetric representations [53].

However, like the entire field of computer vision, deep learning profoundly changed 3D reconstruction and led to new and exciting methods [18]. Some great techniques emerged in this setting, such as Pixel2mesh [58] and Pix2Vox [62]. As impressive as those techniques are, they still use traditional world representations, such as meshes and voxels.

A new research area that is quickly expanding is the use of 3D implicit representations [46]. The key idea behind these techniques is to represent the object or scene as a neural network instead of using traditional representations like meshes or point clouds. Some of the first and most impressive works on this topic are DeepSDFs [42], and Occupancy Networks [31].

Being one of the newest types of an algorithm for 3D reconstruction, Neural Radiance Field (NeRF) [33] techniques experienced an explosion in the number of works published in a relatively short period [9]. These methods, based on the classic theory of light fields [26], allow the capture and rendering of scenes in an impressive photorealistic manner. Figure 2 shows the basic NeRF pipeline.

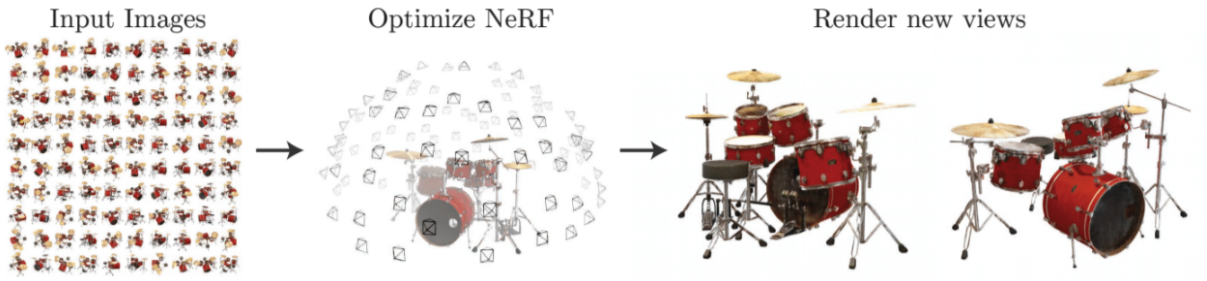

Figure 2: NeRF, basic setup, image from *Mildenhall et al.* [33]

However, the original NeRF technique has a few downsides. For instance, the method is very computationally expensive. The technique takes hours to run on high-end GPUs [33] and has a large memory footprint when learning a single scene. Additionally, the original NeRF only allowed the capture and representation of an entire scene, not being possible to separate, for example, foreground objects from the background, which would be interesting for 3D reconstruction applications. And, as shown in NeRF++ [70], the original NeRF has problems with 360° settings. To counter this issue, in this work, we present a technique that, after receiving a set of images and their respective intrinsic and extrinsic parameters, obtains a 3D model of the object of interest and background. In addition, we aim to make our method efficient in terms of memory usage and scene optimization time.

1.1 OBJECTIVES AND METHODOLOGY

As previously stated, the general objective of this work is to provide a new NeRF-based object reconstruction technique that is efficient both in terms of memory usage and learning

speed and allows the separation of foreground and background in 360° scenes, scenes in which the object does a rotation across an object of interest..

The following hypothesis statements are examined throughout the remainder of this work:

- **h1**: Creating a different pipeline for the foreground and the background can improve DirectVoxGO performance in 360° scenes;
- **h2**: Combining DirectVoxGO with the neural hash function from *Muller et al* [35] can significantly improve its results.

And finally, the specific goals of this work are:

- Provide a literature review on the development of NeRF-based methods;
- Provide an explanation and analysis of the key points of both the original NeRF techniques and the derivatives we used to build our technique;
- Define and develop an improvement of DirectVoxGO for 360° scenes and extraction of a 3D model of an object of interest;
- Perform qualitative and quantitative evaluations of our method on widely-used datasets. In our case, we will test on the Light Field (LF) [69] dataset and compare it with other works such as the original DirectVoxGO [52] and Plenoxels [66].

2

BACKGROUND CONCEPTS

In this chapter, we review some concepts that will be helpful during this document and review some of the theoretical background essential to our research.

2.1 3D RECONSTRUCTION

3D reconstruction can be seen as the inverse problem of rendering. When you render, you have the graphical object and want to obtain the corresponding images. In 3D reconstruction, you want to generate a model from the images. Researchers studied this problem using many techniques from photogrammetry [53]. In this type of setting, researchers use techniques borrowed from projective geometry and similar mathematical constructions, estimating camera parameters (both extrinsic and intrinsic parameters) and, next, trying to estimate 3D points from the 2D images using multi-view stereo reconstruction techniques [19]. One of the most used techniques in this field is COLMAP [47], which utilizes classical optimization techniques to extract camera parameters and a point cloud from a set of images. In Figure 3 we can see a reconstruction generated by the COLMAP process.

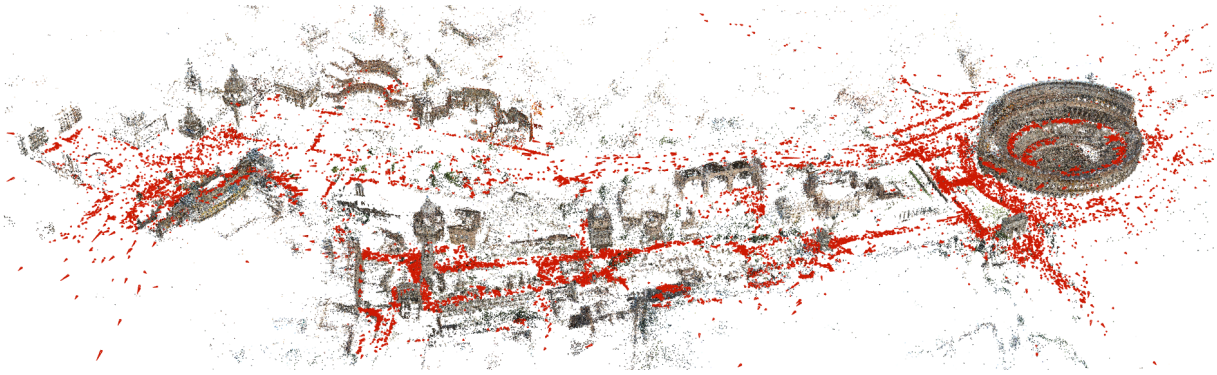

Figure 3: COLMAP point cloud reconstruction, image from [47]

2.2 VOLUME RENDERING

In computer graphics, rendering and representing structures such as fog, clouds, and smoke can be difficult using primitives such as meshes. [13] Due to this fact, researchers have created a class of techniques made to render particles represented as discrete 3D point sets, called volume rendering [10]. Nowadays, volumetric rendering is used in many applications, from cloud effects in Disney films [14] to tomography [37]. In Figure 4 we can see an image from the Disney film Moana [36] that uses volume rendering to create the antagonist of the film.

Figure 4: Volume rendering example, image from *Fong et al.* [14]

2.3 LIGHT FIELD

The light field is a modeling technique representing incoming lights as a vector function, similar to how physics models electromagnetic fields. In fact, in the mid-19th century, Michael Faraday was of the first researchers to conjecture that light could be modeled as a field [12]. Later, at the beginning of the 20th century, physics works, such as the ones by *Gershun et al.* [15], and *Moon et al.* [34], would mathematically model light as a field. Finally, in the late 90s, after the development of digital computers and computer graphics, both *Levoy et al.* [26], and *Gortler et al.* [16] suggested using light-field formulations and the 4D light field for image-based rendering (IBR). Investigations on capturing and manipulating the light field would lead to the development of a multi-camera array for the capture of light fields [57], and even a light-field microscope [27]. However, in this area, one watershed moment was the development of light-field cameras that enabled the easy capture of light fields [38]. These cameras use micro-lens arrays, which enable the camera to take photos from multiple views at once and capture the many light rays that enter the camera aperture, as seen in Figure 5. Moreover, with the acquisition of light fields, it is possible to perform image processing, e.g., changing the focus in the photos.

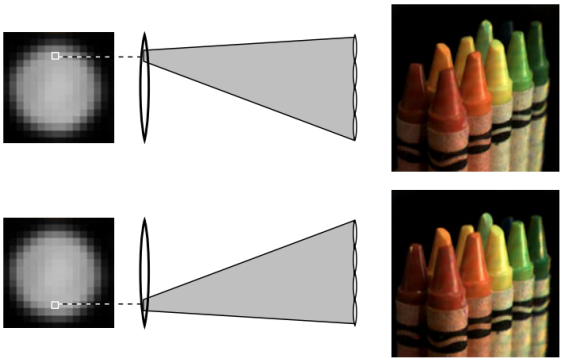

Figure 5: Different view points taken from the same light field, note the parallax on both images, image from *Ng et al.* [38]

2.4 NEURAL NETWORKS FOR IMAGE-BASED RENDERING

IBR is the technique for modeling and rendering that uses images instead of geometry as primitives [48]. In the last few years, many techniques tried to perform novel view synthesis of scenes given a few views. Many of these techniques used neural networks, and representations such as multi-plane images (MPIs) [51, 72].

Although these techniques create impressive novel views given a few input images, they sometimes generate artifacts from the MPI formulation. Due to this, *Mildenhall et al.* [32] developed a new technique for view synthesis using ideas from light-field rendering, as shown in Figure 6. Although these techniques obtain excellent results and great photorealism, they do not output geometry like 3D reconstruction techniques.

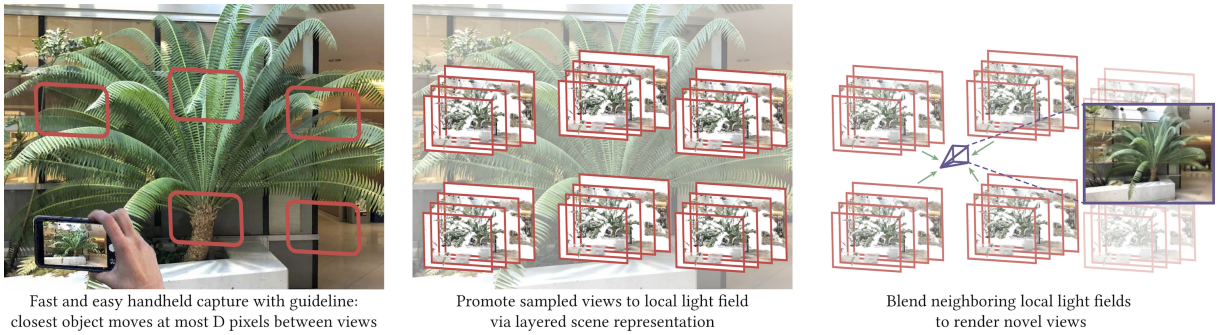

Figure 6: Illustration of the light field reconstruction technique by *Mildenhall et al.*, image from *Mildenhall et al.* [32]

2.5 DIFFERENTIABLE RENDERING

One exciting approach to performing 3D reconstruction would be to give an initial 3D model if we could render it and compare the synthesized images with the ground truth. Then, we would optimize the 3D model given the difference between the ground-truth images and the

rendered images. However, a significant problem with this approach is that traditional rendering algorithms are non-differentiable. Thus, gradients can not propagate from the images to optimize the parameters. Thus, many techniques have been created to render different graphical objects, such as meshes, point clouds, voxels, and, as we will see later, implicit representations. These techniques enabled new optimization and 3D reconstruction algorithms to be devised and created, as exemplified by Figure 7. For more details, the reader is referred to an survey on this area by *Kato et al.* [22].

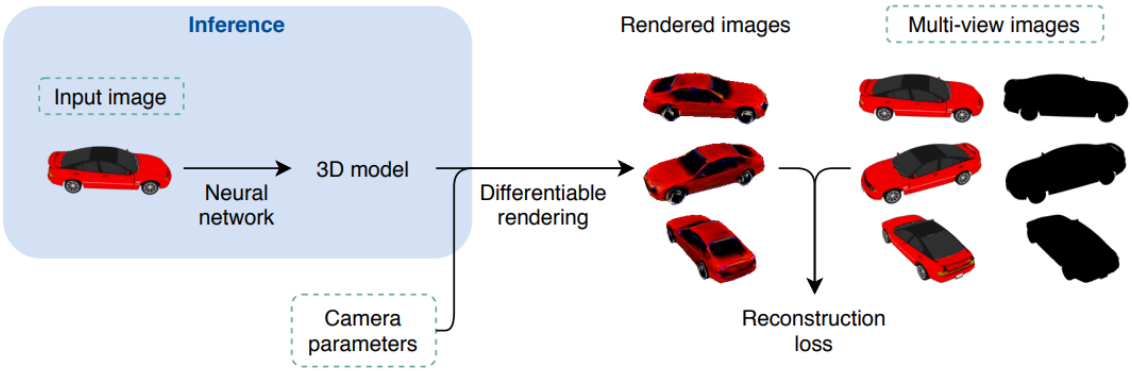

Figure 7: Illustration of how we can use a differentiable rendering pipeline to obtain a 3D model of an object, image from *Kato et al.* [22]

2.6 NEURAL IMPLICIT REPRESENTATIONS

Given a set, we can use implicit and explicit definitions to discover which elements are in it. For example, take the unit circle. Given a parameter $\theta \in [0, 2\pi]$ and $\mathbf{x} \in \mathbf{R}^2$, we can write all the points in the unit circle by the next equation:

$$\mathbf{x} = (\cos\theta, \sin\theta). \tag{2.1}$$

When we vary θ, we can get different points in the unit circle. However, we can also use an implicit representation where, given any point $\mathbf{x} \in \mathbf{R}^2$, it is in the unit circle if $\mathbf{x} = (x, y)$ respects the following equation:

$$x^2 + y^2 - 1 = 0. \tag{2.2}$$

Implicit representations can be instrumental whenever we have various points and want to know if they are in a set. More generally, we can define an implicit function as a function $f(.)$ and a point $\mathbf{x} \in \mathbf{R}^n$ such that $\mathbf{x}$ is in the defined set if and only if:

$$f(\mathbf{x}) = 0 \tag{2.3}$$

Implicit representations are used in computer graphics applications, such as SDFs [40].

Neural implicit representations use the fundamental idea that a neural network can work as the function $f(.)$

Neural implicit representations use implicit representations popular in computer graphics. Some techniques such as Occupancy Networks [31], and DeepSDFs [42] used implicit formulations, replacing the traditional functions used in other implicit formulations by neural networks. An illustration of DeepSDF is provided in Figure 8. Researchers even studied specific neural networks for these tasks, such as SIREN [49]. Later, new models of implicit representations were constructed with NeRFs [33]. A more complete survey is presented by *Schirmer et al.* [46].

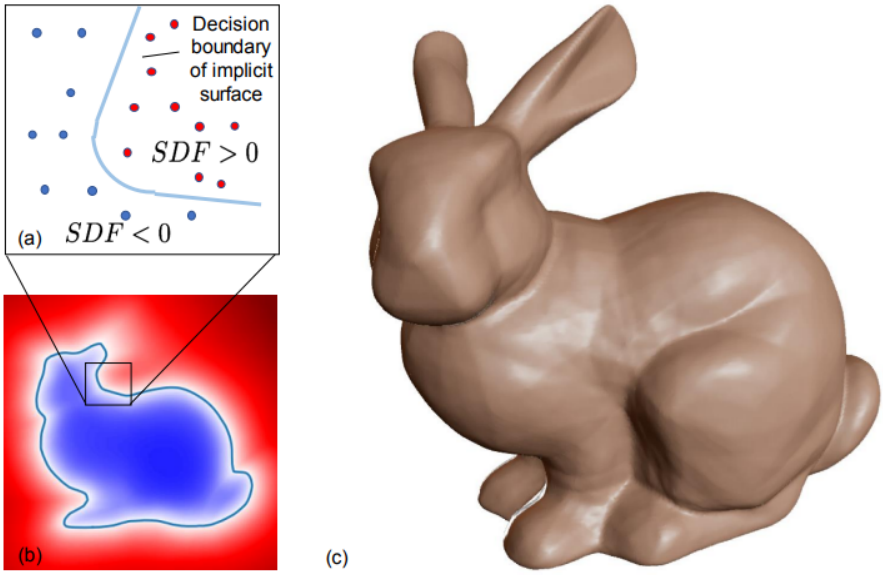

Figure 8: Representation of the DeepSDF technique. In **(a)** we have the implicit curve defined by SDF, in **(b)** we have the cross-section of the SDF (signed-distance curve) and in **(c)** we have the rendered bunny, image from *Park et al.* [42]

3

RELATED WORK

This chapter reviews the many works on NeRFs, including the original [33] technique, its ramifications and improvements, and, finally, the many techniques which tried to speed up NeRFs.

3.1 THE ORIGINAL NEURAL RADIANCE FIELDS

Generally, many practitioners' approaches separate physically based rendering (PBR) and IBR in computer graphics. Both of these fields have been around for some time and have been used in various forms. Nowadays, we have seen staggering progress in the IBR domain, from 3D photos based on MPIs [72] to light fields [21]. These developments enabled taking 3D photos from a single shot on many consumer devices [24].

However, these techniques present limitations, such as relatively low photorealism and not much expressivity. And they are also not capable of acquiring both geometry and appearance. [33]. Due to this, *Mildenhall et al.*[33] proposed NeRF, an IBR technique that encodes the scene as an MLP (Multi-layer Perceptron) and synthesizes novel views using an approach based on volumetric rendering. This approach is similar to previously discussed deep implicit methods. However, the original formulation only allowed for simple shapes.

Due to this, the authors encoded a 5D radiance field as an MLP and performed the consequent optimization. The authors base their technique on classic volumetric rendering, which is trivially differentiable, to perform the rendering. Thus, the technique enables high-quality view synthesis results from a series of images from a sparse set of camera views. In Figure 9 we see a basic representation of this pipeline.

However, NeRF is not without its flaws. Despite the impressive quality of the resulting views, it nonetheless has some drawbacks that many other authors decided to improve. Some of the most popular is the time it takes to optimize a single scene (which can be of the order of days on Nvidia RTX GPUs), the amount of memory it takes, and the speed of rendering a single scene.

Due to these limitations, many different variations and improvements on NeRF have been proposed. Here, we will only highlight the techniques that seem more in line with our

research, which will be techniques for rendering NeRFs photo-realistically and with fast training.

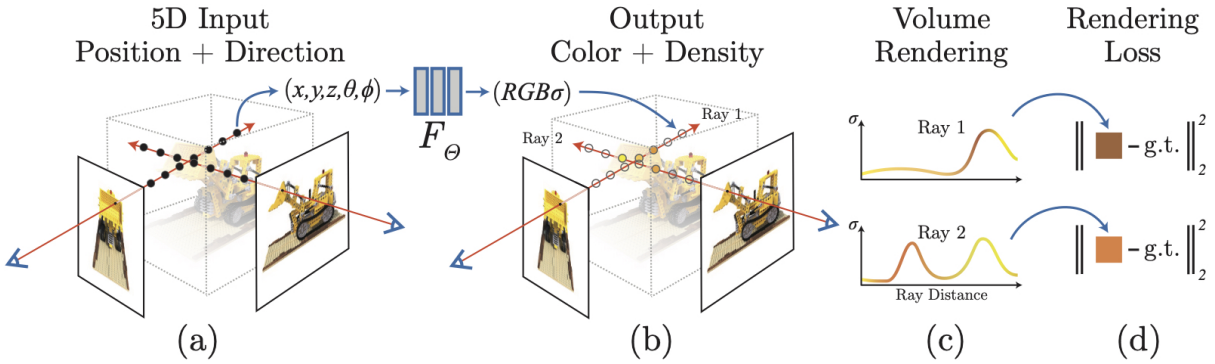

Figure 9: Representation of the NeRF pipeline. In **(a)** we have the sampling of the points in the rays, along with their directions represented by the angles θ, ϕ, in **(b)** we pass this input to a MLP, which outputs a color and density value, in **(c)** we use volumetric rendering to compute the color of the ray and in **(d)** we use a photometric loss to optimize the MLP, image from *Mildenhall et al.* [33]

3.2 IMPROVEMENTS TO THE ORIGINAL PIPELINE

Many authors proposed improvements to the quality of the synthesized views while at the same time diminishing some of the requirements for doing synthesis [60][65][25]. Some authors developed alternative approaches to the original NeRF formulation [64] while others tried to improve the original NeRF technique. From those, we have, for example, NeRF++ [70]. The original light-field rendering models have 4D, but the formulation of NeRF is 5D. Due to this, the original NeRF can create a shape-radiance ambiguity that may lead to failures. Generally, this does not happen because of a structural regularization of the NeRF MLP architecture. However, this can incur problems for large-scale unbounded scenes. NeRF++ proposes using NeRFs for foreground and background using an inverse sphere parameterization. We show an illustration of the background segmentation of NeRF++ in Figure 10.

Other techniques, such as Mip-NeRF [3], use an approach similar to mipmapping [61] and reduce aliasing artifacts in the final result. Later, researchers extended their approach with phenomenal results in Mip-NeRF 360. This technique uses a non-linear scene parameterization, online distillation, and a distortion-based regularizer [4].

(a) NeRF++ prediction (b) predicted foreground (c) predicted background

Figure 10: Composition of NeRF++ result, image extracted from NeRF++ paper [70]

Other researchers created methods to create compositional NeRFs, which can separate multiple objects from the background, being even more capable than NeRF++, and even allowing separate object manipulation. Some can perform this task with multiple objects but with the need of object masks [41] while others do not need masks [63]. The field of NeRFs is still rich with new developtments and novel applications, from style transfer [8], text-guided object generation [20] to even the creation of an entire city with NeRFs [54].

3.3 FAST TRAINING

One of the biggest problems of NeRFs, as mentioned previously, is their lengthy rendering and training times, which caused many techniques to aim at speeding up rendering. First, some of these adapted traditional computer graphics techniques such as plenoctrees [67]. Other techniques explored parallelization using much smaller NeRFs instead of only one [45]. However, other approaches for speeding up rendering have been proposed, such as querying the ray itself [50] to perform the analytical integration instead of computing it by sampling along the ray [28].

However, other researchers are currently discovering ways to speed up the training process. Initially, some researchers tried using techniques from meta-learning [55] to give a pre-initialized set of weights. The authors used fairly common meta-learning algorithms such as Reptile [39]. We give an illustration of their pipeline for the case of 2D images in Figure 11.

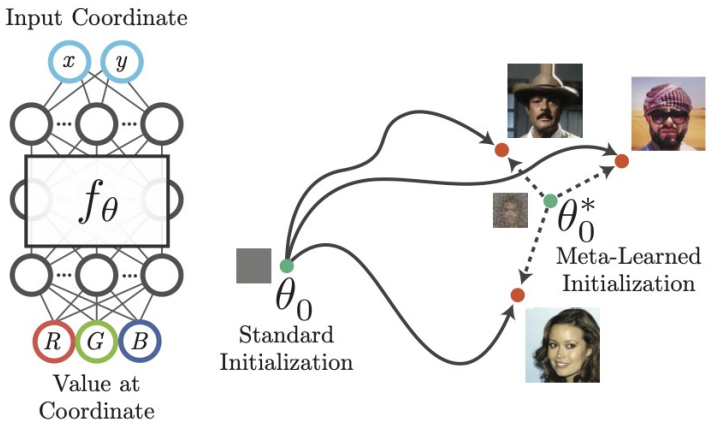

Figure 11: Basic idea of Learned Initializations for 2D images, having a meta-learned prior can greatly speed-up the optimization proccess for arbitrary images, figure from *Tancick et al.* [55]

Another approach was using CNNs to extract features from the images. This way, the technique could use image features as prior for the MLP. This approach was taken in both PixelNeRF [68] and MVSNeRF [7]. We provide some examples of PixelNeRF's output in Figure 12.

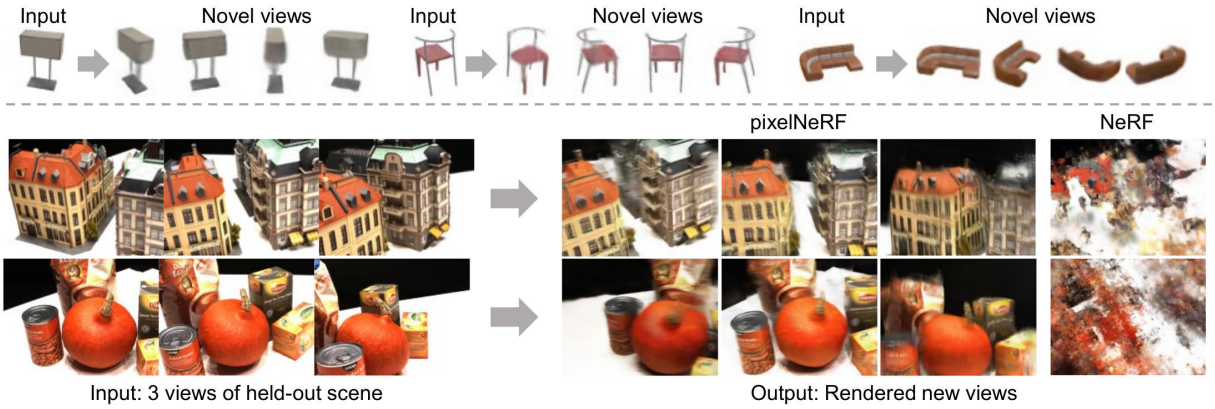

Figure 12: Example from PixelNeRF, their technique enables a NeRF represetation and novel view-synthesis from few input images, figure from *Yu et al.* [68]

Moreover, Instead of using meta-learning or convolutional features to speed-up training, recent approaches aim to shift the inference function to a faster training procedure. For example, both DirectVoxGO [52] and Plenoxels [66] use a voxel-based method instead of a pure MLP that has been traditionally associated with NeRFs. However, their approach assured similar results to the original NeRF technique with training times on the scale of minutes instead of hours. We provide an illustration of their technique in Figure 13. In another similar approach, TensoRF [6] applies tensorial decomposition to leverage a speed-up, with also a decrease in the memory footprint compared to previous methods.

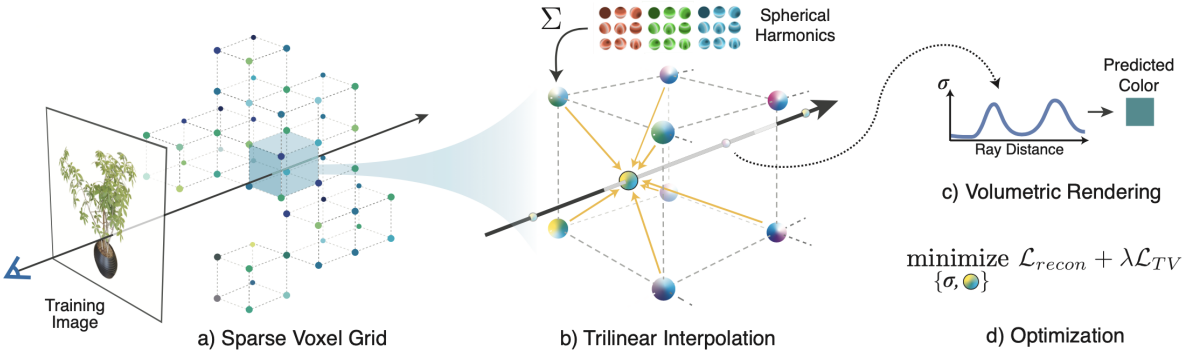

Figure 13: Basic pipeline of Plenoxels technique, in **(a)** they create a ray that passes through a voxel grid, in step **(b)** their technique uses trilinear interpolation of spherical harmonics coefficients to get the color and density of the sampled point and, finally, in **(c)** they perform volumetric rendering to compute the color of the ray and perform an optimization based both on a photometric loss and a TV regularization, image from *Yu et al.* [66]

Recently, neural-hashing-based techniques [35] have managed to do a speed-up that made them take seconds in settings where the original NeRF took hours. On a negative note, this new technique has a very high memory requirement. Finally, as a footnote, there has also been some research in creating a specialized system-on-a-chip (SoC) core for NeRFs, such as the ICARUS architecture [44].

4

DIRECTVOXGO++

In this chapter, we detail our approach. As stated previously, the idea is to augment the DirectVoxGO technique [52] with ideas from NeRF++ [70] and the neural hashing encoding from *Muller et al.* [35].

To better explain our technique, we first describe the original NeRF. Next, we present the original DirectVoxGO, and, finally, we provide a detailed explanation of our improvements compared to the original DirectVoxGO.

4.1 ORIGINAL NERF

The key idea of NeRF [33] is that it models the scene as a radiance and opacity field. To do this, the authors use a multi-layer perceptron we define as **MLP** that receives as input the position of a point $\mathbf{p} \in \mathbf{R}^3$ and a unitary view direction vector $\mathbf{d} \in \mathbf{R}^3$ and outputs both a color $\mathbf{c} = (r,g,b)$ and a density $\sigma \in \mathbf{R}$. The authors then used volume rendering to compute image values. Formalizing, if we want to render a view, given the camera parameters, we estimate a number of rays by computing the origin $\mathbf{o}$ and direction $\mathbf{d}$ of each ray $\mathbf{r}(.)$. With this, any point in the ray can be parametrized by $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with $t \in \mathbb{R}$. To limit the number of samples in the ray, the authors of NeRF used bounds based on the points obtained with COLMAP. Using the ray parameterization $\mathbf{r}(.)$, we can sample a number N of points $\mathbf{p_i}$ from the ray and, using the classic volume rendering equation [10] we can estimate the color of the ray.

For each sampled point $\mathbf{p_i}$, and also using the unitary viewing direction vector $\mathbf{d}$, we compute $\mathbf{c}$ and σ by

$$\sigma_i, \mathbf{c_i} = \mathbf{MLP}(\mathbf{PE}(\mathbf{p_i}), \mathbf{PE}(\mathbf{d})), \tag{4.1}$$

where $\mathbf{PE}(.)$ is a positional encoding, a technique introduced in their paper [33] which aims to stimulate the **MLP** to learn high-frequency features. To perform this positional encoding, for each point or vector $\mathbf{v}$, we perform the following computation, once chosen an integer $L \in \mathbf{N}$:

$$\mathbf{PE}(\mathbf{v}) = (\sin(2^0\pi\mathbf{v}), \cos(2^0\pi\mathbf{v}), ..., \sin(2^{L-1}\pi\mathbf{v}), \cos(2^{L-1}\pi\mathbf{v})) \tag{4.2}$$

Following the volume rendering approach, given a step size of the ray $\delta_i \in \mathbf{R}$, we compute an $\alpha_\mathbf{i}$ value, which is the probability that the ray will terminate at this point. This α_i is computed by:

$$\alpha_i = \mathbf{alpha}(\sigma_i, \delta_i) = 1 - \exp(\sigma_i \delta_i). \tag{4.3}$$

Finally, by making this computation along each ray, we estimate its color $\mathbf{C}(\mathbf{r})$ for each $\mathbf{r}$. But first, we will compute opacity weights T_i for each $i \in [0, N+1]$:

$$T_i = \prod_{j=1}^{j=i-1} (1 - \alpha_j). \tag{4.4}$$

Now, with the color value $\mathbf{c_i}$ for each of the N sampled points in the ray, we can then obtain $\mathbf{C_{pre}}(\mathbf{r})$.

$$\mathbf{C_{pre}}(\mathbf{r}) = \sum_{i=1}^{i=N} T_i \alpha_i \mathbf{c_i}. \tag{4.5}$$

Finally, we can compose with a pre-defined background color $\mathbf{c_{bg}} = (r, g, b)$, where we multiply it by the opacity corresponding to the background T_{N+1}. Doing so will enable us to obtain the final $\mathbf{C}(\mathbf{r})$:

$$\mathbf{C}(\mathbf{r}) = \mathbf{C_{pre}}(\mathbf{r}) + T_{N+1}\mathbf{c_{bg}}. \tag{4.6}$$

Although we wrote this equation in discrete form, it is only a discretized version of an integral, which makes it differentiable. Based on this approach, the authors optimized the MLP using a simple photometric loss for each ray $\mathbf{r_i}$, comparing it with the corresponding observed pixel color corresponding to that ray $\mathbf{C_{pix}}(\mathbf{r})$ while iterating in the set of all rays $\mathscr{R}$:

$$\mathscr{L}_{photo} = \frac{1}{||\mathscr{R}||} \sum_{\mathbf{r} \in \mathscr{R}} ||\mathbf{C_{pix}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}))||^2 \tag{4.7}$$

The authors then optimize the neural network parameters using a gradient-descent-based approach.

4.2 ORIGINAL DIRECTVOXGO

In this section, we give an overview of the DirectVoxGO [52] technique without our proposed modifications. The key idea of DirectVoxGO is to use a voxel grid coupled with a two-stage training procedure that aims to find a coarse model for the basic geometry of the scene. This coarse model allows for a tighter bounding box during the fine-training stage. An important assumption that the authors of this technique made is that the scene is bounded, which means that it is only an object with a pre-defined background color (which can be created using a segmentation algorithm). This bounded scene assumption allows for the authors, in a

preprocessing stage, to find a bounding box that bounds the object using the extrinsic camera parameters used as input for the technique. The authors also use some procedures during both training procedures, such as a per-voxel learning rate that aims to give a higher learning rate to voxels seen by more training images. We provide the pipeline of the original DirectVoxGO technique in Figure 14. For the pipeline, we follow the following steps:

1. In **(a)** we can see the basic volume rendering formulation and optimization procedure;
2. In **(b)** we can see the basic grid suggested for the fine-stage optimization, with the feature and density grids;
3. In **(c)** we can see the idea of the coarse training stage, where we aim to find the coarse RGB and density grids.
4. Finally, in **(d)** we have the feature grids, density and MLP optimized during the fine training.

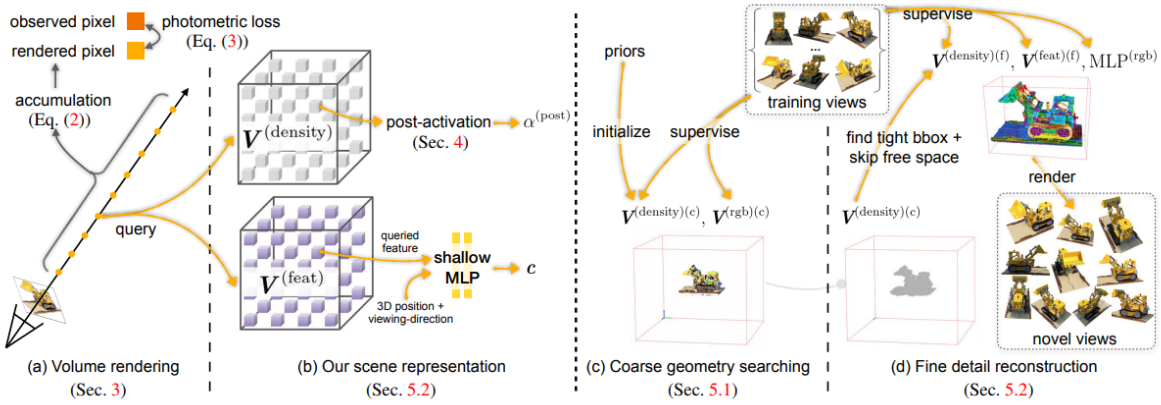

Figure 14: Pipeline of DirectVoxGO. Image from *Sun et al.* [52]

4.2.1 Coarse Training

This stage aims to find a coarse density grid prior. Given as input initial hyperparameter grid dimensions $(\mathbf{N_x^{(c)}}, \mathbf{N_y^{(c)}}, \mathbf{N_z^{(c)}})$, the authors can then use these grids to perform their optimization. For this, the authors optimize two grids, an RGB grid $\mathbf{V_{RGB}^{(c)}} \in \mathbf{V^{3 \times N_x^{(c)} \times N_y^{(c)} \times N_z^{(c)}}}$ and a density grid $\mathbf{V_{density}^{(c)}} \in \mathbf{V^{1 \times N_x^{(c)} \times N_y^{(c)} \times N_z^{(c)}}}$, where the aim is to find a prior $\mathbf{V_{density}^{(c)}}$ that represents the coarse geometry of the scene.

To use the grids, we will use a trilinear interpolation operation, which we will define as **interp**$(.,.)$, where, given as input a point $\mathbf{p_i} \in \mathbf{R}^3$ and a grid with an arbitrary number of channels C and dimensions (H,W,D), the **interp**$(.,.)$ operation has the following mapping:

$$\mathbf{interp} : \mathbf{R}^3 \times \mathbf{V^{C \times H \times W \times D}} \longrightarrow \mathbf{R}^C. \tag{4.8}$$

The voxel grids are initialized with zeros and optimized during training. Following a scheme similar to the original NeRF, with known camera parameters, we can compute for each pixel the ray $\mathbf{r}(.)$ that passes through it and, with this, we sample N points $\mathbf{p_i}$ in the ray and then perform volume rendering to find the color of the ray. To perform this, we need to find the RGB color and density $\mathbf{c_i} \in \mathbf{R}^3, \sigma_i \in \mathbf{R}$ for these N points in the ray.

In the coarse training stage, we can find the RGB color $\mathbf{c_i}$ for each point $\mathbf{p_i}$ and each grid $\mathbf{V}^{\mathbf{3}\times\mathbf{N}_\mathbf{x}^{(\mathbf{c})}\times\mathbf{N}_\mathbf{y}^{(\mathbf{c})}\times\mathbf{N}_\mathbf{z}^{(\mathbf{c})}}$ by computing

$$\mathbf{c_i} = \mathbf{interp}(\mathbf{p_i}, \mathbf{V}^{(\mathbf{c})}_{\mathbf{RGB}}). \tag{4.9}$$

For the density σ_i we use a similar procedure:

$$\sigma_i = \mathbf{softplus}(\mathbf{interp}(\mathbf{p_i}, \mathbf{V}^{(\mathbf{c})}_{\mathbf{density}}) + b), \tag{4.10}$$

where b is a hyperparameter bias term.

To find the probability of termination at point $\mathbf{p_i}$, we just compute $\alpha_i = \mathbf{alpha}(\sigma_i)$, where $\mathbf{alpha}(.)$ is the function defined in Equation 4.3. The stepsize δ_i, a hyperparameter, is ommited for brevity. Also, the $\mathbf{soflphlus}(.)$ function is an activation functon defined as: $\mathbf{soflphlus}(x) = \ln(1+\exp(x))$ [11].

To perform the rendering, the authors follow the same procedure as NeRF, first computing the opacity weights T_i for $i = 0, 1...N+1$ using Equation 4.4 then computing the color of the ray $\mathbf{r}(.)$ using Equation 4.5 to perform the volume rendering and, finally, composing with the background by using Equation 4.6. After this computation is finalized, we can perform the optimization.

For the optimization procedure, the authors compose a final loss $\mathscr{L}_{total}$ using three partial losses: a photometric loss $\mathscr{L}_{photo}$, a per-point regularization $\mathscr{L}_{point}$ and a background entropy regularization $\mathscr{L}_{bg}$.

The photometric loss $\mathscr{L}_{photo}$ is the one used by the original NeRF, as shown in Equation 4.7. The authors define the per-point regularization $\mathscr{L}_{point}$ as:

$$\mathscr{L}_{point} = \frac{\mathbf{1}}{|\mathscr{R}|} \sum_{\mathbf{r}\in\mathscr{R}} \sum_{\mathbf{i=1}}^{\mathbf{i=N}} \mathbf{T_i}\alpha_\mathbf{i}||\mathbf{c_i} - \mathbf{C}(\mathbf{r})||^\mathbf{2}, \tag{4.11}$$

and also a background entropy regularization $\mathscr{L}_{bg}$. that aims to create a balance between the background and the foreground defined as:

$$\mathscr{L}_{bg} = -T_{N+1}\log(T_{N+1}) - (1-T_{N+1})\log(1-T_{N+1}). \tag{4.12}$$

The final loss $\mathscr{L}_{total}$ is defined by using the hyperparameter weights w_{photo}, w_{point} and w_{bg} for each partial loss:

$$\mathscr{L}_{total} = w_{photo}\cdot\mathscr{L}_{photo} + w_{bg}\cdot\mathscr{L}_{bg} + w_{point}\cdot\mathscr{L}_{point}. \tag{4.13}$$

For the optimization, the authors used the Adam optimizer.

4.2.2 Fine Training

After obtaining a coarse density grid, $\mathbf{V}^{(\mathbf{c})}_{\mathbf{density}}$ will allow us to estimate a tighter bounding box and skip points in low-density areas. In this step, we will use a density grid with a much higher resolution $\mathbf{V}^{(\mathbf{f})}_{\mathbf{density}} \in \mathbf{V}^{\mathbf{1}\times\mathbf{N}^{(\mathbf{f})}_{\mathbf{x}}\times\mathbf{N}^{(\mathbf{f})}_{\mathbf{y}}\times\mathbf{N}^{(\mathbf{f})}_{\mathbf{z}}}$, a feature grid that will allow us to to extract local features with dimension $\mathbf{C}$ from the grid $\mathbf{V}^{(\mathbf{f})}_{\mathbf{features}} \in \mathbf{V}^{\mathbf{C}\times\mathbf{N}^{(\mathbf{f})}_{\mathbf{x}}\times\mathbf{N}^{(\mathbf{f})}_{\mathbf{y}}\times\mathbf{N}^{(\mathbf{f})}_{\mathbf{z}}}$ and a **MLP** to extract $\mathbf{c_i}$ and σ_i from the sampled points $\mathbf{p_i}$ in the rays $\mathbf{r}(.)$.

To do so, we use equations similar to Equations 4.10 and 4.9, and follow a rather similar pipeline. For a given sampled point $\mathbf{p_i}$, and its correspondent unitary view direction vector $\mathbf{d}$, we compute $\mathbf{c_i}$ by

$$\mathbf{c_i} = \mathbf{MLP}(\mathbf{interp}(\mathbf{p_i}, \mathbf{V}^{(\mathbf{f})}_{\mathbf{features}}), \mathbf{PE}(\mathbf{p_i}), \mathbf{PE}(\mathbf{d})). \tag{4.14}$$

It is important to note that this **MLP** receives more inputs than the original NeRF MLP. For the density σ_i, we use a similar procedure:

$$\sigma_i = \mathbf{softplus}(\mathbf{interp}(\mathbf{p_i}, \mathbf{V}^{(\mathbf{f})}_{\mathbf{density}}) + b), \tag{4.15}$$

where b is a bias hyperparameter. We follow the standard volume rendering pipeline from the previous sections and the same losses and optimization procedure to compute the ray's color. Also, we will scale the grid using trilinear interpolation so it can have a finer resolution during specific moments during the optimization procedure. In this manner, the grid is not static during training and will scale to its final resolution by the end of the process.

4.3 MODIFICATIONS INTRODUCED BY DIRECTVOXGO++

This section points to the modifications that our technique introduced in the DirectVoxGO pipeline. These can be listed as a separate preprocessing to account for the coloring of the background, using the ideas from NeRF++ [70], and the use of the neural hash encoding introduced by *Muller et al.* [35].

4.3.1 Preprocessing

Before we begin, we must detail our preprocessing steps before DirectVoxGO++ starts. Given a set of N_{images} RGB images I, we first pass this set to the COLMAP [47] camera calibration pipeline for obtaining the intrinsic matrix $\mathbf{K} \in \mathbf{R}^{3\times3}$ and the N_{images} extrinsic matrices. The extrinsic matrices encode the camera's position and orientation in 3D space. We can represent a extrinsic matrix $\mathbf{M} \in \mathbf{R}^{3\times4}$ as $\mathbf{M} = [\mathbf{R}|\mathbf{t}]$, where $\mathbf{R} \in \mathbf{R}^{3\times3}$ is a rotation matrix and $\mathbf{t} \in \mathbf{R}^{3\times1}$ is a translation vector.

Additionally, as in the original NeRF [33], we also extract bounds as to where the scene is localized. No other COLMAP results will be used in the remaining pipeline.

However, following the steps of NeRF++ [70], we need to ensure that the mean of the camera centers is at the origin of the camera coordinate system. Additionally, these camera centers are bounded by a unit sphere centered at the origin of the coordinate system. By this, given a set of all vectors which represent the camera centers $\mathbf{T}$, we compute the mean center using the $\mathbf{c_{mean}}$ function defined by:

$$\mathbf{c_{mean}} = \frac{1}{|\mathbf{T}|}\sum_{\mathbf{c}\in\mathbf{T}} \mathbf{c}. \tag{4.16}$$

With this, we can get a new set of camera centers by translating all points in $\mathbf{T}$ by this mean center, ensuring that the origin of the coordinate system $\mathbf{o}$ is the new mean of the centers and getting the new set $\mathbf{T}'$ by

$$\mathbf{T}' = \{\mathbf{c} - \mathbf{c_{mean}} \mid \mathbf{c} \in \mathbf{T}\}. \tag{4.17}$$

We can now discover the point distance d_{max} which is farthest from the origin by computing:

$$d_{max} = \max_{\mathbf{c}\in\mathbf{T}'} ||\mathbf{c} - \mathbf{o}||. \tag{4.18}$$

Now we scale these points by d_{max} to ensure that they are bounded by the unit sphere centered at the origin $\mathbf{o}$ and obtain the final set of translation vectors $\mathbf{T}''$ doing

$$\mathbf{T}'' = \{\frac{\mathbf{c}}{d_{max}} \mid \mathbf{c} \in \mathbf{T}'\}. \tag{4.19}$$

After performing this processing, we ensure that we can do the background sampling proposed in NeRF++.

4.3.2 Encoding

Inspired by *Mullter et al.* [35], we use a similar encoding technique as was proposed in their work. For the unitary direction vector $\mathbf{d}$, we use a spherical harmonics encoding $\mathbf{SE}(.)$ instead of the positional encoding $\mathbf{PE}(.)$ used by DirectVoxGO. The spherical harmonics encoding has a long history in the computer graphics field, and spherical harmonics coefficients are used to model lightning effects, for example [17]. Spherical harmonics can be seen as akin to Fourier transforms on spherical coordinates where, after we choose a degree, we compute the related coefficients as an encoding. In our case, we use a nested spherical harmonics setup with the degree at most 4, with the degree being a hyperparameter that can be tuned.

For the vectors that are not unitary, we use the neural hashing encoder $\mathbf{NHE}(.)$ proposed by *Muller et al.* [35], illustrated in Figure 15. Similarly to DirectVoxGO [52] and Plenoxels [66],

we define a grid of vertices with values that will be trilinearly interpolated. However, differently from these methods, we perform a L-levels multiresolution sampling in our encoding. In addition, we geometrically scale the resolution in which we will be sampling the grid. The result in each level is concatenated to form the final output feature. Also, instead of each vertex storing a value, we use a hashing approach. We transform the position of each vertex along each coordinate (x, y, z) into a different number along each dimension. We then use a spatial hashing function introduced by *Teschner et al.* [56]:

$$\textbf{hash}(x, y, z) = x \cdot \pi_1 \oplus y \cdot \pi_2 \oplus z \cdot \pi_3 \mod T, \tag{4.20}$$

where π_1, π_2, π_3 are large primes and T is the size of the hash table. The authors demonstrated the efficiency of their method, and due to this we replaced the traditional positional encoding $\textbf{PE}(.)$ by the neural hashing encoding $\textbf{NHE}(.)$.

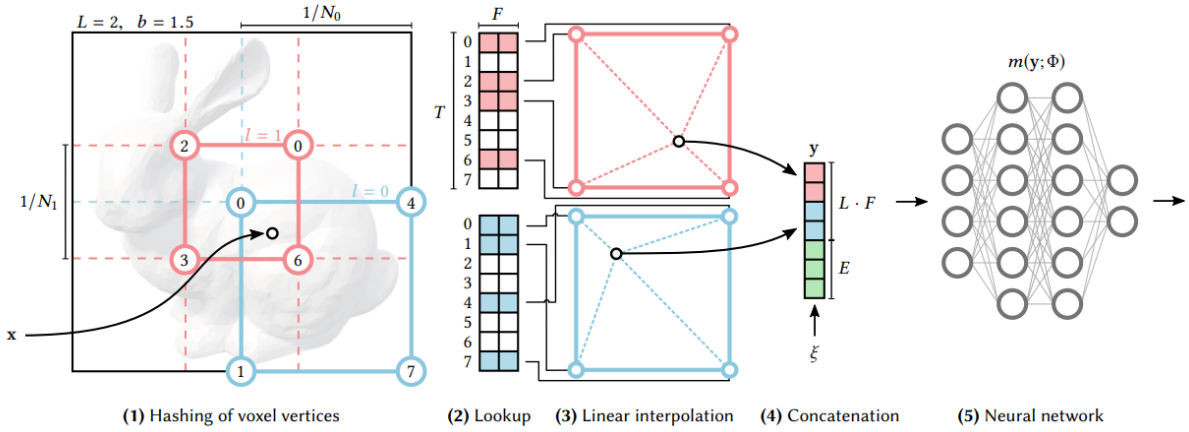

Figure 15: Pipeline for the neural hash encoding. In stage **(1)** we assign each position into an index. Then we use this index in a hash table to get the corresponding feature in step **(2)**. We then perform linear interpolation in step **(3)** and concatenate the resulting values in step **(4)**. Image from *Muller et al.* [35]

Using this neural hashing encoding, we modify Equation 4.14 and optimize the encoder **NHE**:

$$\mathbf{c_i} = \textbf{MLP}(\textbf{interp}(\mathbf{p_i}, \mathbf{V}^{(\mathbf{f})}_{\textbf{features}}), \textbf{NHE}(\mathbf{p_i}), \textbf{SE}(\mathbf{d})). \tag{4.21}$$

4.3.3 Background Colors

As mentioned before, both the original NeRF and DirectVoxGO assumed a pre-defined background color $\mathbf{c_{bg}}$. However, this severely limited the original technique and only allowed it to be used for bounded scenes. Due to this, we aim to extend this formulation to allow the DirectVoxGO technique to be used in unbounded scenes. With this aim, it would be ideal that

the value of $\mathbf{c_{bg}}$ is different for each ray $\mathbf{r}$ and dependent on the viewing parameters. That means that we are going to estimate $\mathbf{c_{bg}}(\mathbf{r})$ for each ray $\mathbf{r}(.)$. To achieve this, we use the approach of *multi-sphere images* (used in NeRF++ [70]).

We first consider a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o}$ is the origin of the ray and $\mathbf{d}$ is its direction. In this approach, we assume that a unit sphere centered at the origin bounds the camera centers and that the cameras are pointing towards the object of interest. First, we adopted a parameterization where, for each point (x,y,z), we have that if it is inside the unit sphere (e.g. $||(x,y,z)|| \leq 1$) then we will represent it in the usual coordinates. Else, if it is outside the sphere we will represent the point as $(x',y',z',\frac{1}{r})$, where r is the distance of the point to the origin and $||(x',y',z')|| = 1$. One form to see this parameterization is as if (x',y',z') gives us the unitary vector associated with the point. In contrast, $\frac{1}{r}$ provides us with the inverse of its distance to the center (or disparity). This representation will significantly aid us during sampling. An illustration is given in Figure 16.

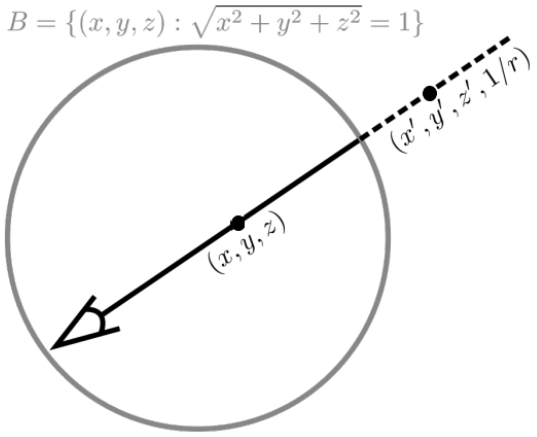

Figure 16: Image modified from NeRF++ [70], we can see that inside the sphere the coordinate system is unchanged while outside the sphere we normalize the coordinates (x,y,z) and add the fourth coordinate $\frac{1}{r}$ where r is the distance to the origin and B is the unit sphere

Now, to perform the sampling up until the intersection $\mathbf{a}$ of the ray and the sphere, we use the traditional sampling pipeline discussed in previous sections. We use this to compute the color of the foreground in Equation 4.5 just as usual. However, to sample the points $\mathbf{p_i^{bg}}$ in the background, we use the fact that since $r \in [1,\infty)$ then $\frac{1}{r} \in (0,1]$. This allows us to use $\frac{1}{r}$ as a parameter to sample background points based on its values, as used in NeRF++ [70].

To do so, we compute $\mathbf{a}$, the intersection of the ray $\mathbf{r}(.)$ and the unitary sphere, and $\mathbf{b}$, the midpoint of the chord aligning with the ray. Since both of these points are in the ray, then we have $\mathbf{a} = \mathbf{o} + t_a\mathbf{d}$ and $\mathbf{b} = \mathbf{o} + t_b\mathbf{d}$. We also define $\mathbf{c_{sph}}$ as the origin of the coordinate system and the center of the unit sphere.

However, we have that $||\mathbf{a} - \mathbf{c_{sph}}|| = ||\mathbf{o} + t_a\mathbf{d} - \mathbf{c_{sph}}|| = 1$ and $\mathbf{d}^T(\mathbf{b} - \mathbf{c_{sph}}) = \mathbf{d}^T(\mathbf{o} + t_b\mathbf{d} - \mathbf{c_{sph}}) = 0$. Next, to obtain a new point $\mathbf{p_i^{bg}}$ in the ray $\mathbf{r}(.)$ given $\frac{1}{r}$, we rotate $\mathbf{a}$ around the axis $(\mathbf{b} - \mathbf{c_{sph}}) \times \mathbf{d}$ by an angle $\omega = \arcsin||\mathbf{b} - \mathbf{c_{sph}}|| - \arcsin||\mathbf{b} - \mathbf{c_{sph}}|| \cdot \frac{1}{r}$. This parameterization

allows us to sample along with the background points.

Finally, as a final step in our parameterization, we convert it to a 3-coordinate system by the transformation $T(x,y,z,\frac{1}{r}) = (\frac{x}{r},\frac{y}{r},\frac{z}{r})$. We transform our originally unbounded set of points into a bounded sphere of unitary radius, which allows us to perform linear interpolation and use the previously discussed neural hashing encoding. We provide an illustration in Figure 17.

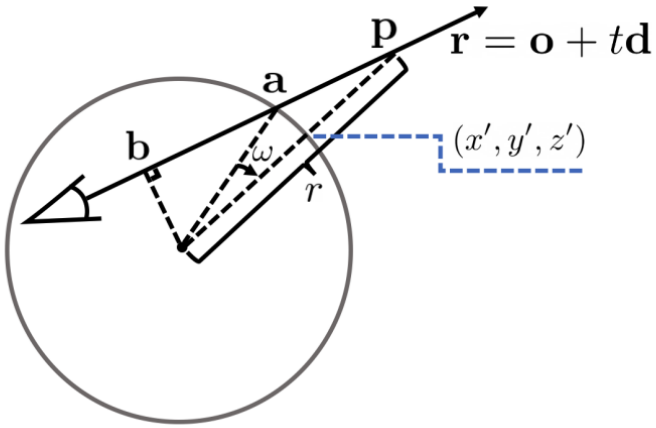

Figure 17: Given a ray defined by $\mathbf{r}(.)$, to compute a value of $\mathbf{p}$ corresponding to a background point given an arbitrary $\frac{1}{r}$ we first need to compute points $\mathbf{a}$ and $\mathbf{b}$, next we perform rotation around the axis $(\mathbf{b}-\mathbf{c_{sph}})\times\mathbf{d}$ of the angle $\omega = \arcsin||\mathbf{b}-\mathbf{c_{sph}}|| - \arcsin(||\mathbf{b}-\mathbf{c_{sph}}||\cdot\frac{1}{r})$. Image from NeRF++ [70]

Since we now have a manner to sample N^{bg} points $\mathbf{p_i^{bg}}$, we can now compute $\mathbf{c_{bg}}(\mathbf{r})$. We use a procedure similar to the main pipeline during the fine training stage for the foreground, where we will optimize a network $\mathbf{MLP^{bg}}$, a neural-hash encoder $\mathbf{NHE^{bg}}$ and a voxel density grid $\mathbf{V^{(bg)}_{density}} \in \mathbf{V^{1\times N_x^{(bg)}\times N_y^{(bg)}\times N_z^{(bg)}}}$, with the dimensions of the grid being hyperparameters. Unlike the grid used in the main pipeline, this grid is static, although we acknowledge this could be a direction for further research.

Similar to the foreground case, given a point $\mathbf{p_i^{bg}}$, we compute $\mathbf{c_i}$ by:

$$\mathbf{c_i} = \mathbf{MLP^{bg}}(\mathbf{NHE^{bg}}(\mathbf{p_i^{bg}}), \mathbf{SE}(\mathbf{d})). \tag{4.22}$$

For the density σ_i, we use a similar procedure:

$$\sigma_i = \mathbf{softplus}(\mathbf{interp}(\mathbf{p_i^{bg}}, G\cdot\mathbf{V^{(bg)}_{density}}) + b), \tag{4.23}$$

where b is a bias hyperparameter and G is a gain hyperparameter. To finally compute $\mathbf{c_{bg}}(\mathbf{r})$, we use the same traditional volume rendering pipeline we have been operating in the previous sections.

5

RESULTS

In this chapter, we show the results of this work. We compare our technique with the basis DirectVoxGO [52] and Plenoxels [66]. We chose to use the Plenoxels as a comparison because, aside from our techniques, it was the only NeRF-based technique that enabled the 3D reconstruction of the desired object in a reasonable timeframe (under the 9 hours required from the original NeRF) and managed to separate the foreground from the background. Before doing the evaluations, we will better explain the environment of our assessment.

To better evaluate the impact of each of our hypotheses individually **h1** (the effect of the background coloring) and **h2** (the effect of the neural hash encoder), we will perform an ablation study to evaluate our hypotheses.

5.1 EVALUATION SET-UP

To better evaluate the set-up of our evaluation, here we provide a more detailed explanation. We implemented our technique using PyTorch and with DirectVoxGO [52] as a starting point. We used the original code for DirectVoxGO and Plenoxels to make the evaluation fairer between the techniques. We tested the techniques in a Samsung Odyssey laptop with a CPU i7-7700HQ @ 2.80 GHz with 16 Gb RAM and a GPU NVIDIA GTX 1060 with 6 Gb. Due to the low memory of our set-up, we had to modify the original parameters of the Plenoxels and DirectVoxGO technique. Still, we tried to tune the parameters to evaluate the methods fairly.

We tested with four scenes from the LF Dataset [69], with a small set of the available images, using the data shared by the authors of NeRF++ [70]. The four scenes are all 360° rotations around a single small object. To extract the poses, we used the COLMAP technique. The same dataset was used to evaluate the three techniques.

As for the parameters, in our technique, we adopted a bias density term of $b = 0.163$. Regarding the grid size, we used 303000 voxels in the coarse stage and 125^3 voxels in the fine stage. For sampling, we used a step size of $\delta_i = 0.5$, and a number of samples $N = N_{bg} = 220$, both in the background and foreground in the fine stage.

As for Plenoxels, we used the default settings adopted in their paper for the Tanks and

Temples dataset, however we reduced the grid resolution parameters, reducing it to

$$\textbf{reso} = [[[128, 128, 128], [256, 256, 256], [256, 256, 256], [320, 320, 320]]]. \quad (5.1)$$

We also reduced the number of background layers to 64.

We used the following evaluation metrics:

- **PSNR**: Peak signal-to-noise ratio, a standard metric for signal processing and compression, which measures the amount of noise available in a signal compared with the original signal. We can use it for many types of signals. A higher PSNR, up to infinity, indicates a less noisy image.
- **SSIM**: Structural Similarity Index [59], which was developed for images and uses perceptual cues, trying to create a metric that attempts to measure the similarity of the structure of the picture. A higher SSIM indicates, up to one, a more similar image.
- **LPIPS**: Learned Perceptual Image Patch Similarity [71], which uses features extracted from deep neural networks to create its similarity metric. In our comparisons, we use a VGG backbone. A small LPIPS, with a minimum of zero, indicates a perceptually similar image in this metric.

We chose these metrics because of their use in similar works [70][66][52].

5.2 QUANTITATIVE COMPARISONS

Table 1 with the average value in the four LF Dataset scenes for each of the metrics shows that our method achieved better results than both Plenoxels and the standard DirectVoxGO. One may note that our increase is relatively modest compared with the Plenoxels technique. However, we must point out that, from our qualitative observations, we observed that the region inside the object of interest is of better quality than the region outside it. Our technique also managed to achieve better object segmentation than Plenoxels. In the next section on qualitative results, we will discuss this in more detail. We also put the values from NeRF++ extracted from its paper. We only display these values for comparisons as a gold standard, since NeRF++ has a long training time (around 9 hours per scene). Since the results are deterministic, we only needed to run our experiments once.

We used a segmentation mask created from the foreground reconstruction obtained by our technique to test this hypothesis. We used the images generated assuming a background entirely with black color. We used a small threshold of 0.09 to account for small density values still present in the image. Doing this allowed us to directly compare the techniques regarding the reconstruction of the object of interest, as shown in Table 2 with the average value in the four LF Dataset scenes for each of the metrics.

Models	Evaluation results in LF Dataset		
	PSNR(↑)	SSIM(↑)	LPIPS(↓)
DirectVoxGO	20.461	0.658	0.366
Plenoxels	21.912	0.746	0.292
DirectVoxGO++ (ours)	**22.436**	**0.804**	**0.266**
NeRF++	24.820	0.885	0.221

Table 1: Comparison with previous methods on LF Dataset, we highlight the best result in each metric. We put the original values of NeRF++ as a gold standard, but we do not make comparisons due to its high running time (9 hours) and memory requirements.

Models	Evaluation results in LF Dataset		
	PSNR(↑)	SSIM(↑)	LPIPS(↓)
DirectVoxGO	27.002	0.904	0.102
Plenoxels	28.919	0.932	0.074
DirectVoxGO++ (ours)	**33.953**	**0.980**	**0.018**

Table 2: Comparison with previous methods on LF Dataset with our masks applied to the objects, we highlight the best result in each metric.

These tests showed that our technique learned and modeled the object very well in low-memory settings compared with the state-of-the-art methods. We obtained better results and a much higher difference between our results and previous techniques.

As for memory usage and execution time for training on each scene, the original DirectVoxGO runs in around 15 minutes, while our technique and Plenoxels run in 25-28 minutes. The techniques tested used around 5 GB of GPU memory.

5.3 QUALITATIVE COMPARISON

In this section, we show a qualitative evaluation of our results. For example, in Figure 18, we can see the results we obtained for one test image from the Africa scene. As can be seen, DirectVoxGO did not manage to reconstruct the giraffe very well, with the resulting image presenting blurry features. Our technique was the one that best managed to model the giraffe, with special note to the checkered stamp on the table. We observe that the Plenoxels technique was better than our technique in the LPIPS metric, most probably due to the background.

(a) DirectVoxGO: (22.857, 0.768, 0.260)

(b) Plenoxels: (25.656, 0.855, 0.201)

(c) Our technique: (26.136, 0.867, 0.209)

(d) **Ground Truth**

Figure 18: Africa scene, with each result of the techniques along with the Ground-truth. They are with their respective metrics, where: (PSNR↑, SSIM↑, LPIPS↓). For comparisons, the gold standard, NeRF++, has the following values in this scene: (27.410, 0.923, 0.163). We used 56 images during training and 8 images during testing with a resolution of 320×180.

From the four scenes we analyzed, the worst result was obtained in the Torch scene, illustrated in Figure 19. This scene is especially challenging for the evaluated techniques due to moving persons in the middle of the photos. This factor is not accounted for in the original NeRF, albeit solved in other works [43]. The fact that it is our worst result may be due to the blurrier background. In comparison, the high-frequency details in our technique are more present than in the others.

(a) DirectVoxGO: (21.479, 0.692, 0.327)

(b) Plenoxels: (23.285, 0.802, 0.229)

(c) Our technique: (23.164, 0.794, 0.242)

(d) **Ground Truth**

Figure 19: Torch scene, with each result of the techniques along with the Ground-truth. They are with their respective metrics, where: (PSNR↑, SSIM↑, LPIPS↓). For comparisons, the gold standard, NeRF++, has the following values in this scene: (24.680, 0.867, 0.226). We used 53 images during training and 8 images during testing with a resolution of 320×180.

However, our best result, compared with the other techniques, is with the basket scene, illustrated in Figure 20. The other methods did not manage to capture the fine detail of the holes in the basket.

(a) DirectVoxGO: (21.479, 0.692, 0.327)

(b) Plenoxels: (20.540, 0.690, 0.354)

(c) Our technique: (21.708, 0.814, 0.290)

(d) **Ground Truth**

Figure 20: Basket scene, with each result of the techniques along with the Ground-truth. They are with their respective metrics, where: (PSNR↑, SSIM↑, LPIPS↓). For comparisons, the gold standard, NeRF++, has the following values in this scene: (21.840, 0.884, 0.254). We used 74 images during training and 11 images during testing with a resolution of 320×180.

From what we observed, one of the most challenging scene was the Ship one, as shown in Figure 21. We conjecture that this may be due to this scene's thin structures, such as the mast on the ship. As happened in the other scenes, our technique was the one that was more able to represent high-frequency details.

(a) DirectVoxGO: (23.345, 0.764, 0.266)

(b) Plenoxels: (25.990, 0.857, 0.206)

(c) Our technique: (25.576, 0.872, 0.200)

(d) **Ground Truth**

Figure 21: Ship scene, with each result of the techniques along with the Ground-truth. They are with their respective metrics, where: (PSNR↑, SSIM↑, LPIPS↓). For comparisons, the gold standard, NeRF++, has the following values in this scene: (25.350, 0.867, 0.241). We used 95 images during training and 14 images during testing with a resolution of 320×180

As shown in Figure 22, the Plenoxels technique did not manage to focus its segmentation on the object. Only a single grid and MLP do not have enough capacity to model the foreground and the background, causing the foreground areas to suffer compared with our work. A similar effect happened in the original DirectVoxGO, and an akin issue was shown and addressed in NeRF++ [70]. In our case, since we managed to segment the foreground from the background successfully, our foreground model can better model it with more high-frequency details.

Our technique	Plenoxels

(a) Masked Score: (36.643, 0.992, 0.006)

(b) Masked Score: (29.956, 0.951, 0.055)

(c) Masked Score: (33.468, 0.983, 0.016)

(d) Masked Score: (29.801, 0.957, 0.056)

(e) Masked Score: (31.449, 0.966, 0.033)

(f) Masked Score: (26.053, 0.884, 0.116)

(g) Masked Score: (34.253, 0.978, 0.018)

(h) Masked Score: (29.865, 0.938, 0.070)

Figure 22: The foreground captured by our DirectVoxGO++ technique compared with the foreground captured by Plenoxels. Bellow, we report the score obtained by applying the masks obtained by our technique, as reported in Table 2. We show the scores, where: (PSNR↑, SSIM↑, LPIPS↓).

5.4 ABLATION STUDY

This section evaluates the impact of both of our proposals individually. In Table 3 with the average value in the four LF Dataset scenes for each of the metrics, we can observe that the modifications proposed in this work, individually, managed to improve upon the original DirectVoxGO. However, we observed that we could significantly improve the results by combining both modifications, confirming the hypotheses **h1** and **h2** set out in our work.

For a qualitative evaluation, we report the results of one of the scenes in Figure 23. Specifically, we observed qualitatively that the background coloring managed to make the object stay sharp. Meanwhile, the neural hash encoder aided in letting our technique infer more detail from the model. Combining both techniques, we obtained the best results.

Models	Evaluation results in LF Dataset		
	PSNR(↑)	SSIM(↑)	LPIPS(↓)
DirectVoxGO	20.461	0.658	0.366
DirectVoxGO+(BG)	21.118	0.727	0.319
DirectVoxGO+(NHE)	21.015	0.722	0.312
DirectVoxGO++ (ours)	**22.436**	**0.804**	**0.266**
NeRF++	24.820	0.885	0.221

Table 3: Ablation study on LF Dataset. We test our DirectVoxGO++ technique, DirectVoxGO augmented with background colors (DirectVoxGO+BG) and a variant of DirectVoxGO with the neural hash encoder (DirectVoxGO+NHE) and the original DirectVoxGO. We provide the results in NeRF++ only as a gold standard since its running time is an order of magnitude greater than the other evaluated techniques.

(a) DirectVoxGO: (22.857, 0.768, 0.260)

(b) DVGO+(BG): (23.991, 0.795, 0,272)

(c) DVGO+(NHE): (23.499, 0.813, 0.210)

(d) **DirectVoxGO++**: (26.136, 0.867, 0.209)

Figure 23: Africa scene, with each result of the ablation study, where DVGO+(BG) corresponds to only adding the background coloring and DVGO+(NHE) corresponds to only adding the neural hash encoder. They are with their respective metrics, where (PSNR↑, SSIM↑, LPIPS↓). For comparisons, the gold standard, NeRF++, has the following values in this scene: (27.410, 0.923, 0.163). We used 56 images during training and 8 images during testing with a resolution of 320×180.

6

CONCLUSIONS

This chapter presents some final thoughts about our work, highlighting the main contributions and pointing out our possible future works.

6.1 CONTRIBUTIONS

We developed a literature review of NeRF techniques, especially those aiming to perform the reconstruction quickly. We also developed and explained a method for 3D reconstruction of an object of interest and background separation, DirectVoxGO++, that outperforms both DirectVoxGO and Plenoxels in the LF Dataset.

We have observed that combining the idea of splitting the rendering equation between the foreground and the background can not only improve the result for 360° scenes but also allows us to create a 3D model of the object of interest separate from the background. We also observed that we could integrate the neural hash encoding suggested by *Muller et al.* [35] into DirectVoxGO. Individually, each of these methods improved DirectVoxGO, with the Neural Hash Encoder enhancing the details of the image and the background colors allowing the foreground to appear more solid. However, combined, both techniques improved the results significantly, outperforming Plenoxels. The results we obtained confirm the hypotheses (**h1** and **h2**) that using the improvements devised in NeRF++ and the neural hash encoding improved the results significantly.

6.2 FUTURE WORKS

Although our technique achieved good results in the evaluated dataset, we also mapped some limitations that suggest some room for improvement. Although we do not take hours, such as the original NeRF, we still need to take time in the order of minutes. As *Muller et al.* [35] showed, using CUDA kernels and implementation in a lower-level language such as C++ as opposed to Python can significantly improve the performance time. Also, we still have much to improve in terms of quality to achieve photorealistic rendering. In future works, one exciting avenue would be to use the tensorial decomposition released in TensoRF [5] in our pipeline to

obtain better results. Although its authors did not account for unbounded scenes, we could do a similar extension as we did to DirectVoxGO for allowing it to deal with this domain. As we have shown with our problems with the Torch scene, a possible direction would be to integrate deformable NeRF [43] ideas into our pipeline and extend it to videos.

REFERENCES

[1] (2019). Visgraf recria múmia de 2 mil anos em realidade virtual. https://impa.br/en_US/noticias/visgraf-recria-mumia-de-2-mil-anos-em-realidade-virtual/. Accessed: 2022-03-05.

[2] Aharchi, M. & Ait Kbir, M. (2019). A review on 3d reconstruction techniques from 2d images. In *The Proceedings of the Third International Conference on Smart City Applications*, 510–522.

[3] Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., & Srinivasan, P. P. (2021a). Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *arXiv preprint arXiv:2103.13415*.

[4] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., & Hedman, P. (2021b). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *arXiv*.

[5] Chen, A., Xu, Z., Geiger, A., , Yu, J., & Su, H. (2022a). Tensorf: Tensorial radiance fields.

[6] Chen, A., Xu, Z., Geiger, A., Yu, J., & Su, H. (2022b). Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*.

[7] Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., & Su, H. (2021). Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. *arXiv preprint arXiv:2103.15595*.

[8] Chiang, P.-Z., Tsai, M.-S., Tseng, H.-Y., Lai, W.-S., & Chiu, W.-C. (2022). Stylizing 3d scene via implicit representation and hypernetwork. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 1475–1484.

[9] Dellaert, F. & Yen-Chen, L. (2020). Neural volume rendering: Nerf and beyond. *arXiv preprint arXiv:2101.05204*.

[10] Drebin, R. A., Carpenter, L., & Hanrahan, P. (1988). Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74.

[11] Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2000). Incorporating second-order functional knowledge for better option pricing. *Advances in neural information processing systems*, 13.

[12] Faraday, M. (1846). Liv. thoughts on ray-vibrations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 28(188):345–350.

[13] Fernando, R. *et al.* (2004). *GPU gems: programming techniques, tips, and tricks for real-time graphics*, volume 590. Addison-Wesley Reading.

[14] Fong, J., Wrenninge, M., Kulla, C., & Habel, R. (2017). Production volume rendering: Siggraph 2017 course. In *ACM SIGGRAPH 2017 Courses*, 1–79.

[15] Gershun, A. (1939). The light field. *Journal of Mathematics and Physics*, 18(1-4):51–151.

[16] Gortler, S. J., Grzeszczuk, R., Szeliski, R., & Cohen, M. F. (1996). The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 43–54.

[17] Green, R. (2003). Spherical harmonic lighting: The gritty details. In *Archives of the game developers conference*, 56:4.

[18] Han, X.-F., Laga, H., & Bennamoun, M. (2019). Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1578–1604.

[19] Hartley, R. & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[20] Jain, A., Mildenhall, B., Barron, J. T., Abbeel, P., & Poole, B. (2021). Zero-shot text-guided object generation with dream fields. *arXiv preprint arXiv:2112.01455*.

[21] Kalantari, N. K., Wang, T.-C., & Ramamoorthi, R. (2016). Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10.

[22] Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., & Gaidon, A. (2020). Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*.

[23] Kersten, T. & Lindstaedt, M. (2012). Potential of automatic 3d object reconstruction from multiple images for applications in architecture, cultural heritage and archaeology. *International Journal of Heritage in the Digital Era*, 1(3):399–420.

[24] Kopf, J., Matzen, K., Alsisan, S., Quigley, O., Ge, F., Chong, Y., Patterson, J., Frahm, J.-M., Wu, S., Yu, M., *et al.* (2020). One shot 3d photography. *ACM Transactions on Graphics (TOG)*, 39(4):76–1.

[25] Kulhánek, J., Derner, E., Sattler, T., & Babuška, R. (2022). Viewformer: Nerf-free neural rendering from few images using transformers. *arXiv preprint arXiv:2203.10157*.

[26] Levoy, M. & Hanrahan, P. (1996). Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 31–42.

[27] Levoy, M., Ng, R., Adams, A., Footer, M., & Horowitz, M. (2006). Light field microscopy. In *ACM SIGGRAPH 2006 Papers*, 924–934.

[28] Lindell, D. B., Martel, J. N., & Wetzstein, G. (2021). Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14556–14565.

[29] Ma, Z. & Liu, S. (2018). A review of 3d reconstruction techniques in civil engineering and their applications. *Advanced Engineering Informatics*, 37:163–174.

[30] Maher, M. M., Kalra, M. K., Sahani, D. V., Perumpillichira, J. J., Rizzo, S., Saini, S., & Mueller, P. R. (2004). Techniques, clinical applications and limitations of 3d reconstruction in ct of the abdomen. *Korean Journal of Radiology*, 5(1):55–67.

[31] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., & Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4460–4470.

[32] Mildenhall, B., Srinivasan, P. P., Ortiz-Cayon, R., Kalantari, N. K., Ramamoorthi, R., Ng, R., & Kar, A. (2019). Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*.

[33] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, 405–421.

[34] Moon, P. & Spencer, D. E. (1981). The photic field. *Cambridge*.

[35] Müller, T., Evans, A., Schied, C., & Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*.

[36] Musker, J. & Clements, R. (2016). Moana.

[37] Ney, D. R., Fishman, E. K., Magid, D., & Drebin, R. A. (1990). Volumetric rendering of computed tomography data: Principles and techniques. *IEEE Computer Graphics and Applications*, 10(2):24–32.

[38] Ng, R., Levoy, M., Brédif, M., Duval, G., Horowitz, M., & Hanrahan, P. (2005). *Light field photography with a hand-held plenoptic camera*. PhD thesis, Stanford University.

[39] Nichol, A., Achiam, J., & Schulman, J. (2018). On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

[40] Osher, S. & Fedkiw, R. (2003). Signed distance functions. In *Level set methods and dynamic implicit surfaces*, 17–22. Springer.

[41] Ost, J., Mannan, F., Thuerey, N., Knodt, J., & Heide, F. (2020). Neural scene graphs for dynamic scenes. *https://arxiv.org/abs/2011.10379*.

[42] Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 165–174.

[43] Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., & Martin-Brualla, R. (2020). Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*.

[44] Rao, C., Yu, H., Wan, H., Zhou, J., Zheng, Y., Ma, Y., Chen, A., Wu, M., Yuan, B., Zhou, P., *et al.* (2022). Icarus: A lightweight neural plenoptic rendering architecture. *arXiv preprint arXiv:2203.01414*.

[45] Reiser, C., Peng, S., Liao, Y., & Geiger, A. (2021). Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *arXiv preprint arXiv:2103.13744*.

[46] Schirmer, L., Schardong, G., da Silva, V., Lopes, H., Novello, T., Yukimura, D., Magalhaes, T., Paz, H., & Velho, L. (2021). Neural networks for implicit representations of 3d scenes. In *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 17–24.

[47] Schönberger, J. L. & Frahm, J.-M. (2016). Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[48] Shum, H.-Y., Chan, S.-C., & Kang, S. B. (2008). *Image-based rendering*. Springer Science & Business Media.

[49] Sitzmann, V., Martel, J. N., Bergman, A. W., Lindell, D. B., & Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*.

[50] Sitzmann, V., Rezchikov, S., Freeman, W. T., Tenenbaum, J. B., & Durand, F. (2021). Light field networks: Neural scene representations with single-evaluation rendering. In *Proc. NeurIPS*.

[51] Srinivasan, P. P., Tucker, R., Barron, J. T., Ramamoorthi, R., Ng, R., & Snavely, N. (2019). Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 175–184.

[52] Sun, C., Sun, M., & Chen, H.-T. (2022). Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction.

[53] Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

[54] Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P. P., Barron, J. T., & Kretzschmar, H. (2022). Block-nerf: Scalable large scene neural view synthesis. *arXiv preprint arXiv:2202.05263*.

[55] Tancik, M., Mildenhall, B., Wang, T., Schmidt, D., Srinivasan, P. P., Barron, J. T., & Ng, R. (2021). Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2846–2855.

[56] Teschner, M., Heidelberger, B., Müller, M., Pomerantes, D., & Gross, M. H. (2003). Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, 3:47–54.

[57] Vaish, V., Wilburn, B., Joshi, N., & Levoy, M. (2004). Using plane+ parallax for calibrating dense camera arrays. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 1:I–I.

[58] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., & Jiang, Y.-G. (2018). Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, 52–67.

[59] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

[60] Wang, Z., Wu, S., Xie, W., Chen, M., & Prisacariu, V. A. (2021). Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*.

[61] Williams, L. (1983). Pyramidal parametrics. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, 1–11.

[62] Xie, H., Yao, H., Sun, X., Zhou, S., & Zhang, S. (2019). Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2690–2698.

[63] Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G., & Cui, Z. (2021). Learning object-compositional neural radiance field for editable scene rendering. In *International Conference on Computer Vision (ICCV)*.

[64] Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Basri, R., & Lipman, Y. (2020). Multiview neural surface reconstruction by disentangling geometry and appearance. *arXiv preprint arXiv:2003.09852*.

[65] Yen-Chen, L., Florence, P., Barron, J. T., Rodriguez, A., Isola, P., & Lin, T.-Y. (2020). inerf: Inverting neural radiance fields for pose estimation. *arXiv preprint arXiv:2012.05877*.

[66] Yu, A., Fridovich-Keil, S., Tancik, M., Chen, Q., Recht, B., & Kanazawa, A. (2021a). Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*.

[67] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., & Kanazawa, A. (2021b). Plenoctrees for real-time rendering of neural radiance fields. *arXiv preprint arXiv:2103.14024*.

[68] Yu, A., Ye, V., Tancik, M., & Kanazawa, A. (2021c). pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4578–4587.

[69] Yücer, K., Sorkine-Hornung, A., Wang, O., & Sorkine-Hornung, O. (2016). Efficient 3d object segmentation from densely sampled light fields with applications to 3d reconstruction. *ACM Transactions on Graphics (TOG)*, 35(3):1–15.

[70] Zhang, K., Riegler, G., Snavely, N., & Koltun, V. (2020). Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*.

[71] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.

[72] Zhou, T., Tucker, R., Flynn, J., Fyffe, G., & Snavely, N. (2018). Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*.