Point Clouds and 3D Modeling
Project Report

Tanguy Magne
tanguy.magne@minesparis.psl.eu

Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction

---

Contents

1 Introduction and Problem Statement

Creating a novel view of a scene from different existing views is a difficult task. The idea is somehow to interpolate these already existing views of the scene. But obviously, a simple linear interpolation between two views of a scene does not create a realistic third viewpoint. It is due to occlusion that appears in real life. Indeed, when looking at a scene from a different point of view, different parts of this scene are occluded. This task of creating new viewpoints of a scene is referred to as view synthesis in the literature.

Because of occlusion, we need to have a sense of the geometry of the scene, i.e. the 3D of the scene, to perform view synthesis. To obtain this knowledge of the scene, we rely on the data that are available to us, namely some views of the scene, in the form of images or pictures. These views are however discrete and we only have a finite, and potentially small, number of them. We would like to be able to generate an infinite number of viewpoints of the scene.

This is a demanding task, and before the advent of deep neural networks and deep learnings methods, there were no simple methods that produced concluding results. Even with the development of these methods, good models and architectures suitable for this task have been discovered only recently. Indeed, until the publication of the NeRF paper in 2020 [14], no architectures had achieved impressive results. However, the proposed network has its flaws, the main one being the important time cost for both training and inference. Many papers came after it to try to address this problem. In this project, we are interested in one of them, namely "Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction" [22].

In this report, I will first make a literature review of the view synthesis task, starting with what has been done before the publication of the NeRF paper [14]. Then I will present the method proposed by the authors of NeRF and some evolution and improvement that have been published. The next step will be to explain the contribution of the paper presented in this project [22], but also its limitations. Finally, I will report some experiments I have conducted with the method proposed to show the benefits and the limits of what the authors of this paper did.

2 Related work

2.1 Before NeRF

Early attempts to solve the problem of novel view synthesis were not convincing. Some of them used simple or more complex interpolation of known views. Some other methods were based on the knowledge of the depth of each input image, thus allowing novel view synthesis with sparser input images but with lower quality and with the cost of having to estimate the depth in an image. Even some early attempts using convolutional neural networks weren't giving good results.

2.2 NeRF

The publication of the NeRF [14] article was a big step up for the view synthesis task. NeRF stands for Neural Radiance Field. In this section, I will explain how NeRF works.

The idea of NeRF is coming from a current trend in the deep learning community, one that we could call representative networks. In the classical deep learning framework, one has a model that he wants to train for a specific task on a training dataset. Then to evaluate the performances of the model, a test dataset is used. This dataset contains samples that are completely different

from the one of the training set. The goal is to encode a function that maps an input in a high dimensional space to an output that is a complex feature of the input.

Representative networks work differently. The goal of the network is to encode a single object in its weights. The network is itself a way of representing an object. Early attempts at doing this were trying to find a new representation for 3D shapes, that differs from meshes or point clouds. This is the case of DeepSDF [16] or Occupancy Networks [12]. Their goal was to learn implicit functions, with a level set corresponding to a 3D shape. Then using the marching cube algorithm [10], it allows to construct a mesh.

Input and Output of the model The NeRF article is based on the same idea, but instead of learning a simple 3D surface shape, it tries to learn a Radiance Field function threw a neural network. In the case of NeRF, one neural network represents one scene from which we want to synthesize new views. The input of the model is a 5-dimensional vector containing the position from where we want to look and the direction in which we want to look at. The spatial location is encoded by 3 parameters, (x, y, z), while the viewing direction can be encoded into two scalars (θ, ϕ), which corresponds to the angles defining the direction of viewing. Note that in practice this second part is also encoded in a 5-dimensional vector, that is unit norm vector pointing in the direction of viewing. Because of the constraint of the norm, this vector has two degrees of freedom. The output of the model is a 4 dimension vector containing the color $c = (r, g, b)$ and the volume density σ of the point (x, y, z) in the direction (θ, ϕ). This quantity σ is linked to the α coefficient of an RGBA color representation, and encodes the presence or absence of an object at a given position, as it can take values between 0 and 1.

Model and Training procedure The model is a simple multilayer perceptron (MLP) composed of 9 layers of 256 neurons each. The architectures is presented on the Figure 1. Actually, to predict σ only the position of the point is used, while the color c is also predicted using the viewing direction. The idea behind the construction of this architecture is that the color of an object can change depending on the direction we are viewing it, but its transparency doesn't. We will see later in this report that this assumption is not completely true and might lead to bad view synthesis.

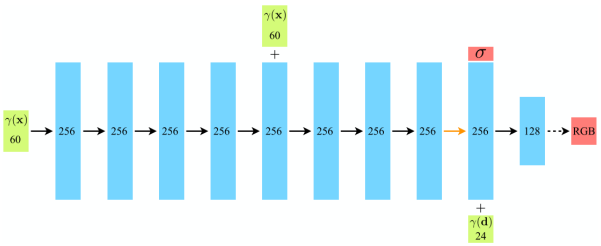

Figure 1: NeRF model architecture, green correspond to inputs, red to outputs. *Figure taken from [14]*

To train a model you need to have already existing views. You also need to know the extrinsic parameters of these viewpoints, that is rotation and translation from real-world coordinate to camera coordinate. Given this you can then use the Volume Rendering with Radiance Fields formula, which gives the expected colors $C(r)$ of a camera ray $r(t) = o + td$:

$$C(r) = \int_{t_n}^{t_f} \mathrm{T(t)}\ \sigma(r(t))\ \mathrm{c(r(t),\ d)}\ \mathrm{d}t \text{ where } T(t) = \exp(-\int_{t_n}^{t} \sigma(r(s))\,\mathrm{d}s) \tag{1}$$

This formula can be understand in the following way. The color observed at point o in direction d is the sum of the color observed at each points in the direction d $\mathrm{c(r(t),\ d)}$ pondered by the fact that the point $r(t)$ is actually visible $\sigma(r(t))$, and the fact that the ray has already seen a non transparent point when it reaches the point $r(t)$ $\mathrm{T(t)}$.

Using this formula, one can get the color estimated by the model at each pixel of the image corresponding to the views already at disposal. Then the loss is just the square norm between the true color and the color given by the model. Since equation (1) is differentiable, the loss thus obtained is also differentiable and it is possible to train the network.

Important tricks The authors presented two tricks that allow their model to work well. The first is linked to the small details of the scene. Indeed they explain that providing as input only the position and the viewing direction to the model gives results that are too smooth, and without fine details. To deal with this issue, they introduce what they called positional encoding, also called Fourier features in [24]. The idea behind this is to create multiple features out of each input of the model. To do so, they simply transform each of the inputs by passing them through sinusoidal function with different frequencies. In this way, close input will have close features for low frequencies, but different features for high frequencies allowing the model to distinguish them. This is presented in Figure 2. More technical and mathematical explanations of why this tricks works are developed in the following article [24].

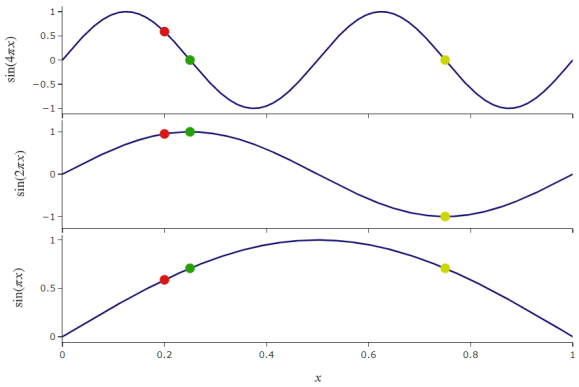

Figure 2: Illustration of Fourier Fourier Features transformations of input. Red and Green points are close, but they have different high-frequency features. On the opposite, yellow and red dots are different, they have different low-frequency features.

The second trick is used to improve inference time. Indeed, as explained above, the creation of a new view requires computing an integral over a ray direction for each pixel. This can be costly. However, we know that in a scene, most of the space is air so it is transparent, and we don't need to evaluate the function at these points. The idea is thus to have 2 networks, a coarse one and a fine one. The coarse one is used to find the relevant part to sample the point for evaluating the integral, and the fine one is used to sample densely in these interesting areas.

2.3 Improving NeRF

There have been many follow-ups to the NeRF paper, in very different directions. Indeed, even if NeRF is impressive, it also has strong limitations.

The first obvious limitation of the NeRF approach is that it requires the position of the camera to be known. In most real situations, these parameters are unknown, and therefore, they must be estimated using software such as COLMAP [18, 19]. Recent work tries to do without this information. This is the case of [8, 25], where the estimation is done during the training of the network.

Another limitation of NeRF is in the task it can accomplish. It can only be used for static tasks, where all input views are the same, with the same lighting conditions, and for relatively small scenes. Some articles have proposed methods to tackle all these problems. HeadNeRF [6] allows dealing with different facial expressions for novel view synthesis of faces. NeRF in the wild [11] allows to deal with noisy viewpoints, with changing occlusion, and so to deal with more general and real data. Block-NeRF [23] recently came out and allows to create novel views of entire neighborhoods.

Finally, a practical limitation of NeRF is the training and inference time of this method. This is the problem the paper studied in this project is trying to solve.

3 Analysis of the proposed method

The reason I went into the details of the NeRF [14] article is because the main paper of interest in this report [22] is using the same ideas. Indeed, the authors of this article also try to address the task of novel view synthesis by using radiance fields. Therefore, they also use the formulas presented before (1), and even the same tricks used by NeRF, namely the two-stage (coarse and fine) learning of the radiance fields, and the positional encoding.

3.1 General Idea

The method proposed still differs from the original one in many ways. The first one is linked to the way the authors represent the radiance field. Indeed, here the radiance field is not learned completely as an implicit function (the MLP) as done in NeRF, but, at least partially, explicitly on a grid of voxels.

In the coarse stage, there is no neural network. The space occupied by the scene is subdivided into voxels (generally 100 voxels in each direction). Then the goal is just to optimize the value inside each voxel, which contains a color c and a density σ, to be able to reconstruct the images of the dataset. To get the color of a point in a viewing direction, the method is simple and is the same as the NeRF one. The only difference is that to sample the values along the ray, it is not possible to use a neural network. However because we have the values on the grid, we can just interpolate this grid (with a trilinear interpolation) to get the values everywhere in the space.

The fine stage is a bit more sophisticated. For the density σ, the process is the same, but a finer grid (160 voxels in each direction) is used, and we sample points only if the density of the coarse grid at that point is higher than a certain threshold. This is done to sample only relevant regions, as in NeRF. However, for the colors, the process is different. On this finer grid, features of size $D = 12$ (D can be set to different values) are learned. These features are then fed to a shallow MLP (two hidden layers of 128 channels), alongside the Fourier features [24] of the position and the viewing direction. The output of the MLP is the color at this position in the viewing direction considered.

The idea of using both MLP and voxel grid to represent the radiance field is not new. It has already been proposed by [9] and [5]. However these methods still required a long training due to the presence of a bigger MLP in their architecture, or because they needed a conversion step from a trained implicit model to their final voxel grid representation.

3.2 Contributions

To overcome this problem, and be able to have fast training by directly optimizing on the voxel grid, the authors of the paper used 2 tricks.

Post-activated density voxel grid In general, the problem of having a voxel grid to represent a discrete version of a function is that a trade-off needs to be made. To have a good resolution, one needs to have lots of voxels. But because this is a 3-dimensional grid, this becomes space and time-consuming. Reducing the number of voxels will reduce the resolution and thus the memory footprint of the grid, but it will cause a smoothing effect, because of the trilinear interpolation. Therefore, we have to choose between efficiency and quality.

To overcome this problem, the authors of [22] introduced a post-activated density voxel grid. To understand it, we need to have a look at the discrete version of the equation (1) :

$$\hat{C}(r) = \sum_{i=1}^{K} T_i \alpha_i c_i(d) \text{ with } \alpha_i = \alpha(\sigma_i, \delta_i) = 1 - \exp(\sigma_i \delta_i) \text{ and } T_i = \prod_{j=1}^{i-1}(1 - \alpha_i) \tag{2}$$

and where we called $\sigma_i = \sigma(r(i))$ and $c_i(d) = c(r(i), d)$. We also have to remember that, on the grid, the values of the density are not necessarily positive. Therefore we have to apply an activation function, which is, in this case, the softplus function. To get the transparency α of a point on the grid, we need to apply the softplus and then the α function to the density.

For points that are on the grid, there is no question, but for points that are not on the grid, we can ask at what step should we perform the interpolation. The authors of this article showed mathematically that performing first the interpolation on the density voxel and then applying the activations function allows getting sharper edges inside voxels than first applying the activation function and then interpolating. This is what is called the post-activation, contrary to the pre-activation of the voxel grid.

This post-activation allows producing sharp edges even inside a voxel. Therefore it allows reducing quite significantly the resolution of the grid while keeping good and sharp results. This made the method more efficient in terms of memory footprint and less time-consuming in the training and inference phases.

Priors on the density voxel grid To further improve the performance of their method, the authors introduce two priors on the density voxel grid.

Low density initialization The first idea is pretty simple. It is based on the fact that most of the space is empty. Therefore, at the beginning of the optimization, all values of the density voxel grid should be set to something close to 0. Only a few voxels with non-zeros density will have to be changed. In addition, this prerequisite prevents voxels far from the camera from becoming less important at the beginning of the optimization. Indeed, as shown by the equation 2, if we set the initial values to something not close to zero, the transmittance T_i will drop quite significantly for points far from the camera, and thus they will have less importance. This can lead to cloudy results, where all densities are concentrated to voxels that are near the camera. In the article, the authors gave a formula that allows choosing the parameter of the softplus function so that the transmittance of the last voxel on the ray is whatever $1 - \alpha^{(init)}$ value required. In practice, the author proposed to tune this parameter so that the transmittance is $1 - \alpha^{(init)(c)} = 1 - 10^{-6}$ in the coarse stage and $1 - \alpha^{(init)(f)} = 1 - 10^{-2}$ in the fine stage.

View-count-based learning rate The second prior comes from the idea that voxels that are visible from a large set of training views will be more precise, and have more importance than the less visible ones. Therefore, the authors chose to make the learning rate different on each voxel. For voxels that are visible from more viewpoints, the learning rate is higher. The learning rate of each voxel is scaled with the factor $\frac{n}{n_{max}}$ where n is the number of training views to which the voxel is visible, and n_{max} is the maximum number of views to which any voxel is visible.

3.3 Limits

The contributions of this article allow to drastically reduce the training and inference time of NeRF while keeping similar performances. This is a really good point. However, it still has some limits.

The main issue is, to me, the memory cost of the model after the training. Indeed, with NeRF [14] the training time was quite long, but in the end, the representation of the scene is sparse and compact and weighs around 5 Mb. Here, the computational cost of training is much less important, but on the contrary, the memory footprint of the scene representation is much higher, around 0.7 Gb per scene. This can be a huge limitation for real-world applications.

Another limitation is the fact that the paper only achieves results that are as good as NeRF. Indeed, since the publication of the NeRF paper, new state-of-the-art methods have been published that have achieved much better results than NeRF. This is the case for instance of Mip-NeRF [1], which allows reducing the aliasing and blurred effect that can appear with the NeRF reconstructed views. This is also the case of the Jax implementation of NeRF [3].

Finally, I want to criticize an assumption made by both articles. Indeed both papers assumed that the density of a point doesn't depend on the viewing direction, as mentioned when presenting NeRF in section 2.2. However, if you look at tinted windows, on one side they are almost transparent, and therefore have a low density, while on the other side they are almost black, and have a high density. This is not possible with the proposed architectures, and therefore they do not model the world correctly.

4 Experimental results

After having seen the theoretical aspects of the article and having analyzed it in detail, I will now present some experiments I conducted to understand deeper the proposed method. For that, I used the code that was available on the `GitHub` repository of the project. It worked almost from scratch but I still had to modify a few things. In all this report, if not explicitly mentioned, the parameters used are the default ones, presented in the original article.

Dataset used Many datasets are available for this task. I will use many of them in my experiment. I'll explain each time which dataset I choose and why.

4.1 Metrics used

Before looking at the results, we need to have metrics to assess the performance of the method and the quality of the results. We thus want to have a way to compare two images and assess if they are the same. In this report and as it is done in most articles, I will use 3 of them.

PSNR The first one is the peak signal-to-noise ratio (PSNR). It is expressed in a logarithmic scale (dB), and it compares the mean square error between the predicted image and the ground truth image to the maximum error possible on an image. Mathematically, it can be written in the following way: $PSNR = 10 \log_{10}(MAX_I^2/MSE)$ where $MSE = mean((GT - Pred)^2)$ and MAX_I is the maximum error possible so it corresponds to the MSE between a black and a white image. Therefore, the greater the PSNR is, the better the result is because it measures in a way the distance to the maximum error, and we want this distance to be important. For this metric, higher is better.

SSIM Then there is the structural similarity index measure (SSIM). This metric is based on the statistics of the images (mean and variance) and corresponds to a combination of comparisons of the luminance, contrast, and structure of the two images. Even if there is a debate whether this indicator is independent and better than the PSNR [4], I will look at it as it is done in most papers. For this metric, higher is better.

LPIPS Learned Perceptual Image Patch Similarity (LPIPS) is a similarity measure based on features extracted by deep neural networks. It was proposed in [26] where it was presented as a better metric compared to all previous ones. The most commonly use is based on AlexNet [7] or VGG [20] backbone. For this metric, lower is better.

4.2 Reproducing Results

The first experimental test I wanted to run was to reproduce the results presented in the article. Indeed, a paper with results that are not reproducible is not a good paper. For this experiment, I choose to reproduce the results obtained on the Synthetic-NeRF and the Synthetic-NSVF dataset, as they are probably the two most commonly used datasets for the task of novel view synthesis.

	PSNR↑	SSIM↑	LPIPS^{VGG} ↓	LPIPS^{AlexNet} ↓
Results from paper [22, Table 1]	31.95	0.957	0.053	0.035
Personal results	31.90	0.956	0.054	0.035

Table 1: Table resuming results obtained on the Synthetic-NeRF dataset

	PSNR↑	SSIM↑	LPIPS^{VGG} ↓	LPIPS^{AlexNet} ↓
Results from paper [22, Table 1]	35.08	0.975	0.033	0.019
Personal results	34.93	0.975	0.033	0.019

Table 2: Table resuming results obtained on the Synthetic-NSVF dataset

The results I obtained and the one presented in the paper are reported in Tables 1 and 2 for the Synthetic-NeRF and the Synthetic-NSVF datasets respectively. We observe that on both datasets the results presented by the paper and the one I obtained are similar. This is a good sign, it shows that the paper didn't "cheat". The small difference observed can be imputed to the randomness present in the process. Details of the results for each scene of the datasets are presented in appendix A.1.

4.3 Ablation study

Now that I have checked that the code is working well, I would like to check the influence of the different contributions made by the authors. In the papers, they did a quite extensive ablation study, with a lot of details in the supplementary materials. These experiments show the benefits of the proposition made by the authors. However, these ablations are evaluated using only the PSNR metric. Moreover, the DeepVoxels [21] dataset is not used. Finally, some parameters are not tested. I will try to obtain these missing results in the following section.

Evaluation of other metric I first ran the ablation study on the scene *Materials* from the NeRF-Synthetic dataset, as it is one of the most used datasets for the novel view synthesis task. The goal here was to compare the results of the ablation study on the different metrics. The results obtained are reported in Table A3. I also reported in this table the results presented in the complementary materials of the article.

We can notice with these results that whatever the metric we choose, the best performances are achieved in the same settings. This supports the idea that reporting only the PSNR, as it is done in some tables of the article, is sufficient. I also run this experiment on another scene (*Mic*) of the NeRF-Synthetic dataset. Result are reported in appendix A.2.1. Overall, they reach the same conclusion, although there is a bit more uncertainty since several parameters give the same best scores for some metrics. Also, my results don't correspond exactly to the ones of the article, but the difference is really small, and therefore, these results allow us to confirm the results from the ablation study made in the paper.

	$\alpha^{(init)(c)}$				No Pervoxel LR	Pre Activation
	1.10^{-3}	1.10^{-4}	1.10^{-5}	1.10^{-6}		
↑PSNR	28.03	29.54	**29.59**	29.58	29.54	29.22
↑PSNR [22, Table J1]	28.96	29.35	**29.60**	29.57	29.57	29.24
↑SSIM	0.931	0.950	**0.951**	0.950	0.950	0.946
↓LPIPS^{VGG}	0.091	0.059	**0.058**	0.059	0.059	0.066
↓LPIPS^{AlexNet}	0.053	0.027	**0.026**	0.027	0.027	0.035

Table 3: Ablation study of the scene *Materials* from the NeRF-Synthetic dataset, with all metrics presented. Number in bold highlights the highest values of the corresponding metric.

Influence of $\alpha^{(init)(f)}$ In the article, the authors made an ablation study to evaluate the influence of the post-activation in the voxel grid, but also of the priors imposed to the representation, which are low-density initialization and view-count-based learning rate. However the model is composed of 2 stages, a fine and a coarse one, and they tested the influence only on the coarse stage. In this part, I focused on making an ablation study to look at the influence of these parameters on the fine stage of the model. I did this again on the NeRF-Synthetic dataset. The default parameters that the authors gave are $\alpha^{(init)(f)} = 0.01$ and $pervoxel\_lr = False$. I tried to change both of these parameters. The average results on the dataset are presented in Table 4, while detailed results on each of the scenes are presented in Appendix A.2.3.

	$\alpha^{(init)(f)}$					Pervoxel LR
	1.10^{-1}	1.10^{-2}	1.10^{-3}	1.10^{-4}	1.10^{-5}	*fine stage*
↑PSNR	31.77	**31.89**	31.82	31.76	10.84	**31.90**
↑SSIM	0.955	**0.956**	0.955	0.954	0.792	**0.956**
↓LPIPS^{VGG}	0.054	**0.053**	0.054	0.055	0.248	**0.053**
↓LPIPS^{AlexNet}	**0.035**	**0.035**	0.036	0.036	0.373	**0.035**

Table 4: Influence of the parameters on the fine part of the model. Ablation study made on NeRF-Synthetic dataset. Number in bold highlights the highest values of the corresponding metric.

We can notice two things on this table. Firstly the choice of $\alpha^{(init)(f)} = 0.01$ made in the paper is indeed a good choice, as it is the value of this parameter that gives the best results. However, choosing not to use different learning rates for each voxel based on the number of views from which it is seen can be discussed. Indeed, using per voxel learning rate seems to slightly increase the PSNR, and for the other metrics, we don't have enough precision to determine which setting is the best. We would need to perform further ablation study on other datasets to check that this behavior is consistent, but I found it weird not to use it as it was proposed in the paper. I believe that they don't use it to improve training time, as it doesn't seem to decrease much the performances.

4.4 Results on Forward Facing scene

Everything we have discussed so far has been applied to the case of synthesizing new views around an object. Another important case of application in the field of novel view synthesis is the case of a forward-facing scene. In this case, we don't want to reconstruct a point of view around a scene, but we are looking at a scene from the front and we want to change our viewing

directions slightly. This task was introduced by the paper Local Light Field Fusion [13] that presented the first method to solve it and introduced the LLFF dataset now widely used.

This task requires a slightly different method than the one used for the classical settings. The original papers [22] didn't present any method to tackle this, but the authors introduced a method on the `GitHub` repository recently. It is slightly different compared to the original methods, as it uses the NeRF normalized device coordinate (NDC) warping to preserve parallel lines on an image, but also multiplane images, adapt the loss, and skip the coarse stage.

The results of the method are presented on the `GitHub` repository of the project but no ablation study to assess the effective advantages of the contribution of the method are reported. I conducted one myself on this dataset. The average result are reported in the Table 5, and the per-scene results are presented in appendix A.3.

	$\alpha^{(init)(f)}$						Pre Activation
	1.10^{-1}	1.10^{-2}	1.10^{-3}	1.10^{-4}	1.10^{-5}	1.10^{-6}	$\alpha^{(init)(f)} = 1.10^{-2}$
↑PSNR	26.59	26.59	26.56	26.60	26.58	26.54	26.33
↑SSIM	0.839	0.839	0.839	0.839	0.839	0.839	0.831
↓LPIPS^{VGG}	0.227	0.227	0.227	0.226	0.227	0.227	0.240
↓LPIPS^{AlexNet}	0.143	0.143	0.142	0.143	0.143	0.142	0.160

Table 5: Ablation study on the LLFF dataset, composed of forward facing scene

We notice that the value of $\alpha^{(init)(f)}$ does not influence at all the results, all metrics are close. However, using post-activation indeed improves the performance quite significantly. The results are slightly better than the one presented on the `GitHub` page of the project (average PSNR of 25.71) and are comparable to the one obtained by NeRF, which is a good result.

4.5 Results on custom scene

A final experiment I wanted to run was to apply the method on custom and personal data to see if it gives good results when the data are not the standard ones. It is difficult to do it in the first case, on a single object, because you need to take pictures all-around an object with a white background. However, it is much more feasible in the case of the forward-facing scene.

I took photos of two scenes, one of a teddy bear and one of a plush panda holding chocolate. The results are presented in the Table 6, and visual results are presented in Appendix A.4.

	PSNR↑	SSIM↑	LPIPS^{VGG} ↓	LPIPS^{AlexNet} ↓
TeddyBear	31.25 (31.01)	0.875 (0.870)	0.298 (0.305)	0.192 (0.202)
Panda	24.61 (23.97)	0.834 (0.818)	0.446 (0.470)	0.374 (0.395)

Table 6: Table resuming results obtain on the custom scene (Number in gray correponds to results using pre-activation instead of post-activation)

The results are better when using post-activation even on this custom data. However, we can note that the results on the Panda scene are a bit blurry. This is probably because, on this scene, I took a pretty wide range of images, with viewing directions that differ much more than in the case of the LLFF dataset. Also the camera parameters estimation may not be perfect. The results on the TeddyBear scene look pretty good.

5 Possible improvements

With the theoretical analysis and the practical experiments conducted, we have seen that the method presented can achieve really good results with a lower computational cost than other previously proposed procedures. However, it also has its limitations as seen in sections 3.3 and 4.5. To overcome these problems, we can think of several methods. I will expose some here.

To deal with the problem of the memory cost of the method, we can think of more advanced data structures such as octree or kdtree, or even sparse voxel grid. This can be used to reduce the number of voxels in areas with low density, or in areas where the density evolves linearly.

To improve the performance of this method, and be able to get results that are as good as the one of Mip-NeRF [1], an idea would be to use higher-order interpolation on the voxel grid. For instance, using splines of order 3 or 5 could lead to improvements in the performance of the model.

To overcome the problem of tinted windows exposed above, one solution would be to consider the viewing direction from the beginning. This means taking it into account to predict both the color and the density. In the case of NeRF, it can be done easily. We only need to feed the MLP directly with both viewing direction and position. In the case of this article, it is a bit complicated, as it would require to have a 5-dimensional grid, the 5 dimensions corresponding to the position, and the viewing direction. But this would make the training and the memory footprint probably too high.

Finally, to be able to solve the problem of estimating the camera parameters, we could think of implementing a previously proposed method that estimates the camera pose alongside the optimization of the radiance field.

6 Conclusion

As we have seen through this report, the method proposed by the article brings significant improvements to the novel view synthesis task. It allows for fast training of the model and fast inference too, without decreasing the performance of the original NeRF article. The three main contributions of the paper indeed have an impact on the performance, as shown in the experimental part of this report. Moreover, it is easily applicable to new data. I would say that the main drawback is here the fact that it requires to have cameras parameters. These parameters can be computed using software such as COLMAP, but this takes a lot of time and thus reduce the benefits of the method.

Finally, like many tasks approached using deep learning methods, novel view synthesis is quite trending currently. New methods appear almost every week, and the state-of-the-art methods often don't remain state of the art for a long time. This is the case for the proposed method. Indeed it was published in late 2021, and was probably the fastest method for novel view synthesis at that time, alongside the Plenoxels methods [17], that achieved similar results in terms of training time, but slightly better performance, using similar methods but using sparse voxel grid.

However, as things are rapidly evolving in this field, at least two papers I am aware of have been published since then, and are capable of better performance and/or shorter training time. [15] has similar performance to MipNeRF [1], with a reduced training time of only 5 minutes, and uses hash encoding. TensoRF [2] is capable of results that are again as good as MipNeRF, with a memory footprint comparable to NeRF, and a training time comparable to the presented paper.

References

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

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, , Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields, 2022.

[3] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020.

[4] Richard Dosselmann and Xue Dong Yang. A comprehensive assessment of the structural similarity index. *Signal, Image and Video Processing*, 5(1):81–91, 2011.

[5] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021.

[6] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. 2021.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[8] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[9] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[10] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[11] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

[12] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[13] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[14] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[15] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv:2201.05989*, January 2022.

[16] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

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

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

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

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[21] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

[22] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *arXiv preprint arXiv:2111.11215*, 2021.

[23] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. *arXiv preprint arXiv:2202.05263*, 2022.

[24] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[25] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[26] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

A Appendix - Experimental Results

A.1 Reproducing Results

Scene		PSNR↑	SSIM↑	LPIPS^{VGG} ↓	LPIPS^{AlexNet} ↓
Chair	Results from paper [22, Table 1]	34.09	0.977	0.027	0.016
	Personal results	34.11	0.976	0.027	0.016
Drums	Results from paper [22, Table 1]	25.44	0.930	0.077	0.061
	Personal results	25.40	0.929	0.080	0.062
Ficus	Results from paper [22, Table 1]	32.78	0.978	0.024	0.015
	Personal results	32.65	0.978	0.025	0.015
Hotdog	Results from paper [22, Table 1]	36.74	0.980	0.034	0.017
	Personal results	36.68	0.980	0.034	0.017
Lego	Results from paper [22, Table 1]	34.64	0.976	0.028	0.014
	Personal results	34.60	0.976	0.027	0.013
Materials	Results from paper [22, Table 1]	29.57	0.951	0.058	0.026
	Personal results	29.52	0.950	0.059	0.027
Mic	Results from paper [22, Table 1]	33.20	0.983	0.017	0.014
	Personal results	33.20	0.983	0.018	0.014
Ship	Results from paper [22, Table 1]	29.13	0.879	0.161	0.118
	Personal results	29.04	0.877	0.161	0.117

Table A1: Table resuming per-scene results obtained on the Synthetic-NeRF dataset

Scene		PSNR↑	SSIM↑	LPIPS^{VGG} ↓	LPIPS^{AlexNet} ↓
Wineholder	Results from paper [22, Table 1]	30.26	0.949	0.055	0.038
	Personal results	30.17	0.948	0.057	0.039
Steamtrain	Results from paper [22, Table 1]	36.56	0.989	0.019	0.010
	Personal results	36.42	0.988	0.018	0.009
Toad	Results from paper [22, Table 1]	33.10	0.966	0.047	0.030
	Personal results	32.98	0.965	0.046	0.030
Robot	Results from paper [22, Table 1]	36.36	0.992	0.013	0.005
	Personal results	36.27	0.992	0.013	0.005
Bike	Results from paper [22, Table 1]	38.33	0.991	0.011	0.004
	Personal results	38.06	0.991	0.012	0.004
Palace	Results from paper [22, Table 1]	34.49	0.962	0.043	0.027
	Personal results	34.28	0.961	0.044	0.028
Spaceship	Results from paper [22, Table 1]	37.71	0.988	0.019	0.009
	Personal results	37.53	0.987	0.020	0.010
Lifestyle	Results from paper [22, Table 1]	33.79	0.965	0.054	0.027
	Personal results	33.69	0.965	0.054	0.027

Table A2: Table resuming per-scene results obtained on the Synthetic-NSVF dataset

A.2 Ablation study

A.2.1 Evaluation of other metrics

	$\alpha^{(init)(c)}$				No Pervoxel LR	Pre Activation
	1.10^{-3}	1.10^{-4}	1.10^{-5}	1.10^{-6}		
↑PSNR	30.68	33.08	**33.15**	33.11	33.12	31.73
↑PSNR [22, Table J1]	32.97	**33.23**	33.22	33.20	33.22	32.43
↑SSIM	0.965	0.982	**0.983**	**0.983**	0.982	0.977
↓LPIPS^{VGG}	0.047	**0.018**	**0.018**	**0.018**	**0.018**	0.027
↓LPIPS^{AlexNet}	0.046	**0.014**	**0.014**	**0.014**	**0.014**	0.022

Table A3: Ablation study of the scene *Mic* from the NeRF-Synthetic dataset, with all metrics presented. Number in bold highlights the highest values of the corresponding metric.

A.2.2 Ablation study on DeepVoxels Dataset

The ablation study done in the article isn't performed on the DeepVoxels [21] dataset. In this small part of the appendix, I performed the ablation study as done in the paper, looking at the influence of the three main contributions of the article, on one of the scenes of this dataset, the *Cube*. The conclusions are similar to the one obtained in the paper, which is why I didn't report these results directly in the main body of the report. The results obtained are presented in the table A4 below.

	$\alpha^{(init)(c)}$					No Pervoxel LR	Pre Activation
	1.10^{-2}	1.10^{-3}	1.10^{-4}	1.10^{-5}	1.10^{-6}		
↑PSNR	32.89	32.83	43.50	**43.53**	43.37	43.35	43.30
↑SSIM	0.981	0.981	0.998	0.998	0.998	0.998	0.998
↓LPIPS^{VGG}	0.020	0.019	0.002	0.002	0.002	0.002	0.002
↓LPIPS^{AlexNet}	0.014	0.013	0.002	0.002	0.002	0.002	0.002

Table A4: Ablation study of the scene *Cube* from the DeepVoxels dataset. Number in bold highlights the highest values of the corresponding metric. On this dataset again, as is the case for the other one presented in the paper, using view-count-based learning rate and post-activation improve performance. Choosing correctly $\alpha^{(init)(c)}$ is also important.

A.2.3 Influence of $\alpha^{(init)(f)}$

		$\alpha^{(init)(f)}$					No Pervoxel LR
		1.10^{-1}	1.10^{-2}	1.10^{-3}	1.10^{-4}	1.10^{-5}	*fine stage*
Chair	↑PSNR	33.98	34.15	34.04	33.95	14.04	34.13
	↑SSIM	0.977	0.976	0.975	0.975	0.846	0.976
	↓LPIPS^{VGG}	0.028	0.027	0.028	0.029	0.206	0.027
	↓LPIPS^{AlexNet}	0.016	0.016	0.017	0.017	0.298	0.016
Drums	↑PSNR	25.39	25.42	25.42	25.37	10.95	25.42
	↑SSIM	0.929	0.930	0.930	0.929	0.795	0.930
	↓LPIPS^{VGG}	0.079	0.078	0.080	0.079	0.256	0.080
	↓LPIPS^{AlexNet}	0.061	0.061	0.061	0.061	0.424	0.062
Ficus	↑PSNR	32.65	32.64	32.53	32.73	14.23	32.63
	↑SSIM	0.978	0.978	0.977	0.978	0.854	0.978
	↓LPIPS^{VGG}	0.025	0.025	0.025	0.025	0.155	0.025
	↓LPIPS^{AlexNet}	0.015	0.015	0.016	0.015	0.239	0.015
Hotdog	↑PSNR	36.31	36.73	36.70	36.46	10.39	36.68
	↑SSIM	0.978	0.980	0.980	0.979	0.834	0.980
	↓LPIPS^{VGG}	0.037	0.034	0.034	0.035	0.261	0.034
	↓LPIPS^{AlexNet}	0.019	0.017	0.017	0.018	0.366	0.017
Lego	↑PSNR	34.19	34.47	34.53	34.39	9.480	34.62
	↑SSIM	0.974	0.976	0.976	0.975	0.749	0.976
	↓LPIPS^{VGG}	0.030	0.028	0.027	0.028	0.295	0.027
	↓LPIPS^{AlexNet}	0.015	0.014	0.014	0.014	0.416	0.013
Material	↑PSNR	29.52	29.60	29.49	29.39	8.740	29.51
	↑SSIM	0.950	0.950	0.949	0.949	0.766	0.950
	↓LPIPS^{VGG}	0.060	0.059	0.060	0.060	0.263	0.059
	↓LPIPS^{AlexNet}	0.027	0.027	0.027	0.028	0.405	0.027
Mic	↑PSNR	33.14	33.13	33.11	33.07	13.04	33.19
	↑SSIM	0.983	0.982	0.982	0.982	0.881	0.983
	↓LPIPS^{VGG}	0.018	0.018	0.018	0.018	0.154	0.017
	↓LPIPS^{AlexNet}	0.014	0.014	0.014	0.014	0.298	0.014
Ship	↑PSNR	29.05	29.00	28.75	28.75	5.880	29.03
	↑SSIM	0.876	0.877	0.872	0.871	0.615	0.877
	↓LPIPS^{VGG}	0.162	0.161	0.167	0.167	0.396	0.162
	↓LPIPS^{AlexNet}	0.118	0.117	0.123	0.122	0.545	0.117
Avg	↑PSNR	31.77	31.89	31.82	31.76	10.84	31.90
	↑SSIM	0.955	0.956	0.955	0.954	0.792	0.956
	↓LPIPS^{VGG}	0.054	0.053	0.054	0.055	0.248	0.053
	↓LPIPS^{AlexNet}	0.035	0.035	0.036	0.036	0.373	0.035

Table A5: Influence of the parameters on the fine part of the model. Ablation study made on NeRF-Synthetic dataset - Complete results

A.3 Forward Facing scene

		$\alpha^{(init)(f)}$						Pre Activation
		1.10^{-1}	1.10^{-2}	1.10^{-3}	1.10^{-4}	1.10^{-5}	1.10^{-6}	$\alpha^{(init)(f)} = 1.10^{-2}$
Fern	↑PSNR	25.81	25.82	25.80	25.82	25.79	25.77	25.65
	↑SSIM	0.828	0.828	0.827	0.828	0.827	0.827	0.821
	↓LPIPS^{VGG}	0.237	0.236	0.239	0.237	0.239	0.239	0.246
	↓LPIPS^{AlexNet}	0.164	0.163	0.163	0.163	0.164	0.163	0.175
Flower	↑PSNR	27.96	27.92	27.90	27.93	27.95	27.92	27.83
	↑SSIM	0.861	0.861	0.861	0.861	0.861	0.861	0.857
	↓LPIPS^{VGG}	0.186	0.185	0.186	0.185	0.184	0.185	0.193
	↓LPIPS^{AlexNet}	0.121	0.121	0.121	0.120	0.120	0.121	0.131
Fortress	↑PSNR	30.70	30.76	30.77	30.75	30.70	30.70	30.65
	↑SSIM	0.876	0.878	0.877	0.878	0.877	0.877	0.875
	↓LPIPS^{VGG}	0.201	0.198	0.199	0.198	0.199	0.199	0.201
	↓LPIPS^{AlexNet}	0.115	0.111	0.112	0.111	0.112	0.111	0.117
Horns	↑PSNR	27.75	27.73	27.76	27.75	27.75	27.72	27.35
	↑SSIM	0.859	0.857	0.859	0.858	0.859	0.859	0.847
	↓LPIPS^{VGG}	0.252	0.255	0.252	0.253	0.253	0.250	0.270
	↓LPIPS^{AlexNet}	0.167	0.169	0.167	0.168	0.166	0.166	0.190
Leaves	↑PSNR	21.70	21.70	21.71	21.66	21.68	21.69	21.69
	↑SSIM	0.756	0.756	0.757	0.754	0.756	0.755	0.752
	↓LPIPS^{VGG}	0.216	0.216	0.216	0.218	0.217	0.218	0.225
	↓LPIPS^{AlexNet}	0.169	0.170	0.169	0.173	0.170	0.171	0.186
Orchids	↑PSNR	20.49	20.50	20.45	20.49	20.52	20.49	20.33
	↑SSIM	0.692	0.694	0.692	0.693	0.693	0.693	0.678
	↓LPIPS^{VGG}	0.247	0.246	0.245	0.245	0.246	0.246	0.269
	↓LPIPS^{AlexNet}	0.182	0.181	0.182	0.182	0.182	0.182	0.212
Room	↑PSNR	31.50	31.47	31.26	31.56	31.44	31.33	31.01
	↑SSIM	0.943	0.942	0.942	0.943	0.942	0.942	0.937
	↓LPIPS^{VGG}	0.239	0.242	0.241	0.239	0.242	0.241	0.252
	↓LPIPS^{AlexNet}	0.117	0.118	0.118	0.116	0.119	0.117	0.131
Trex	↑PSNR	26.84	26.85	26.86	26.85	26.87	26.77	26.17
	↑SSIM	0.899	0.900	0.899	0.899	0.900	0.899	0.882
	↓LPIPS^{VGG}	0.239	0.238	0.239	0.240	0.239	0.239	0.261
	↓LPIPS^{AlexNet}	0.111	0.111	0.111	0.111	0.111	0.112	0.136
Avg	↑PSNR	26.59	26.59	26.56	26.60	26.58	26.54	26.33
	↑SSIM	0.839	0.839	0.839	0.839	0.839	0.839	0.831
	↓LPIPS^{VGG}	0.227	0.227	0.227	0.226	0.227	0.227	0.240
	↓LPIPS^{AlexNet}	0.143	0.143	0.142	0.143	0.143	0.142	0.160

Table A6: Ablation study on the LLFF dataset, composed of forward facing scene - Complete results

A.4 Custom scene

Here I present some visual results on the custom data I used. They illustrate the importance of post-activation in the method.

(a) Ground Truth

(b) Prediction of the model with pre-activation

(c) Prediction of the model with post-activation

(d) Ground Truth

(e) Prediction of the model with pre-activation

(f) Prediction of the model with post-activation

Figure A1: Results on the TeddyBear scene. We can notice that the result is slightly blurry, but is still really good. In this case, the difference between the pre-activation and the post-activation setting is small.

(a) Ground Truth

(b) Prediction of the model with pre-activation

(c) Prediction of the model with post-activation

(d) Ground Truth

(e) Prediction of the model with pre-activation

(f) Prediction of the model with post-activation

(g) Ground Truth

(h) Prediction of the model with pre-activation

(i) Prediction of the model with post-activation

Figure A2: Results on the Panda scene. We can notice that the results are pretty blurred, with some “cloudy” shape at the front plane. In this case, the difference between the pre-activation and the post-activation setting is a bit more important. Note that the amplitude of movement is much more important than in the TeddyBear case.