Project report
Study of "Depth-supervised NeRF"

Theïlo Terrisse
theilo.terrisse@eleves.enpc.fr
Ecole des Ponts Paristech

1 INTRODUCTION

Neural Radiance Fields (NeRFs) are a declination of techniques used to render novel, photorealistic views of a scene from a sparse set of 2D views of that scene. This methodology spans a large set of applications, from content creation [10], to vision in robotics [2], to surface modelling from satellite images [5].

Prior approaches to new view synthesis include a variety of methods, from learning signed distance functions as deep neural networks to get implicit representations of scenes [8], to optimizing meshed representations thanks to differentiable pathtracers [3].

NeRFs learn an Multi-layer-perceptron (MLP) network to predict opacity and colour in every 3D point of the scene, then render each pixel of a new view using traditional volumetric rendering. Drawbacks of the original version of NeRF [7] include long training times, as well as the need for a large amount of input views. As a partial remedy to these issues, Kangle Deng *et al.* [1] suggest exploiting the depth information collected by the algortihms usually applied during dataset creation to estimate the pose of input views (*e.g.* COLMAP [9]). This depth information can be used to guide the learning of the MLP network as a simple guidance term in the loss function.

In this report, we introduce the original NeRF [7] and more particularly the new contributions of Kangle Deng *et al.* [1] in Section 2. In Section 3, we describe experiments performed to reproduce results of the article on a new hand-crafted dataset designed to include transparent objects, so as to assess potential limitations of the proposed depth supervision. The results are analysed in Section 4, and limitations and possible improvements are discussed in Section 5. Conclusion is given in Section 6.

2 METHOD

2.1 Neural Radiance Fields

Volumetric rendering. Neural Radiance Fields (NeRFs) [7] are a method used to generate novel views from a finite number of input views of a scene. To do so, NeRFs leverage traditional rendering techniques. Each pixel of an image corresponding to a view is generated by integrating the transmitted light along the ray $\mathbf{r}$ of direction $\mathbf{d}$ that extends from the center $\mathbf{o}$ of that view and passes through that pixel:

$$\hat{\mathbf{C}}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma(\mathbf{r}(t))\, \mathbf{c}(\mathbf{r}(t), \mathbf{d})\, \mathrm{d}t, \tag{1}$$

where $\hat{\mathbf{C}}$ is the rendered color, $\sigma(\mathbf{x})$ is the opacity at point $\mathbf{x}$, $\mathbf{c}(\mathbf{x}, \mathbf{d})$ is the color at point $\mathbf{x}$ and in direction $\mathbf{d}$ (to model non-lambertian objects), and $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\, \mathrm{d}s\right)$ is the accumulated transmittance along the ray up to the abscissa t. t_n and t_f are near- and far-boundaries delimiting the rendered content of the scene. In this expression, t parameterizes the ray as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$.

NeRFs as MLPs. The main contribution of NeRFs is to model the opacity and color by an MLP. More precisely, a first series of 8 fully-connected layers takes as input the position $\mathbf{x}$ and predicts the opacity $\sigma(\mathbf{x})$, along with a 256-dimensional feature vector. The latter is concatenated with the direction $\mathbf{d}$ of the ray and given to a last layer that predicts the color $\mathbf{c}(\mathbf{x}, \mathbf{d})$.

Hierarchical sampling. In practice, scenes are usually composed of vast volumes of empty space. Oversampling this space generates unnecessary computational overhead. To circumvent this, a common trick consists in performing hierarchical sampling by training two MLPs at once, one "coarse" and one "fine". To render a pixel from a ray, the coarse network starts by dividing the ray into K_C equal segments, then uniformly draws a sample abscissa t_k at random within each segment. Using a quadrature rule, the pixel is then rendered by

$$\hat{C}(\mathbf{r}) = \sum_{k=1}^{K} T_k\, (1 - \exp\left(-\sigma_k \delta_k\right))\, \mathbf{c}_k, \tag{2}$$

with $T_k = \exp\left(-\sum_{l=1}^{k-1} \sigma_l \delta_l\right)$ and $\delta_k = t_{k+1} - t_k$. Treating the weights $w_k = T_k\, (1 - \exp\left(-\sigma_k \delta_k\right))$ as weights of a probability distribution function (after proper normalization), the fine network is sampled at K_F new positions sampled according to this distribution, in addition to the K_C original positions. Therefore, the resolution is adapted in regions of expectedly high opacity.

Training procedure. To conclude on traditional NeRFs, the learning of a new scene proceeds as follows. For each scene, a new NeRF MLP has to be trained. A dataset for that scene consists in training and test views. Each view j is composed of an image I_j of the scene along with the pose $\mathbf{P}_j = [\mathbf{R}_j, \mathbf{T}_j]$ for that view, where $\mathbf{R}_j$ and $\mathbf{T}_j$ are the rotation and translation matrices that give the position and orientation of the camera within the world coordinates. A training iteration then consists in taking a batch $\mathcal{R}_{RGB}$ of B_{RGB} pixels (equivalently, rays) at random among all pixels of all views, rendering them with the two networks to get colors $C_C(\mathbf{r})$ and $C_F(\mathbf{r})$, then performing an optimization step for the following loss:

$$\mathcal{L}_{RGB} = \sum_{\mathbf{r} \in \mathcal{R}_{RGB}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]. \tag{3}$$

At the end of training, only the fine network is kept and can be evaluated on the test views, for instance using PSNR.

2.2 Depth Supervision

Limitations of NeRFs. Traditional NeRFs are known to come with two main limitations: they take very long to train, and they require a lot of input views of the scene to work. One reason for these limitations is that NeRFs have to implicitly solve the difficult problem of reconstructing the geometry of the scene. Solving this difficult problem is a key step in training the network, notably as it unlocks the benefits of hierarchical sampling by guiding positive weights w_k of the distribution given by the coarse network towards positions of actual surfaces within the scene.

Now, as NeRFs require an estimation of the pose of the input views, NeRF pipelines usually use Shape-from-Motion (SfM) software like COLMAP [9]. COLMAP takes as input a set J of images I_j (along with metadata) and outputs the poses $\mathbf{P_j}$ of the associated views, as well as a set $\mathbf{S}$ of N 3D key points in world coordinates, along with their 2D correspondences $\mathbf{x} = (\mathbf{x}_{ij})_{j\in J,\, i\in \mathbf{S}^j}$ (where $\mathbf{S}^j$ denotes the set of points of $\mathbf{S}$ visible on view j). Therefore, set $\mathbf{S}$ can be used to define a loss term that will guide the NeRF towards the geometry of the scene. Note that such a set of points can also be obtained with other techniques, such as RGB-D cameras.

Probabilistic interpretation of volumetric rendering. To define this loss, Kangle Deng *et al.* use a probabilistic interpretation of the rendering formula (1). In particular, they prove that, conditionally to assuming an opaque wall at t_f, the function $h(t) = T(t)\sigma(\mathbf{r}(t))$ is a probability distribution function. It can be interpreted as the probability that a ray terminates at depth t, so that (1) can be seen as the expectation $\mathbb{E}_{t\sim h}[\mathbf{c}(\mathbf{r}(t))]$. Now, the authors empirically observe that as a NeRF trains, this distribution tends to converge to a $\delta-$function centered in $\mathbf{D}$, the position of the closest opaque surface in the scene. Thereby, for each 2D key point $\mathbf{x}_{ij}$, they aim at guiding the distribution h_{ij} of the associated ray towards the $\delta-$distribution centered in the z coordinate of $\mathbf{X}_i$ in the coordinate system of view j, which we denote D_{ij}. This can be done using the KL divergence as a distance between distributions.

Handling depth uncertainty. In practice, tools such as COLMAP do not always give exact positions for sparse 3D keypoints. To assess this uncertainty, for a given point $\mathbf{x}_{ij}$ on a given view j, COLMAP uses the pose $\mathbf{P}_j$ and the intrinsic matrix $\mathbf{K}$ to reproject $\mathbf{X}_i$ on image I_j and gives a reprojection error $\sigma_{ij} = \|\mathbf{x}_{ij} - \mathbf{K}\mathbf{P}_j\mathbf{X}_i\|_2$. Hence the supervisory depth $\mathbf{D}_{ij}$ is replaced with a random variable $\mathbb{D}_{ij} \sim \mathbb{N}(D_{ij}, \hat{\sigma}_i)$, where $\hat{\sigma}_i = \frac{1}{|J|}\sum_{j=1}^{|J|}\sigma_{ij}$. Consequently, the target termination distribution of ray associated to pixel $\mathbf{x}_{ij}$ becomes $\delta(\cdot - \mathbb{D}_{ij})$.

Depth supervision loss. To summarize, for a ray associated to a given point $\mathbf{x}_{ij}$ on view j, the depth supervision consists in minimizing the KL divergence

$$\mathbb{E}_{\mathbb{D}_{ij}}\left[\mathrm{KL}\left[\delta(\cdot - \mathbb{D}_{ij})\middle|h_{ij}\right)\right] = \mathrm{KL}\left[\mathbb{N}(D_{ij}, \hat{\sigma}_i)\middle|h_{ij}\right] + const. \quad (4)$$

Averaging over rays $\mathbf{r}_{ij}$ from a random batch $\mathcal{R}_D$ (which may intersect $\mathcal{R}_{RGB}$ at a given iteration) of B_D key points $\mathbf{x}_{i,j}$ taken from all views, this amounts to minimizing the following term (later referred to as "KL loss"):

$$\begin{aligned}\mathcal{L}_D &= -\mathbb{E}_{\mathbf{r}_{ij}\in\mathcal{R}_D}\left[\int_{t=t_n}^{t_f}\log(h_{ij}(t))\exp\left(-\frac{\left(t-D_{ij}\right)^2}{2\hat{\sigma}_i^2}\right)\mathrm{d}t\right] \\ &\approx -\mathbb{E}_{\mathbf{r}_{ij}\in\mathcal{R}_D}\left[\sum_{k=1}^{K-1}\log(h_{ij}^k)\exp\left(-\frac{\left(t_k-D_{ij}\right)^2}{2\hat{\sigma}_i^2}\right)\delta_k\right]\end{aligned} \quad (5)$$

where $\delta_k = t_{k+1} - t_k$ and $h_{ij}^k = (1-\exp(-\sigma_k\delta_k))\exp\left(-\sum_{l=1}^{k-1}\sigma_l\delta_l\right)$, as in (2) (where σ_k and σ_l here refer to opacities, not reprojection errors).

In practice, the authors have implemented this loss, but use a simpler loss in their code, which consists in the mean square error between the supervision depth D_{ij} and the estimated depth $\hat{D}_{ij}$ (later referred to as the "MSE loss"):

$$\mathcal{L}_D = \frac{1}{B_D}\sum_{\mathbf{r}_{ij}\in\mathcal{R}_D}\left[\beta_{ij}(\hat{D}_{ij} - D_{ij})^2\right] \quad (6)$$

where

$$\hat{D}_{ij} = \mathbb{E}_{t\sim h_{ij}}[t] \approx \sum_{k=1}^{K} h_k t_k \quad (7)$$

and the β_{ij} are used to weight according to reprojection error:

$$\beta_{ij} = 2\exp\left(-\left(\frac{\sigma_{ij}}{\hat{\sigma}}\right)^2\right) \quad (8)$$

with $\hat{\sigma} = \frac{1}{N}\sum_{i=1}^{N}\hat{\sigma}_i$. The full training loss is then given by

$$\mathcal{L}_{RGB} + \lambda_D\mathcal{L}_D \quad (9)$$

with $\mathcal{L}_D$ chosen among the KL or MSE depth losses, and λ_D a regularisation hyper-parameter.

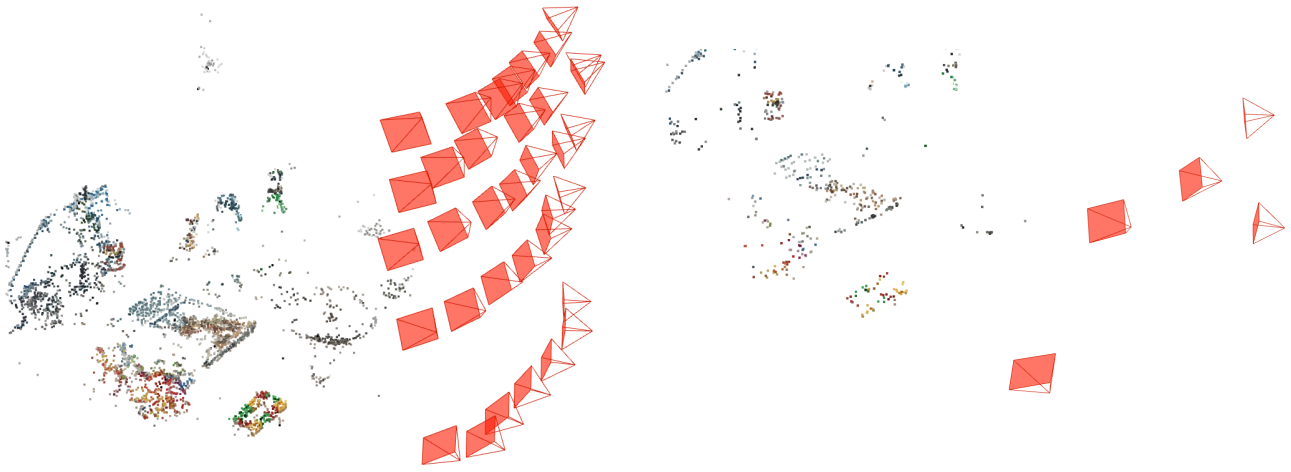
To conclude on the methodology, the authors also suggest that the loss may be used on other NeRF projects, as the only requirement for this is to inject the depth of COLMAP keypoints into the NeRF pipeline and to add the supervision term to the original RGB loss. As an example, they tested this approach by finetuning PixelNeRF [12] and IBRNet [11] with the added loss.

3 NEW EXPERIMENTS

Experiments objectives. A set of experiments were carried out to test the proposed method of [1], with the following objectives in mind.

(1) Reproducing the experiments of the authors on a new handcrafted dataset. Can we illustrate the acceleration in training as well as the performance boost for small numbers of input views, as announced by the paper?
(2) Comparing the two loss terms (5) and (6) proposed in the article.
(3) Exploring potential limitations of the DSNeRF method, by designing a dataset that contains many transparent objects. Does DSNeRF manage to reconstruct objects behind transparent surfaces?

This last objective is motivated by the following idea: the loss of the authors is designed to bring the ray distribution h closer to a unimodal distribution centered in the depth of the first encountered



(a) 35 views. **(b) 5 views.**

Figure 1: COLMAP sparse reconstructions of the scene.

obstacle. However, in the case of transparent objects, the distribution is precisely multi-modal; therefore, the design of the loss might be problematic in this case.

Dataset creation. Two sample views of the chosen scene are presented on Figure 2. The dataset is both simple (with clear near- and far-bounds and well-separated items), varied (with different shapes, colours and textures) and adapted to our objectives (since objects of various opacity and colours are placed in front of opaque surfaces). The dataset was created with a hand-held phone, with fixed exposition, white-balancing and focal length to ease subsequent sparse reconstruction by COLMAP.

Figure 2: Two sample views of the scene.

The scene was used to generate two datasets, one with 35 training views and 5 test views, and one with 5 training views and 4 test views. The two datasets are temporarily made available on Google Drive[1]. Note that in the NeRF training pipelines, these views will be resized to 25% of their original size to ease the computational burden.

For each dataset, COLMAP is used to determine the pose of the input views and get a set of sparse points. The result of this process is shown on Figure 1. For the 5-view dataset, 506 points are collected with a mean reprojection error value of 0.71px (with pixels expressed before resizing images); for the 35-view dataset, 6551 points are collected with a mean reprojection error of 1.12px.

Tuning the KL loss. In the design of the experiments carried out to test the various proposed loss terms, parameters for the chosen loss have to be tuned. In particular, the article leaves an ambiguity in a design choice for the KL loss (5). Indeed, in the argument of the exponential of the loss term, the numerator $t_k - D_{ij}$ is a depth expressed in COLMAP normalized units, while $\hat{\sigma}_i$ as defined by the authors is naturally expressed in pixels. Instead, $\hat{\sigma}_i$ should correspond to an error on the estimated depth, and not on the pixel coordinates of $\mathbf{x}_i$ on image j. Therefore, a choice has to be made to transition from an error in pixels to an error in COLMAP depth. This choice is of importance: too large a $\hat{\sigma}_i$ guides h_{ij} (with j the current view) towards a flat Gaussian distribution and tends to uniformize the ray distribution. Conversely, if the scaling for $\hat{\sigma}_i$ is too small, the weighting by the reprojection error becomes too stringent, in the sense that the loss takes a very large range of values depending on the key point, with very few key points having a sufficiently small reprojection error to reach the upper bound of this value range. As a consequence, only a small portion of key points are actually used for supervision, and the loss landscape is very ill-shaped, making the choice for λ_D particularly difficult.

Unfortunately, the article does not explain how to transition from errors in pixels to errors in COLMAP depth, and while the code does contain a class for the KL loss, it does not plug this loss into the rest of the NeRF training pipeline. As a compromise, two scales for $\hat{\sigma}_i$ were tested: in one case (later referred to as the "unscaled" case), for a ray i on image j, we simply replace $\hat{\sigma}_i$ with the weights β_{ij} proposed by the authors for the MSE loss. In the other case (later referred to as the "scaled" case), the conversion from pixels to COLMAP units is estimated by measuring an object of the scene both in *cm* (in the real, physical world) and in COLMAP

[1] https://drive.google.com/drive/folders/1aPhoAmgle8Ut9v33e-ullDFQc_cyjbK8?usp=sharing

Training id	1	2	3	4	5	6
# views	35	35	35	35	5	5
# iterations	100 000	100 000	100 000	100 000	150 000	150 000
Loss	RGB	RGB + MSE	RGB + unscaled KL	RGB + scaled KL	RGB	RGB + MSE
λ_D	–	0.1	10^{-13}	500	–	0.1

Table 1: List of NeRF trainings performed with associated loss parameters

unit (using point selection in the COLMAP GUI), then by knowing the length in *cm* of a pixel of the phone's camera used to take the input views. A conversion thus becomes possible from pixels to *cm*, then from *cm* to COLMAP unit; in the case of our scene, the scaling factor is about $1.2 \times 10^{-5} units/px$. Note that this is a very rough estimation; in fact, finding the exact dependency between reprojection error and depth estimation error would necessitate solving a problem as complicated as bundle adjustment itself. While other tuning strategies could be considered, we kept only these two options. For each option, to choose an appropriate λ_D and considering that no baseline is given by the authors, we proceeded as follows: typical values of $\mathcal{L}_{RGB}$ and $\mathcal{L}_D$ were estimated for a few batch of rays fed into an untrained NeRF, then λ_D was chosen to balance the two terms. Finally, typical values of the two losses were likewise estimated on a NeRF already trained on 35 views for 150 000 iterations without depth supervision, to check that the chosen λ_D would keep $\lambda_D \mathcal{L}_D$ in the range of values of $\mathcal{L}_{RGB}$ and would not deteriorate final performance.

Parameters for NeRF trainings. As a starting point, a NeRF network pre-trained by the authors for 100 000 iterations on 2 views of the "fern" scene from the NeRF Real dataset [6] was used to generate a video of that scene [2]. While the result is noisy, it constitutes a remarkably good reconstruction with only 2 views. After this preliminary test was carried out, in total, 6 new NeRF networks were trained with different choices for the dataset size and parameterization of the training loss, so as to reach the objectives of the experiments.

- 1 NeRF was trained for 100 000 iterations on 35 views without depth supervision, and served as a reference in the many-view setting.
- 3 NeRFs were trained for 100 000 iterations on 35 views with the RGB loss $\mathcal{L}_{RGB}$ and with 3 different depth supervision loss terms, either the MSE loss, the unscaled KL loss, or the scaled KL loss. The objective of these trainings is to investigate a potential gain in training time compared to the unsupervised NeRF, compare the loss terms and retain the one that works best for the few-view setting.
- 1 NeRF was trained for 150 000 iterations on 5 views without depth supervision and served as a reference in the few-view case.
- 1 NeRF was trained with the $\mathcal{L}_{RGB}$ and MSE supervision loss which, as we will see, is among the best performing supervisions for the many-view case.

This is summarized in Table 1 which also presents the choice for λ_D in each case. The code containing the various configurations is also available on a fork of the code of [1] created on GitHub[3]. Each training was performed on Google Cloud using a NVIDIA Tesla T4 GPU with a batch size of 4096 rays, a starting learning rate of 5×10^{-4} and an exponential learning rate decay with decay rate 0.1 and 250 000 decay steps. For hierarchical sampling, we take $K_C = 64$ for the coarse network and $K_F = 128$ for the fine network.

Metrics. For the metrics, we compute the average PSNR on the test scenes every 10 000 steps, and give the final PSNR values after training. We also give qualitative results by showing final rendered RGB and depth images for the test views presented on Figure 3 (one for the 35-view case and one for the 5-view case, since the test sets were different), as well as rendered videos, temporarily made available on a shared Google Drive[4]. The depth images are obtained by computing the $\hat{D}_{ij}$ from formula (7) at each pixel of the rendered image. Last, to explore the third objective of our experiments, for each training, we plot the distribution $(h_k)_{k \in K_C}$ for a ray that passes through a transparent object, namely the one associated to the pixels shown on figure 4.

Figure 3: Test views used for qualitative analysis (left: 35 views; right: 5 views).

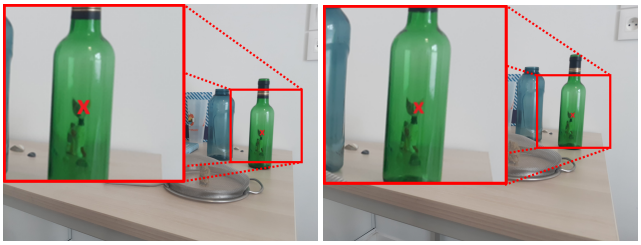


Figure 4: The pixels associated to the rays selected to plot termination distributions in the 35-view and 5-view cases.

[2] https://www.youtube.com/watch?v=F-TFkbHQbmo&ab_channel=TheiloT

[3] https://github.com/TheiloT/MVA-DSNeRF

[4] https://drive.google.com/drive/folders/1pRrQIwSNpDG5yeIoavpYT6rf5FImaZ1L?usp=sharing

4 RESULTS AND ANALYSIS

4.1 35-view setting

Quantitative results. On Figure 5a we plot the PSNR over iterations for the MSE loss, scaled KL loss and unscaled KL loss, with a NeRF trained without depth supervision as a reference. We get final average PSNR values of 34.4 for the unsupervised NeRF, 34.5 for NeRFs trained with MSE and unscaled KL losses, and 28.8 for the NeRF trained with scaled KL loss. This indicates that while the MSE and unscaled KL losses bring no particular improvements to the final results, the scaled KL loss actually deteriorates the performance. Moreover, Figure 5a also illustrates that depth supervision brings no acceleration in the training process, even for early steps of the training.

Qualitative results. On Figure 6a, we get similar conclusions, namely that the MSE and unscaled KL loss give comparable results to that of the unsupervised NeRF, while the scaled KL loss gives blurrier RGB images and depth maps. The cleanest depth map is obtained with the unscaled KL loss, while the MSE loss produces some depth artifacts, for instance around the neck of the bottles. This might be explained by some parasitic points observed floating to the right of the green bottle, as visible on Figure 1; the weights β_{ij} in (6) do not seem to make a good job at filtering these out. The cat behind the green bottle appears very blurry on all images, especially for the scaled KL loss where it is invisible because the bottle appears opaque. Although not visible here, but as seen on the videos, the cat is more visible behind the blue bottle in all cases, except the scaled KL loss where it is also invisible.

Ray termination distribution. On Figure 7a, the termination distribution $h(t)$ of the ray selected on Figure 4 is plotted. Note that the unit used for t is an arbitrary unit deduced from the COLMAP unit; in particular, it will be different for the 5-view case. As a control sample, we also plot on Figure 7b the distribution for a ray directed towards an opaque surface (namely, the surface of the table) and observe that the NeRF trained without depth supervision follows the ideal unimodal distribution expected by the authors of [1]. However, this is not the case anymore when considering the ray passing through the green bottle, where the distribution appears clearly multi-modal with two peaks, one at coordinate 2.2 and one at coordinate 3.4. This seems to correspond to the positions of the bottle and of the cat behind it. On the other hand, the three depth-supervised NeRFs produce mostly unimodal distributions centered in the approximate position of the bottle, with a shrunken peak in the position of the cat. The NeRF trained with the scaled KL loss gives a peak that is offset compared to the position of the peaks given by the other NeRFs. This might be explained by the weak performance and blurry depth given by this network, with the presence of "ghost" artifact volumes in the scene. These results indicate that depth supervision tends to overestimate the opacity of the first encountered object.

4.2 5-view setting

Quantitative results. Following the results obtained in the 35-view case, we exclude the scaled KL loss for further experiments. Due to limited resources, a full training could not be performed for all three remaining loss combinations. To solve this issue, trainings with few iterations were performed with the MSE and unscaled KL losses. Although not shown here, results with the unscaled KL loss showed poor performance, possibly due to an insufficient adaptation of the λ_D factor, which would have necessitated more time and resources. Figure 5b plots the PSNR every 10 000 iterations (with extra evaluations at 1 000, 2 000 and 5 000 iterations to account for the steep increase of early steps) for the MSE loss, with the unsupervised NeRF as reference. We observe a clear improvement brought by the depth supervision, as the supervised NeRF surpasses the final performance of the traditional NeRF even before the 1 000 first iterations. However, unsurprisingly, none of the trained NeRF attains the performance of the many-view case.

Qualitative results. As visible on Figure 6b, as expected, the unsupervised NeRF only recovers the main elements of the scene, and introduces many colour and depth artifacts that deteriorate the result; in particular, many "ghost" volumes appear and make the rendered view blurry. This phenomenon is particularly visible when looking at the animation, which the reader is encouraged to visualize. On the other hand, the MSE-supervised NeRF is able to reconstruct details in the geometry of the scene, although the depth map remains dirtier than in the many-view case. Additionally, the cat is surprisingly well rendered in the few-view case, more than in the many-view case. Looking at the animation hints that the NeRF is actually printing the colours of the cat on the surface of the bottle, as several cats can be seen (one on the bottle and one in the real figurine's position) depending on the point of view. This phenomenon was not observed in the many-view case, probably because in that setting the network is given multiple views of the surface of the bottle, which condition it to render the real color of this surface.

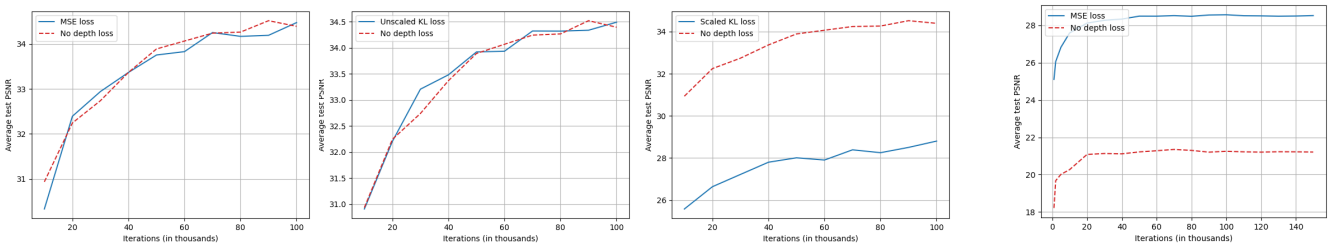
Ray termination distribution. In this case, as seen on Figure 7c, the ray distribution is less interpretable, as the units have changed compared to the previous case, and considering that the unsupervised NeRF yields a distribution that spreads over a large region, due to its poor performance. Nevertheless, the distribution of the MSE-supervised loss seemes peaked in what can be guessed as the front surface of the bottle, meaning that the network fails at creating a distribution with two modes. This is in adequacy with the observations made for the qualitative results.

5 LIMITATIONS AND DISCUSSION

5.1 Discussion of the results

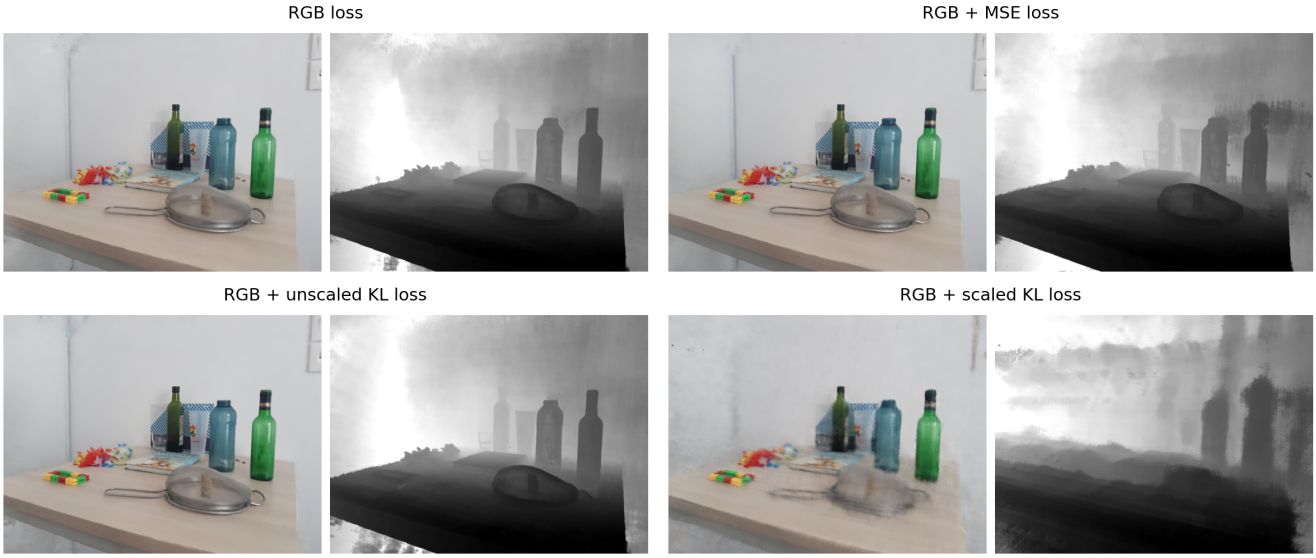
As an answer to the first objective of our experiments, it seems that there is no benefit in using depth supervision when many views are available. In fact, using a supervision loss may be counterproductive, in the sense that training takes longer in terms of wall time: for 35 views, while an iteration of traditional NeRF training takes $1.52s$ in average, it takes $1.58s$ when adding the MSE loss term, and up to $1.70s$ when using the KL loss. However, for few views, depth supervision is clearly beneficial as it significantly outperforms regular NeRFs in a few hundred iterations, which is consistent with the work of the authors of DSNeRF [1].

When it comes to comparing the loss terms, while the unscaled KL loss yields slight improvements in the many-view setting, the

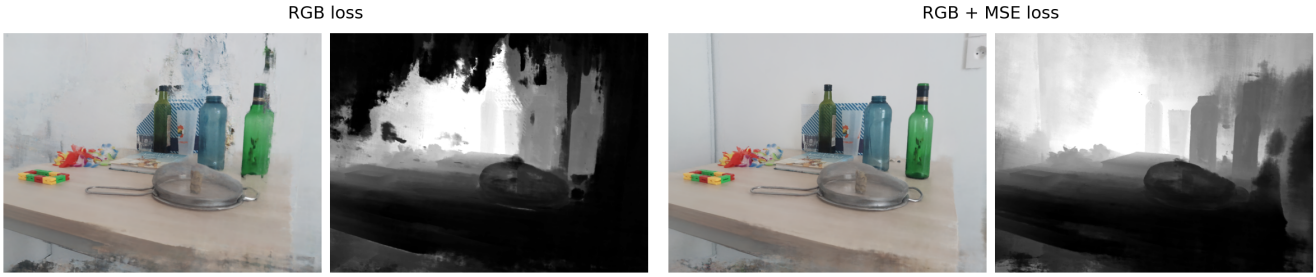


(a) 35 views. **(b) 5 views.**

Figure 5: Average test PSNR over iterations.



(a) 35 views.

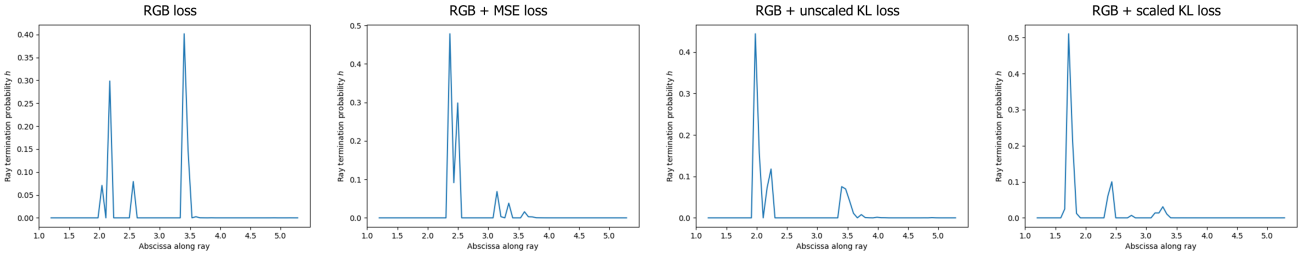


(b) 5 views.

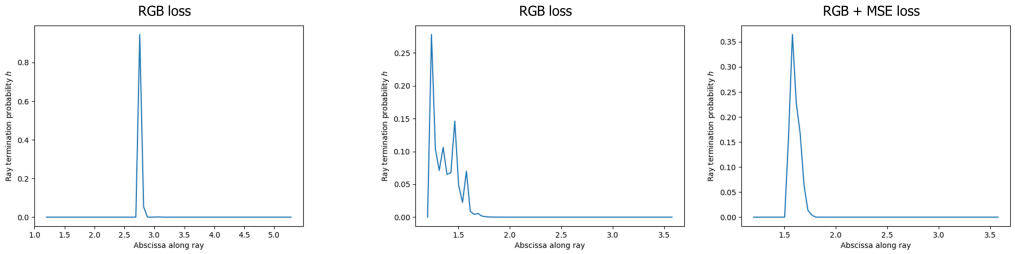
Figure 6: Rendered test view (RGB + depth map) for all trainings. Best viewed zoomed-in.

MSE loss remains the best choice in the general case. In particular, it is significantly easier to parameterize, as it was easy to get the right choice for λ_D contrary to the KL terms. This might also partly explain why the scaled KL loss gave such poor results, as already mentioned in Section 3. However, as detailed in Section 5.2 below, it seems reasonable that reprojection errors close to 1 will be associated with a better-behaved KL loss.

Last, the analysis of the ray termination distribution is made difficult by the fact that the distribution is often noisy, and that the COLMAP unit used along the ray is arbitrary. Nonetheless, depth supervision seems to guide the distribution towards a unimodal distribution, even when a multi-modal one is expected. However, this was never a noticeable issue in our study, because in the multi-view case, even the traditional NeRF failed at rendering the occluded cat,



(a) 35 views.



(b) 35 views (opaque surface). **(c) 5 views.**

Figure 7: Termination distributions for the ray of Figure 4 for all trainings.

while in the few-view setting, the cat was rendered by imprinting it onto the bottle's surface. In fact, the conclusions should also be mitigated by the fact that, especially in the 5-view case, few points were actually collected on the green bottle and on the cat, so that supervision in the considered region was only permitted indirectly by key-points of surrounding surfaces. In other words, while the depth supervision does seem to change the NeRF's behaviour as regards to transparent objects, we can hardly conclude on quantitative limitations of the supervision as regards to transparently occluded objects.

5.2 Limitations of the method

One main limitation of the paper under scrutiny is that it leaves out many ambiguities on the implementation of the proposed loss. A first ambiguity in formula (5) lies in the possible angular mismatch between the direction of the ray (along which t is defined) and the direction of the z-axis (which is used to define the depth D_{ij} of the keypoint, as given by COLMAP). In other words, the article introduces t as parameterizing the ray as $\mathbf{r}(t) = \mathbf{o}+t\mathbf{d}$, but in the loss, it uses t as if it were defined as a depth expressed in the coordinate system of the current view rather than an abscissa expressed along the considered ray. That is, if $\mathbf{p}$ designates the pixel coordinates of the point of the considered image, through which ray $\mathbf{r}$ passes, then, denoting $\bar{\mathbf{p}} = [\mathbf{p}, 1]^T$ the homogeneous coordinates of that point, $\mathbf{r}$ would be parameterized as $\mathbf{r} = t\bar{\mathbf{p}}$. For t evolving between fixed t_n and t_f, the two approaches are only equivalent if $\|\mathbf{d}\|_2 = \|\bar{\mathbf{p}}\|_2$, which is unclear as one could expect $\mathbf{d}$ to be of norm 1. Then, it is unclear how much of an approximation this would constitute. As for the code, we hypothesize that the formula is only correct because we are in the specific case where $\|\mathbf{d}\|_2 = \|\bar{\mathbf{p}}\|_2$ thanks to the way $\mathbf{d}$ is defined. Otherwise, to correct the formula, D_{ij} could be taken as the norm of point $\mathbf{X}_i$ in the coordinate frame of camera j.

Another ambiguity, already detailed in Section 3, comes from the transition from the COLMAP reprojection error $\hat{\sigma}_i$, expressed in pixels, to an error in the estimation of the depth, expressed in COLMAP units, as should be used in (5). This is important as the loss term seems to be particularly sensitive to $\hat{\sigma}_i$. More generally, little indication is given regarding the tuning of the loss term, and in particular λ_D. This tuning is all the more difficult as the code provided by the authors does not use the KL loss by default, but rather the MSE loss, so a value of λ_D was only suggested in a configuration file for this particular loss term. In fact, several adaptations and bugfixes had to be carried out to plug the actual loss of the article into the training pipeline, but also to get the software to work altogether.

When it comes to limitations of the overall approach, a practical limitation of the supervision comes from the fact that getting supervision key points constitutes a technical burden, either computational (such as solving a possibly heavy bundle-adjustement equation with COLMAP, which also requires favourable exposition and conditions of acquisition of views), or material (such as resorting to a kinect or a LiDAR camera). This may limit usage in the context, for instance, of casual photography. To do away with this set of limitations, certain approaches remove the need for giving estimates of the input poses altogether, such as BARF [4] which uses a coarse-to-fine approach to learn scene representations and resolve camera pose misalignment at the same time. Using such approaches would make it impractical to use DSNeRF, as no key points are needed to get the calibrated views. Yet, most of the benefits of DSNeRF appear when considering few-view settings, in which a method like BARF would probably misbehave. Therefore, again, the usage of DSNeRF should reasonably be restricted to the

few-view configuration. However, while depth supervision does improve results in this configuration when comparing to classical NeRFs, the synthesized views remain far from perfect, which questions whether DSNeRF could be used in real applications, at least in its form as presented in the paper of Kangle Deng *et al.*. Therefore, DSNeRF brings meaningful improvement, but further development needs to be done to allow using NeRFs with few views.

5.3 Possible improvements

Among possible paths for improving the method, we have already mentioned adjusting the possible angular mismatch between t and D_{ij} in (5).

Another possible improvement consists in fine-tuning a depth-supervised NeRF by removing the depth supervision after training. Indeed, while depth supervision can offer acceleration and improvement in the training process in the few-view setting, it might converge towards a results that is polluted with points of badly-estimated depth, as weighting the depth loss with $\hat{\sigma}_i$ (or β_{ij}, depending on the loss choice) might not do away with all of these incorrect points, depending on the scaling of the σ_{ij}. Removing the loss after training and fine-tuning for a few iterations might let the NeRF eliminate these imperfections.

Additionally, instead of simply guiding supervision using a loss term, one other way to enforce the guidance would be to directly bias the weights w_k used in hierarchical sampling (see equation (2)) to take higher values at depths close to the supervisory depth. Doing this at least for the early steps of the training procedure could guide the effort of the fine network towards an opaque surface of the scene, in a stricter fashion than by using loss supervision. This could further avoid sampling empty space during the training process.

6 CONCLUSION

In this report, we presented the depth supervision of NeRFs as proposed in the article "Depth-supervised NeRF: Fewer Views and Faster Training for Free" from Kangle Deng *et al.*. Doing so, we highlighted that two losses are actually proposed, which we referred to as the MSE loss (6) and the KL loss (5). We also pointed out an ambiguity in the transition from pixel reprojection errors to errors in the estimation of depth, which incited us to compare two variants of the KL loss. We then introduced a new dataset designed to include transparent objects, and used it to reproduce some of the experiments of the authors and to test the various losses in a many-view and a few-view settings.

While we did observe improved performance in the few-view case, we did not notice particular improvement in the many-view case. We also observed that the MSE loss tends to be the best performing and most practical loss term to choose, notably thanks to its ease of use, as the KL loss tends to be hard to tune in terms of scaling and choice of λ_D parameter. Lastly, while we did observe change in the behaviour of the NeRF in the presence of transparent surfaces, with a tendancy to adopt unimodal termination ray distributions when a multi-modal distribution would be expected, we could not quantify how detrimental this phenomenon could be for the final rendered view, as traditional NeRFs were not able to render the transparent surface and the transparently occluded object either.

Last, we identified some limitations of the method, notably in the writing of the article and code which lack clarity regarding certain aspects such as the design of the loss and the choice of hyperparameters. Regarding the approach itself, we highlighted that it brings significant improvements in the few-view setting, but warned that progress still needs to be made before using NeRFs with few views. Some options for improvement were suggested, such as directly guiding hierarchical sampling of the NeRF fine network using the supervisory depth.

ACKNOWLEDGEMENT

I would like to express my gratitude to the pedadogical team of the course "3D Point Cloud and Modeling" of the Master Mathématiques Vision Apprentissage and to the Google Cloud platform for granting the credits that funded the computational resources used in this project.

I would also like to thank Hugo Blanc for his advice and support in understanding the content of the studied article.

REFERENCES

[1] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-supervised NeRF: Fewer Views and Faster Training for Free. https://doi.org/10.48550/arXiv.2107.02791 arXiv:2107.02791 [cs].

[2] Justin Kerr, Letian Fu, Huang Huang, Yahav Avigal, Matthew Tancik, Jeffrey Ichnowski, Angjoo Kanazawa, and Ken Goldberg. 2022. Evo-NeRF: Evolving NeRF for Sequential Robot Grasping of Transparent Objects. https://openreview.net/forum?id=Bxr45keYrf

[3] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics* 37, 6 (Dec. 2018), 1–11. https://doi.org/10.1145/3272127.3275109

[4] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. 2021. BARF: Bundle-Adjusting Neural Radiance Fields. http://arxiv.org/abs/2104.06405 arXiv:2104.06405 [cs].

[5] Roger Marí, Gabriele Facciolo, and Thibaud Ehret. 2023. Multi-Date Earth Observation NeRF: The Detail Is in the Shadows. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Vancouver, BC, Canada, 2035–2045. https://doi.org/10.1109/CVPRW59228.2023.00197

[6] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. http://arxiv.org/abs/1905.00889 arXiv:1905.00889 [cs].

[7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. http://arxiv.org/abs/2003.08934 arXiv:2003.08934 [cs].

[8] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. https://doi.org/10.48550/arXiv.1901.05103 arXiv:1901.05103 [cs].

[9] Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. 2023. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*. 1–12. https://doi.org/10.1145/3588432.3591516 arXiv:2302.04264 [cs].

[11] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021. IBRNet: Learning Multi-View Image-Based Rendering. http://arxiv.org/abs/2102.13090 arXiv:2102.13090 [cs].

[12] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021. pixelNeRF: Neural Radiance Fields from One or Few Images. http://arxiv.org/abs/2012.02190 arXiv:2012.02190 [cs].