## Supplementary Materials of RING-NeRF

Doriand Petit<sup>1,2</sup>, Steve Bourgeois<sup>1</sup>, Dumitru Pavel<sup>1</sup>, Vincent Gay-Bellile<sup>1</sup>, Florian Chabot<sup>1</sup>, and Loïc Barthe<sup>2</sup>

<sup>1</sup> Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
<sup>2</sup> IRIT, Université Toulouse III, CNRS, France



**Fig. 1:** Different LODs Outputs of the model when only trained on the last level L = 7. We observe that, even without supervising intermediate LODs during the training, a notion of LOD is captured in the scene reconstruction. We also observe visually continuous LOD since, as expected, the level L = 3.5 outputs a 3D representation in between LOD L = 3 and L = 4 in term of details.

## 1 RING-NeRF

## 1.1 Architecture details

While our architecture presents the advantage of being quite simple and able to be adapted to different tasks, there are a few subtle technical choices that shouldn't be overlooked, especially the decoder's details.

As illustrated in the figure 2 of the article, the final feature after multiresolution combination is passed into a normalization layer to improve convergence stability. Novel View Synthesis experiments showed slightly better results when using standard, non-learnable normalization over the feature. Because a summation replaces the concatenation usually seen in grid-based architectures, the feature is shorter and could face problems for expressing high frequencies. It



Fig. 2: Different LODs Outputs of the model when only trained on the last level L = 7. This illustrates the LOD inductive biases even on more complex scenes than DTU single objects.



**Fig. 3:** Comparison of LOD when supervising (a) every level of detail or (b) solely the finest level of detail

is thus projected into a higher dimension space with a Random Fourier Feature mapping<sup>3</sup>. This consists in a learnable frequency filter, for which we chose a sinus filter : y = sin(Wx) with W a linear matrix (without bias). A linear layer transforms the filter output into both density (or SDF) and one color feature. This feature, concatenated with the direction of observation encoded into Spherical Harmonics, is fed to a MLP of 3 layers that predicts the radiance.

Following the baselines' protocol of the different evaluated tasks, we also did not use any appearance embedding for all the experiments.

<sup>&</sup>lt;sup>3</sup> M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, Ravi, J. Barron and R. Ng, Fourier features let networks learn high frequency functions in low dimensional domains, in: Advances in Neural Information Processing Systems, 2020.

#### 1.2 Differences with other architectures

Several works intended to modify the initial architecture of grid-based NeRF. However, our architecture differs from them in several respects which have a major impact on the architecture abilities.

A first noticeable difference is related to the notion of LOD that is induced by the architecture itself, independently to any supervision. This property is justified through the interpretation of the grid as a mapping function from the scene space to the decoder latent space and is also validated experimentally (see section 3.2 and figure 1). Except TriMip-RF [7] and LoD-NeuS [30] which possess such property but for tri-plan, other architectures need an explicit supervision to capture the notion of LOD (see ResidualMFN <sup>4</sup> for instance). This property is probably one of the main reason of the convergence robustness of RING-NeRF, as observed through few-view reconstruction and SDF reconstruction experiments. It also simplifies the setup of a coarse-to-fine reconstruction process and a distance-aware forward mapping without scarifying speed since no convolution [30] nor super-sampling [4] is implied. It is also to be noted that this architecture-induced property is especially designed by RING-NeRF and other close methods such as VR-NeRF [24] will not present intrinsic LOD as coherent as RING-NeRF, as it uses concatenation rather than sum of the multi-resolution features. Figure 4 shows how RING-NeRF's intrinsic LOD are more qualitative when using sum rather than concatenation of the features. Furthermore, we also demonstrate the decrease of performance the concatenation brings to the model with an example on the generalization to novel unseen observation distances.

A second difference is the invariance of the decoder latent space towards position (unlike NGLOD [19], Tri MipRF [7], etc, that concatenate Positional Encoding with the feature extracted from the grid) but also level of detail. LOD is exclusively encoded in the grid, unlike PyNeRF [21], NGLOD [19] or MFLOD [6] that use a per LOD decoder, or Zip-NeRF [4] and Mip-NeRF [2] which modify the feature depending on the LOD. This property facilitates the generability of the model as illustrated in section 4.5, since the learnt latent space is independent of the scale and position of the scene elements, and then favor the generalization of a (possibly) pre-trained decoder to new scenes.

Finally, the last specificity is related to the combination of the independence of the decoder latent space with respect to the size of the grid hierarchy, and the top-down representation of the LOD. As we confirmed experimentally (see section 4.5 of the article), this combination makes the maximal level of detail unbounded since the size of the grid pyramid can grow up dynamically (similarly to [12]) while keeping the access to the different LODs available (unlike [12]). This property is unique and cannot be expressed by both methods that use per-LOD decoders [19] and methods using concatenation of the multi-resolution features such as I-NGP [14] or VR-NeRF [24], as they cannot increase the size of the decoder's input.

<sup>&</sup>lt;sup>4</sup> S. Shekarforoush, D. Lindell, D. Fleet and M. Brubaker, Residual Multiplicative Filter Networks for Multiscale Reconstruction, in: Advances in Neural Information Processing Systems, 2022.



Fig. 4: Qualitative Ablative Experiment when using concatenation or sum of the features in the RING-NeRF architecture. (a) shows unsupervised LOD (L = 4) where, although the concatenation presents semi-coherent results, they are far less qualitative than the sum's results. (b) illustrates the impact on the full-scale training and 1/8th-scale testing experiment, with closer-to-ground-truth results with the full RING-NeRF architecture.

## 1.3 LOD Inductive Bias

As discussed in section 3, the inductive biases of the RING-NeRF architecture permits to naturally produce Level Of Detail of the reconstruction, even when we only use the full resolution images to supervise the full resolution LOD and not a pyramid of images supervising intermediate LOD like usual LOD NeRF architectures. This property is further illustrated on figure 1 that shows more examples on several scenes of the DTU dataset. By supervising all the LOD, we observe however a correlation between the LOD used to compute the image and the level of details in the images (see figure 3).

Other examples of unsupervised LOD in more complex scenes from the mip-360 dataset are shown in Figure 2.

## 1.4 Limitations of RING-NeRF

While simple, qualitative and efficient, we can still pinpoint a few limitations of our work which could be interesting to address in future research. First of all, in the method itself, our distance-aware forward mapping may not be as physically realistic as possible. In order to better compare the volume of the projected pixel at distance d with grids' cells, we chose to cast from the pixel a cubic cone and to extract a cube depending on the distance d. This means that, rather than sampling a true pyramid, we consider a leveled-pyramid and do not consider the growing size of the pixel inside the sampled cube. Hence, a more

realistic distance-aware LOD computation would compute this more complex shape's volume rather than consider this simpler cube.

Moreover, while we carefully chose our experiments to demonstrate as much as possible the capacity and potential of our method, we may not have gone as deep as possible to showcase the implications of our work. RING-NeRF is capable of performing dynamic resolution adaptive reconstruction. We demonstrated in section 4.5 that adding grids a posteriori improved the final reconstruction (while not damaging the previous ones). However, this property alone has very specific use case. In order to be useful for the most of situations, the architecture should be coupled with a carefully designed stopping criterion to determine the optimal resolution of the model. This rule should explicitly decide when does the quality/efficiency ratio reaches its maximum and then stop to increase the configuration. Moreover, for maximum performance, the stopping criterion should be local rather than global, in order to define a sparse grid hierarchy which adapts itself locally to the content of the scene. However, these extensions are considered out of the scope of this article and would need further research.

#### 1.5 Cone Casting with Scene Contraction

Scene Contraction Function. We use the scene contraction function from Nerfstudio's implementation [20], as defined originally in Mip-NeRF [2] :

$$contract(p) = \begin{cases} p & \text{if } ||p|| \le 1\\ (2 - \frac{1}{||p||}) \cdot \frac{p}{||p||} & \text{if } ||p|| > 1 \end{cases}$$
(1)

Following Nerfstudio implementation, we use  $L_{\infty}$  norm rather than  $L_2$ , as it bounds the position to a cube rather than a sphere, which is convenient with grid-based representations. Note that this contraction function bounds the scene into a cube of size 2, which is then reduced to a cube of size 1 for coding implementation reasons. This adds a factor 2 in the LOD computation formula, which is overlooked in the rest of the article for clarity reasons.

**Computation of LOD L in Contracted Space.** We defined in section 3.3 of the main article a formula to compute an appropriate LOD  $L \in \mathbb{R}^+$  based on the grid configuration, the sample position and the image resolution, such that :

$$(d.c)^{3}.det(J(p)) = \left(\frac{1}{f^{L}b}\right)^{3} \iff L = -\frac{\log\left(d.c.b.\sqrt[3]{det(J(p))}\right)}{\log\left(f\right)}$$
(2)

Based on the contraction function defined in 1, this becomes :

$$L = \begin{cases} -\frac{\log(d.c.b)}{\log(f)} & \text{if } ||p|| \le 1\\ -\frac{\log(d.c.b) + 2\log(2 - \frac{1}{||p||}) - 4\log(||p||)}{\log(f)} & \text{if } ||p|| > 1 \end{cases}$$
(3)

Impact of Contraction and Discussion. With this formula, we can first notice that, in the non-contracted space (inside the cube of size 1), the LOD L

does not depend on the contraction function, and its value decreases with the distance as expected.

However, further in the scene, inside the contracted space, we notice that the LOD L might increase with the distance (due to  $log(||p||^4)$ ), as illustrated in figure 5. While this can seem pretty non-intuitive at first, this has a logical explanation. Because of the contraction, the fixed volume of a grid cell will represent an increasing volume of the scene as the distance to the scene's center increases, up to an infinite volume. This clashes with the idea that the further our sample, the lower our chosen LOD needs to be. This duality is represented in the equation by the  $log(||p||^4) - log(d)$  part, and it shows that, at some point, the contraction function will always take the advantage on the distance term as our sample grow further from the center. Hence, whatever the position of the camera, if a ray continues far enough, the chosen LOD will increase at some point because of the contraction function. This also implies that the more resolute grids are used both for near objects and further areas, which would not have been the case if we did not consider the contraction function and which provides a more optimal use of the grid hierarchy with less unused parameters.



Fig. 5: Illustration of the effect of the contraction function on the chosen LOD. The depth is normalized and, in the LOD heatmap, the more intense the color, the higher the chosen LOD is.

## 2 New view synthesis - Section 4.2

## 2.1 Configuration

Our Novel View Synthesis and Anti-Aliasing experiments on the 360 dataset are done on one unique configuration for our baselines Nerfacto, ZipNeRF and PyNeRF and for our model RING-NeRF, both for the mono-scale and multiscale setups. We mostly used the configuration provided in the article ZipNeRF with few differences that we will underline in the following explanations. Each of these three baselines were trained for 25k iterations with a batch size of  $2^{16}$ rays. We use the RAdam Optimizer with an initial learning rate of 1e - 2, and a cosine-decayed scheduler over the whole training to 1e - 3.

Regarding the model itself, we also use a hash grid pyramid of 10 grids with a growth factor of 2, with grid resolutions ranging from 16 to 8192. Contrary to the

original Zip-NeRF, we use 8 features per level rather than 4, but with a similar hashmap size of  $128^3 = 2^{21}$  (however please note that the ZipNeRF experiments in this paper were done also with a feature size of 8). We chose this parameter to maximize the quality of every baseline, as it is compatible with our GPU. We also use proposal samplers introduced by mip-NeRF 360, and once again follow Zip-NeRF configuration : 2 rounds of sampling via 2 different grid-based proposal samplers (hashgrid pyramid + one linear layer), and one last forward process into the true model to generate the pixel values. Our 2 samplers have respectively 512 and 2048 max resolution and both use features of size 1 as the sampling process only needs density information. For simplicity of implementation and comparison, we actually use the usual I-NGP-based concatenation model for RING-NeRF's proposal samplers.

Following Nerfacto's architecture, we also used scene contraction (as described in section 1.5 of the supplementary materials, as well as the distortion loss in these experiments, as implemented in NerfStudio for Nerfacto, PyNeRF and RING-NeRF, and their adapted improved versions for ZipNeRF as presented in their article. We also did not use appearance embedding for any method as we also noticed a decrease in metrics when using it, as observed in Mip-NeRF 360 and ZipNeRF. Finally, for Nerfacto, PyNeRF and RING-NeRF, we did not use any additional mechanism such as the ones introduced in ZipNeRF (no novel interlevel loss, nor scale featurization, nor weight decay loss, nor Affine Generative Latent Optimization as appearance embedding, ...).

The same configurations are used for both mono-scale and multi-scale experiments.

#### 2.2 Result Analysis and Ablative Experiments

Mono-Scale Setups. In order to further analyze the separate impact of our architecture and of the associated mechanisms (distance-aware forward mapping and continuous coarse-to-fine), we run ablative mono-scale experiments. Our model is thus evaluated on 4 configurations with varying setups and the corresponding results can be found in table 1. First of all, we notice a gap between the results of Nerfacto (27.09, 0.779 and 0.181 for respectively PSNR, SSIM and LPIPS) and our method without the distance-aware mechanism. This means that the architecture in itself (described in section 3.2) enables better scene reconstruction than Nerfacto.

Moreover, the addition of the distance-aware mechanism still increases the performances significantly. While this idea was initially developed as an antialiasing process, this experiment demonstrates that it also benefits Novel View Synthesis in situations with little variations of distances of observation.

Finally, while our continuous coarse-to-fine brings slightly better results than when using the usual discrete coarse-to-fine, it still presents slightly lower results than when not using coarse-to-fine. While the difference is not prohibitive (especially in SSIM and LPIPS) and mainly due to randomness in the training process, this observation demonstrates that, even though using coarse-to-fine on harder setups is necessary to avoid catastrophic failure or improve consistency,

as explained respectively in section 4.3 and 3.2 of the sup. mat., it does not improve results in already stable setups for Novel View Synthesis.

Table 2 displays the results of our model against several other baselines, including Vanilla baselines (resp. NeRF, Mip-NeRF and Mip-NeRF 360). As expected, the main baselines PyNeRF, ZipNeRF and RING-NeRF all outperforms every Vanilla architectures, both in terms of quality and speed. Mip-NeRF 360 is the only vanilla method which performs close to our baselines, and also outperforms Nerfacto (although by being 50 times slower!).

Tables 4, 5, 6, 7, 8 and 9 shows per-scene results of our mono-scale experiments, evaluated on the whole pyramid of resolution (note that these tables present without coarse-to-fine results to give the most qualitative possible renderings).

Discussion on the metrics in Mono-Scale Training and Multi-Scale **Testing.** When evaluating the models on novel resolutions, RING-NeRF is the only method which produces coherent rendering without aliasing artifacts, as described in section 4.2. On the other hand, the other methods each behave quite differently (see table 1). ZipNeRF produces unintelligible renderings, and thus presents the lowest quantitative results, as illustrated in figure 4. However, while Nerfacto and PyNeRF both create similarly aliased but coherent renderings. their metrics (especially PSNR) are quite spread apart, with unexpectedly the distance-unaware Nerfacto method being better than PyNeRF. The explanation actually lies in the stability of both of these methods. As shown in figure 6, PyNeRF often produces unstable images with novel unsupervised observation distances and viewpoints. These coarser errors have a way larger impact on metrics than the aliasing artifacts from Nerfacto, which are very localized. While this is true for every evaluation metrics, PSNR is by far the most affected by this phenomenom, as RING-NeRF is only (in average on 1/2th, 1/4th and 1/8th res. on the entire 360 dataset) 3% better in PSNR than Nerfacto while they are 10% and 33% apart in respectively SSIM and LPIPS. On the other hand, PyNeRF and Nerfacto are 16%, 6% and 16% apart in resp. these 3 metrics (the RING/Nerfacto gap in PSNR is smaller than the Nerfacto/PyNeRF gap but higher in both SSIM and LPIPS).



Fig. 6: Examples of instability and aliasing artifacts on novel observation distances (renders at 1/8th resolution) and impact on the metrics. Respective PSNR, SSIM and LPIPS values of the corresponding image are inset.

 Table 1: Ablative study of RING-NeRF in Novel View synthesis performances for the Mono-Scale setup on the 360 dataset.

	Distance-Awar	e Coarse-To-Fine	$\mathrm{PSNR}\uparrow$	SSIM $\uparrow$	$\text{LPIPS}\downarrow$
Nerfacto	-	-	27.09	0.779	0.181
RING-NeRF	-	-	27.65	0.786	0.183
RING-NeRF	√	-	28.26	0.803	0.155
RING-NeRF	$\checkmark$	Discrete	28.06	0.799	0.158
RING-NeRF	$\checkmark$	Continuous	28.09	0.799	0.157

**Table 2:** Novel View synthesis performances for the full resolution images (trainingand testing) on the 360 dataset.

	$\mathrm{PSNR}\uparrow$	SSIM $\uparrow$	$\mathrm{LPIPS}\downarrow$	Training Time $\downarrow$
NeRF	23.85	0.605	0.451	12.65 h
Mip-NeRF	24.04	0.616	0.441	9.64 h
Mip-NeRF $360$	27.57	0.793	0.234	21.69 h
Nerfacto	27.09	0.779	0.181	$0.45 \ h$
PyNeRF	27.87	0.802	0.160	0.96 h
Zip-NeRF	28.06	0.808	0.154	1.10 h
Our Model	28.09	0.799	0.157	0.45 h

Multi-Scale Setup. Figure 7 illustrates how anti-aliasing models behaves with multiple distances of observations against the Nerfacto baseline. On one side, the aliased rendering of Nerfacto presents different types of artifacts at different resolutions. This is due to the NeRF's model property to "average" the distances of observations between the trained views. This thus results in under-contrasted image at full resolution, over-contrasted image at resolution 1/4 and aliasing artifacts at resolution 1/8. On the other hand, RING-NeRF takes into account the distance in the computation of the feature and succeeds in adapting to the different resolutions.

Per-scene results can be found respectively in tables 10, 11, 12, 13, 14 and 15.

## 3 Few Viewpoints Supervision - Section 4.3

#### 3.1 Configuration and Evaluation Details

We used the same training and evaluation protocols as FreeNeRF and its predecessors PixelNeRF and Reg-NeRF. We used 15 scans among the 124 existing of the DTU [9] dataset, their IDs being : 8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, and 114. In each scene, the images with the following IDs are the train views: 25, 22, 28, 40, 44, 48, 0, 8, 13 (the 3, 6 and 9 views setups respectively using the first 3, 6 and 9 images). The images with IDs in [1, 2, 9, 10, 11, 12, 14, 15, 23, 24, 26, 27, 29, 30, 31, 32, 33, 34, 35, 41, 42, 43, 45, 46, 47] are the evaluation images. We also downsample each training and testing view



Fig. 7: Comparison of renderings of the same viewpoint with 4 different resolutions for our architecture and Nerfacto. We observe that our solution is close to the ground truth while Nerfacto faces under-contrasted image at full resolution, over-contrasted image at resolution 1/4 and aliasing artifacts at resolution 1/8. PSNR values of the corresponding image at the different resolutions are inset.

by a factor 4, resulting in 300 x 400 pixels for each image. The masks used to evaluate the results are the ones used by FreeNeRF.

For cleaner renderings, we also decide on every of our Nerfstudio baselines (referring to the grid-based methods) to not consider any sample which is not observed by any training views when rendering novel views. This removes unnecessary noise as our model cannot imagine unseen areas.

Regarding the model, we follow FreeNeRF's choices as much as possible and, because there are no grid-based methods to compare ourselves to, the rest of the hyperparameters (mainly regarding the hashgrids) were derived from the configuration used in our Novel View Synthesis experiments. Hence, we followed the number of samples used in FreeNeRF (128 samples) and also disable the mechanisms of nerfacto dedicated to unbounded scenes : scene contraction and distortion loss. Our experiments on Nerfacto/Nerfacto+ and RING-NeRF are all done using a pyramid of 9 grids of growth factor 2, a configuration similar to our NVS experiments where we removed the last level of the highest resolution as we consider the object-centric DTU dataset simple enough to avoid to overcomplexify the model. We once again use features of size 8. Based on the same observation of the dataset being simpler than the 360 dataset, we used a smaller MLP than in NVS experiments, with a hidden dimension of 64, both for density and color. However, it is important to note that because our goal was rather to demonstrate the impact of our architecture on this task against the Nerfacto+ baseline, we did not focus our research on optimizing the configuration of the models, which could prove to further increase the metrics.

Finally, we follow FreeNeRF's density loss implementation and also use the introduced black and white prior, where we minimize the density of the M + 10 (5 more samples than in FreeNeRF's implementation) first samples rather than the M first when the value of the pixel is either black or white, in order to benefit from the particular form of the DTU dataset with uniform backgrounds.

**Table 3:** Ablative Experiments on reconstruction from few viewpoints (respectively 3, 6 and 9, separated by "/") on the DTU dataset. The reported metrics are computed based on the mask of the object.

	Density Loss	Coarse-to-fine	$PSNR\uparrow$	SSIM $\uparrow$	$\rm LPIPS \downarrow$
RING-NeRF	-	-	10.56 / 12.32 / 13.00	$0.618\ /\ 0.694\ /\ 0.732$	$0.341\ /\ 0.256\ /\ 0.252$
RING-NeRF	√	-	14.58 / 19.53 / 22.43	0.669 / 0.780 / 0.840	0.247 / 0.132 / 0.0901
RING-NeRF	-	Continuous	12.41 / 13.75 / 13.95	$0.604 \ / \ 0.670 \ / \ 0.742$	0.307 / 0.274 / 0.249
RING-NeRF	√	Discrete	15.79 / 20.16 / 22.93	0.706 / 0.785 / 0.847	0.201 / 0.127 / 0.085
RING-NeRF	√	Continuous	16.18 / 20.47 / 23.19	$0.713 \ / \ 0.808 \ / \ 0.847$	0.200 / 0.127 / 0.085

#### 3.2 Ablative Experiments

In order to better estimate the increase of stability brought by our architecture, we further developed the ablative experiments on the complex few images setup. Table 3 shows the results of different RING-NeRF versions on respectively 3, 6 and 9 images with the 15 previously stated DTU scenes. We evaluate the impact of the coarse-to-fine process (both our continuous version and the original version) and of the density loss introduced by FreeNeRF.

First of all, the results without both of these mechanisms are, as expected, quite low. However, we notice that its results are in average better than the basic Nerfacto architecture, especially when training with 6 and 9 views. While both are low results, we see qualitatively (see figure 8) that the sole RING-NeRF architecture mostly succeeds in creating a coherent 3D geometry, although not perfect and most of all very affected by floaters artifacts. This proves that our RING-NeRF architecture is more stable than the concatenation architecture by design, even though it is still lacking of a way to get rid of artefacts-inducing ambiguities.

We also evaluate the model both without coarse-to-fine and without the density loss separately. Using solely the density loss surprisingly results in correct metrics. However, the visualization demonstrates that the 3D reconstruction in itself is slightly less precise than when using only coarse-to-fine, although the metrics are better as they are more impacted by the artefacts. Moreover, the reconstruction without coarse-to-fine is less stable, with worse coherency in very hard setups, such as when observing the scene with only 3 views and with novel views very far from the training views, as illustrated in figure 9. Note that, to further demonstrate that the density loss is not the sole explanation of the results, figure 8 (b) shows the results of the nerfacto architecture coupled with the density loss, and we notice that, while a beginning of 3D consistency appears, it is still way more approximate than its RING-NeRF counterpart (e).

These experiments thus demonstrate that our model, coupled with continuous coarse-to-fine is able to create a coherent and qualitative 3D reconstruction, although surrounded by a huge amount of background misplaces, which have a huge impact on the metrics. The simple density loss is an efficient solution to counter this issue, but without continuous coarse-to-fine, it will in return slightly degrade the reconstruction, as well as loose in stability. The combination of both is thus the best trade off between 3D reconstruction and NVS quality.



Fig. 8: Results of the Few View Ablative Experiments. (a) nerfacto (b) nerfacto w/ density loss (c) RING-NeRF w/o both density loss and continuous coarse-to-fine (d) RING-NeRF w/ continuous coarse-to-fine and w/o density loss (e) RING-NeRF w/ density loss and w/o continuous coarse-to-fine (f) RING-NeRF w/ both continuous coarse-to-fine and density loss. The model is trained on 9 images.



Fig. 9: Example of Inconsistency when removing the continuous coarse-to-fine from RING-NeRF. (a) RING-NeRF w/ density loss (b) RING-NeRF w/ continuous coarse-to-fine and density loss. The model is trained with 3 images that are far from the evaluated view.

## 4 SDF Reconstruction - Section 4.4

#### 4.1 Choice of the baselines and Implementation

We chose NeuS-facto and Neuralangelo as main comparing baselines. The first one is considered a fast and efficient Python SDF-based implementation and the latter is considered state-of-the-art for surface reconstruction methods.

While NeuS-facto is solely implemented in the Sdfstudio [26] framework (which derives from Nerfstudio), Neuralangelo's code has been officially released and also possesses a Sdfstudio implementation. It is to be noted that, while the specificities of the article are coded in a similar fashion, both implementations make many different choices on NeRF specificities. For instance, a major difference between both frameworks is the way they sample rays at each iteration. While SdfStudio randomly chooses pixels among all the images (as in Nerfstudio), Neuralangelo follows I-NGP framework by sampling every pixels of n images (n being the batch size). Both solutions are valid but they enable different behaviors. In order to harmonize as much as possible our different experiments, we decided to use the Sdfstudio implementation, as the major part of the framework is common with Nerfstudio, which was used in the other experiments.

#### 4.2 Configuration Details

Our experiments are all done on the Sdfstudio framework, using the provided replica subset dataset. We use all the scenes, except the scan5, for which we experienced failure of training for every method without initialization, and expect the darker environment to heavily complexify the reconstruction in an already hard initialization-lacking setup. The estimated depths are provided by Sdfstudio and predicted via a pre-trained OmniData<sup>5</sup> model, they serve as an indication rather than true ground truths.

We mostly follow the basic Neuralangelo configuration implemented by Sdfstudio. This results in 16 levels of resolutions 32 to 4096 with features of size 8 and a hashmap size of 2<sup>19</sup>. The main differences regards the scheduling of the training, initially implemented to last 500k epochs. However, we noticed a similar convergence when scaling down the scheduling parameters for the training to last 100k epochs. Hence, we decided to evaluate all three methods (RING-NeRF, NeuS-facto and Neuralangelo) on 100K epochs (with accordingly scaled down schedulers and coarse-to-fine duration). Because the replica dataset is a synthetic indoor dataset, we remove background MLPs, appearance embedding and scene contraction from all evaluated baselines.

In the following additional experiments, we provide more results without initialization, as well as results with correct scene-specific initialization of the scene using SDF inverted spheres. Note that because RING-NeRF does not input the position on its MLP, it results in a different way to code the initialization. NeuS-facto and Neuralangelo both hard code the SDF inverted sphere in the weights of the MLP, benefiting from the position input while we initialize the model randomly and then overfit the network for 1000 epochs to the required initialization scheme (inverted sphere) before beginning the "true" training on the scene.

#### 4.3 Results

Additional results of SDF training without initialization on several Replica scenes can be found in figure 15. While most training of NeuS-facto without initialization results in catastrophic failure, some scenes still succeed in producing a coherent reconstruction. The scan 6, shown in figure 15, is one of those scenes, and while NeuS-facto avoids failure, the results are still way noisier than our RING-NeRF both in RGB and depth, and results in way lower PSNR metrics. Similarly, RING-NeRF without coarse-to-fine will often face failure, and therefore needs its continuous coarse-to-fine mechanism to forego the initialization. **Reconstruction Results.** Figure 10 shows some meshes obtained from the models with random initialization. As expected, the NeuS-facto mesh is com-

pletely useless, with the cloudy artefacts from the model resulting in opaque matter. On the other hand, both NeuralAngelo and RING-NeRF produces coherent meshes. However, the NeuralAngelo mesh is very blurry and contains very little details, while ours is way more precise, although rather noisy. These qualitative results are coherent with the previously described results.

<sup>&</sup>lt;sup>5</sup> A. Eftekhar, A. Sax, J. Malik, and A. Zamir, Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In Proc. of the IEEE International Conf. on Computer Vision (ICCV), 2021



Fig. 10: Resulting Meshes for models without initialization. Chamfer Distances in cm are inset.

**Results with initialization.** While not the subject of the initial experiment, it is interesting to compare those results with their correctly initialized counterparts. Figure 11 gives an example on one scene of the dataset.

First of all, our approach presents similar results whether we initialize the model or not. This further proves that RING-NeRF is robust to the initialization process. We can notice a slight difference in the PSNR metric which can be considered within the randomness interval of two reconstruction reruns of the same model and scene.

NeuS-facto presents an expected behavior as it does not face catastrophic failure anymore and succeeds in obtaining coherent RGB renders. However, the depth remains a bit blurrier than RING-NeRF and the final PSNR results on the associated RGB images are thus affected and remains slightly lower than RING-NeRF.

Regarding Neuralangelo, initializing the model improves both the RGB metrics and the depth estimation. However, the depth is still surprisingly blurry, which in turns prevents the RGB renders to overcome RING-NeRF results. This may be caused by the curvature loss, which is supposed to smooth the reconstruction. Because RING-NeRF introduces a bit of noise in the reconstruction, combining thoses two contributions could be an interesting research focus for robust and smooth SDF reconstruction.



Fig. 11: Example of results with initialization.

**Results on real data.** We demonstrated that the scene-specific initialization was crucial for other methods on the Replica dataset, contrary to our RING-NeRF. Replica being a perfect synthetic dataset, this further highlights the importance of the initialization in SDF representation. However, we also performed this no-initialization experiment on more complex real data such as the Tanks and Temples dataset. Figure 12 shows results on one such scene ("Meeting Room"). As expected, while NeuS-Facto faces catastrophic failure, both NeuralAngelo and RING-NeRF succeeds in reconstructing the scene, although with varying issues. NeuralAngelo is both blurrier and with more coarse errors (eg. the ceiling's beams) while RING-NeRF faces some noise. The latter however presents better global results as confirmed by the PSNR values which are even out on a test images subset of the scene.



Fig. 12: Depth prediction of SDF reconstructions without initialization on one Tanks and Temple scene. Note that NeuS-Facto fails to reconstruct the scene. PSNR mean values on the test set are inset.

## 5 Resolution Extensibility - Section 4.5

# 5.1 Demontration of the resolution extensibility on an unbounded complex scene.

Since the resolution extensibility is an intrinsic property of RING-NeRF, the experiment can be reproduced on any scene or configuration. Here, we demonstrate the property on a more complex scene of the 360 dataset, namely Garden. We follow the same experimental protocol described in section 4.5 with an increased hierarchy (6 levels from 16 to 512 max resolution). We begin the reconstruction with only the first three levels, train both the grid and the decoder to convergence and then freeze both of them. We then proceed to train to convergence one novel grid at a time, freezing the previous one at convergence. Figure 13 shows the final reconstruction at different output of the grid hierarchy. We notice that adding a grid always improves the reconstruction (with an increasing PSNR). This showcases both the capacity of our model to reconstruct finer details after the decoder's training and its ability to keep the coarser LOD valid.



Fig. 13: Resolution Extensibility on the Garden scene of the 360 dataset. Apart from the first grid, each novel grid is trained alone, with both the previous grids and the decoder frozen.

#### 5.2 Discussion on the Utility of the resolution extensibility.

While resolution extensibility has not been heavily studied in previous works, it actually is a crucial property in several different use cases and applications. On one side, this can be useful for better compression of NeRF models. An important issue in grid-based NeRFs is the choice of hyper-parameters, and more especially the maximum resolution of the grid pyramid. Being able to dynamically change the resolution of your model gives the possibility to dynamically choose the optimal local maximum resolution during training depending on the scene's complexity, therefore discarding any useless parameters for a more optimal model. On the other side, the usefulness becomes obligation for embedded systems with important resources limitations. We can for instance imagine two types of limitations in SLAM situations, where the robot is moving without interruption:

- Time limitation : Because it is constantly discovering new environment to reconstruct, it does not have enough time to reconstruct with precision the previously seen areas. However, RING-NeRF will let the robot optimizes with higher resolution previous areas later on when it will have more resources available.
- Memory Limitation : As the scene to reconstruct is of unknown size and the system possesses limited memory, it can not afford to reconstruct the entirety of the scene with maximum precision. A more viable strategy would be to reconstruct coarsely the majority of the scene and only reconstruct finer details on important areas (whether the most looked at places or those with the most objects for instance).



 ${\bf Fig. 14:} \ {\rm Few} \ {\rm Views} \ {\rm experiments} \ {\rm examples} \ {\rm on} \ {\rm different} \ {\rm scans} \ {\rm of} \ {\rm DTU} \ {\rm dataset}.$ 



Fig. 15: Examples of Depth Prediction of different SDF-based methods while foregoing the initialization in different Replica scenes. PSNR values are the average values of the evaluation images of the entire scene.

 Table 4: Novel View Synthesis PSNR for the Mono-Scale setup per scene on the 360

 Dataset (outside scenes).

$PSNR \uparrow$		Bic	vcle			Flov	vers			Gar	den			Stu	mp			Tree	ehill	
Res	x1	x2	x4	x8	x1	x2	x4	$\mathbf{x8}$	x1	x2	x4	x8	x1	x2	x4	x8	x1	x2	x4	x8
Nerfacto	24.75	25.84	23.84	22.08	21.43	22.54	21.99	20.26	27.05	26.41	23.91	22.20	26.11	23.37	23.38	22.90	22.71	24.99	24.58	23.41
PyNeRF	25.46	22.95	22.37	21.25	21.93	17.73	16.65	16.55	28.55	22.96	21.75	20.71	26.40	21.65	21.40	20.87	23.28	18.50	17.89	17.60
Zip-NeRF	25.19	15.09	14.48	12.55	21.84	14.96	9.59	7.58	27.63	14.86	15.31	13.99	26.84	19.09	13.19	9.67	23.42	17.42	11.88	9.67
Our Model	25.26	26.11	25.63	24.29	21.91	23.35	22.89	21.30	28.03	27.14	25.32	23.50	26.69	26.20	25.82	24.59	23.12	24.93	25.19	23.92

**Table 5:** Novel View Synthesis PSNR for the Mono-Scale setup per scene on the 360Dataset (inside scenes).

$PSNR \uparrow$		Ro	om			Cou	nter			Kite	hen			Bor	nsai	
Res	x1	x2	x4	x8	x1	x2	$\mathbf{x4}$	$\mathbf{x8}$	x1	x2	x4	x8	x1	$\mathbf{x}^2$	x4	$\mathbf{x8}$
Nerfacto	31.59	29.77	29.44	26.58	26.33	27.18	25.79	23.81	30.54	29.70	25.95	23.46	33.37	31.45	28.23	25.24
PyNeRF	32.12	24.51	23.52	22.63	27.76	19.38	18.76	18.52	32.56	25.35	24.25	22.86	32.80	24.41	23.73	22.48
Zip-NeRF	32.42	17.07	10.56	8.36	28.49	17.12	10.79	8.19	31.76	19.58	14.45	12.70	34.96	14.01	6.64	4.20
Our Model	32.43	28.02	26.00	24.29	29.06	27.73	25.27	24.36	32.66	29.42	27.42	26.88	35.15	31.68	28.83	26.32

 Table 6: Novel View Synthesis SSIM for the Mono-Scale setup per scene on the 360
 Dataset (outside scenes).

SSIM $\uparrow$	Bicycle					Flo	vers			Gar	den			Stu	mp			Tree	hill	
Res	x1	x2	x4	x8	x1	x2	x4	x8	x1	x2	x4	x8	x1	x2	x4	x8	x1	x2	x4	$\mathbf{x8}$
Nerfacto	0.694	0.774	0.662	0.560	0.579	0.661	0.645	0.551	0.819	0.767	0.632	0.554	0.730	0.613	0.555	0.481	0.590	0.698	0.683	0.647
PyNeRF	0.726	0.734	0.682	0.613	0.592	0.535	0.538	0.543	0.847	0.748	0.638	0.582	0.752	0.637	0.598	0.533	0.632	0.602	0.599	0.586
Zip-NeRF	0.730	0.237	0.156	0.073	0.604	0.581	0.451	0.364	0.845	0.319	0.303	0.251	0.763	0.687	0.471	0.259	0.636	0.654	0.495	0.408
Our Model	0.726	0.768	0.786	0.738	0.601	0.672	0.683	0.628	0.852	0.817	0.751	0.681	0.759	0.731	0.717	0.646	0.610	0.692	0.761	0.755

**Table 7:** Novel View Synthesis SSIM for the Mono-Scale setup per scene on the 360Dataset (inside scenes).

SSIM $\uparrow$		Ro	om			Cou	nter			Kite	hen			Bor	nsai	
Res	x1	x2	x4	x8	x1	x2	$\mathbf{x4}$	$\mathbf{x8}$	x1	x2	$\mathbf{x4}$	$\mathbf{x8}$	x1	x2	x4	$\mathbf{x8}$
Nerfacto	0.911	0.874	0.876	0.827	0.844	0.861	0.808	0.752	0.907	0.869	0.706	0.606	0.940	0.922	0.841	0.749
PyNeRF	0.919	0.827	0.780	0.747	0.873	0.649	0.585	0.567	0.930	0.837	0.745	0.677	0.946	0.852	0.796	0.727
ZipNeRF	0.929	0.769	0.543	0.438	0.885	0.723	0.512	0.406	0.922	0.805	0.637	0.519	0.961	0.593	0.251	0.189
Our Model	0.921	0.899	0.870	0.836	0.876	0.859	0.821	0.765	0.926	0.871	0.808	0.756	0.954	0.928	0.879	0.830

Table 8: Novel View Synthesis LPIPS for the Mono-Scale setup per scene on the360 Dataset (outside scenes).

$LPIPS \downarrow$		Bic	ycle			Flo	wers			Gar	den			Stu	mp			Tre	æhill	
Res	x1	x2	$\mathbf{x4}$	$\mathbf{x8}$	x1	$\mathbf{x}^2$	x4	$\mathbf{x8}$	x1	x2	$\mathbf{x4}$	$\mathbf{x8}$	x1	x2	$\mathbf{x4}$	$\mathbf{x8}$	x1	$\mathbf{x}2$	x4	$\mathbf{x8}$
Nerfacto	0.212	0.167	0.281	0.331	0.259	0.191	0.259	0.332	0.121	0.183	0.291	0.338	0.192	0.270	0.370	0.471	0.372	0.265	0.283	0.306
PyNeRF	0.196	0.176	0.236	0.287	0.282	0.278	0.341	0.341	0.096	0.183	0.288	0.349	0.160	0.243	0.334	0.480	0.305	0.299	0.310	0.320
Zip-NeRF	0.186	0.651	0.641	0.698	0.246	0.244	0.432	0.508	0.105	0.744	0.645	0.666	0.151	0.207	0.349	0.482	0.297	0.280	0.368	0.422
Our Model	0.184	0.161	0.141	0.164	0.236	0.166	0.159	0.209	0.090	0.112	0.159	0.197	0.168	0.193	0.218	0.246	0.308	0.258	0.187	0.199

Table 9: Novel View Synthesis LPIPS for the Mono-Scale setup per scene on the 360 Dataset (inside scenes).

$\mathrm{LPIPS}\downarrow$		Ro	om			Cou	nter			Kite	hen			Bor	ısai	
Res	x1	x2	x4	x8	x1	x2	x4	$\mathbf{x8}$	x1	x2	x4	x8	x1	x2	x4	$\mathbf{x8}$
Nerfacto	0.121	0.091	0.120	0.153	0.187	0.122	0.161	0.207	0.093	0.102	0.198	0.294	0.071	0.069	0.145	0.239
PyNeRF	0.111	0.194	0.208	0.207	0.149	0.293	0.309	0.322	0.076	0.123	0.182	0.262	0.069	0.114	0.166	0.245
Zip-NeRF	0.111	0.215	0.373	0.405	0.146	0.214	0.307	0.351	0.081	0.163	0.353	0.456	0.059	0.380	0.717	0.716
Our Model	0.120	0.087	0.082	0.086	0.150	0.109	0.105	0.119	0.080	0.094	0.120	0.155	0.056	0.062	0.109	0.130

Table 10: Novel View Synthesis PSNR for the Multi-Scale setup per scene on the 360 Dataset (outside scenes).

 $\mathrm{PSNR}\uparrow$ Bicycle Flowers Garden Stump Treehill Res x1 x2 $\mathbf{x4}$  $\mathbf{x8}$ x1 $\mathbf{x}^2$ x4x8  $\mathbf{x}\mathbf{1}$  $\mathbf{x}\mathbf{2}$ x4 $\mathbf{x8}$  $\mathbf{x}\mathbf{1}$  $\mathbf{x}^2$  $\mathbf{x}4$ x8 x1  $x^2$ x4 x8 Nerfacto 23.73 27.20 26.95 24.63 20.96 24.53 24.95 22.95 25.01 28.50 27.35 24.95 23.82 25.31 25.46 24.30 22.67 25.51 26.27 25.21 
 PyNeRF
 25.77
 28.44
 30.12
 30.11
 22.13
 25.80
 27.80
 28.35
 31.23
 32.02
 31.54
 27.21
 28.34
 29.87
 30.05
 23.23
 26.17
 27.98
 29.43

 Zip-NeRF
 24.85
 27.71
 29.08
 20.99
 21.51
 25.06
 27.37
 28.61
 27.03
 29.83
 31.38
 31.95
 27.84
 29.43
 29.55
 27.56
 28.56
 Our Model 25.17 27.94 29.21 29.91 21.72 25.27 27.20 28.20 27.40 30.00 31.22 31.70 25.97 27.81 29.05 29.81 23.28 26.18 27.82 28.49

Table 11: Novel View Synthesis PSNR for the Multi-Scale setup per scene on the 360 Dataset (inside scenes).

 $\mathrm{PSNR}\uparrow$ Room Counter Kitchen Bonsai x4  $\operatorname{Res}$ x2  $\mathbf{x8}$ x1 x2  $\mathbf{x4}$ x8  $\mathbf{x}\mathbf{2}$  $\mathbf{x}^2$ x1x4x1 $\mathbf{x8}$ x1x4x8 Nerfacto  $28.09\ 29.14\ 28.71\ 27.92\ 22.82\ 23.51\ 23.77\ 23.51\ 27.85\ 30.42\ 29.96\ 26.72\ 29.89\ 31.68\ 31.00\ 27.49$ PyNeRF  $31.73 \ \ 32.75 \ \ 32.71 \ \ 31.43 \ \ 26.89 \ \ 27.47 \ \ 27.72 \ \ 27.64 \ \ 30.68 \ \ 31.45 \ \ 31.89 \ \ 30.98 \ \ 32.74 \ \ 33.52 \ \ 33.63 \ \ 33.08$ Zip-NeRF 31.65 32.52 32.78 32.74 28.19 29.15 29.45 29.49 31.72 32.21 32.51 32.53 34.07 34.20 34.03 33.94  $\underbrace{ \text{Our Model} 31.74 \ 32.69 \ 32.86 \ 32.64 \ 28.63 \ 29.51 \ 30.12 \ 30.33 \ 31.17 \ 32.27 \ 33.15 \ 33.07 \ 33.87 \ 34.29 \ 33.97 } \\ \underbrace{ \text{Our Model} 31.74 \ 32.69 \ 32.69 \ 32.64 \ 28.63 \ 29.51 \ 30.12 \ 30.33 \ 31.17 \ 32.27 \ 33.15 \ 33.07 \ 33.87 \ 34.29 \ 33.97 } \\ \underbrace{ \text{Our Model} 31.74 \ 32.69 \ 32.69 \ 32.64 \$ 

Table 12: Novel View Synthesis SSIM for the Multi-Scale setup per scene on the 360 Dataset (outside scenes).

$SSIM \uparrow$		Bic	vcle			Flow	vers			Gar	den			Stu	mp			Tree	ehill	
Res	x1	x2	x4	x8																
Nerfacto	0.564	0.787	0.812	0.719	0.478	0.706	0.761	0.679	0.647	0.837	0.806	0.714	0.597	0.696	0.698	0.601	0.511	0.717	0.782	0.757
PyNeRF	0.710	0.840	0.891	0.896	0.587	0.764	0.836	0.842	0.827	0.887	0.921	0.910	0.745	0.815	0.853	0.843	0.622	0.769	0.845	0.882
Zip-NeRF	0.703	0.834	0.887	0.896	0.565	0.752	0.835	0.866	0.820	0.897	0.923	0.926	0.725	0.807	0.847	0.845	0.591	0.758	0.831	0.874
Our Model	0.716	0.840	0.881	0.890	0.575	0.753	0.829	0.861	0.822	0.890	0.916	0.927	0.726	0.791	0.823	0.827	0.594	0.762	0.845	0.883

Table 13: Novel View Synthesis SSIM for the Multi-Scale setup per scene on the 360 Dataset (inside scenes).

SSIM $\uparrow$		Ro	om			Cou	nter			Kite	chen			Bon	Isai	
Res	x1	$\mathbf{x}2$	x4	x8	x1	x2	x4	$\mathbf{x8}$	x1	$\mathbf{x}2$	x4	x8	x1	$\mathbf{x}2$	x4	x8
Nerfacto	0.826	0.861	0.874	0.872	0.715	0.739	0.769	0.769	0.794	0.873	0.845	0.755	0.876	0.908	0.897	0.825
PyNeRF	0.899	0.917	0.919	0.923	0.845	0.884	0.866	0.875	0.891	0.902	0.917	0.920	0.909	0.935	0.928	0.940
$\operatorname{Zip-NeRF}$	0.910	0.932	0.937	0.936	0.857	0.869	0.875	0.881	0.909	0.924	0.932	0.941	0.946	0.945	0.938	0.939
Our Model	0.907	0.932	0.937	0.937	0.858	0.880	0.897	0.910	0.900	0.912	0.922	0.936	0.939	0.944	0.938	0.941

 Table 14: Novel View Synthesis LPIPS for the Multi-Scale setup per scene on the 360 Dataset (outside scenes).

 $\mathrm{LPIPS}\downarrow$ Bicycle Flowers Garden Stump Treehill Res x2 x4 x8 r2 x4 **v**1 x2 x4 x8 **x**1 x2 . v4 **v**8 **v**1 **v**1 **x**1 x2 x4 Nerfacto 
 A1
 A2
 A1
 A2< 
 PyNeRF
 0.211
 0.000
 0.055
 0.052
 0.000
 0.055
 0.042
 0.011
 0.114
 0.102
 0.094
 0.119
 0.114
 0.022
 0.036
 0.119
 0.011
 0.012
 0.094
 0.036
 0.011
 0.010
 0.025
 0.010
 0.025
 0.010
 0.025
 0.010
 0.025
 0.011
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 0.012
 <t Our Model 0.217 0.103 0.067

Table 15: Novel View Synthesis LPIPS for the Multi-Scale setup per scene on the 360 Dataset (inside scenes).

$\rm LPIPS \downarrow$		Ro	om			Cour	nter			Kit	chen			Bor	nsai	
Res	x1	x2	x4	$\mathbf{x8}$	x1	x2	x4	$\mathbf{x8}$	x1	$\mathbf{x}^2$	$\mathbf{x4}$	$\mathbf{x8}$	x1	x2	x4	$\mathbf{x8}$
Nerfacto	0.320	0.163	0.071	0.093	0.438	0.288	0.166	0.176	0.258	0.113	0.092	0.176	0.233	0.112	0.085	0.160
PyNeRF	0.170	0.066	0.041	0.037	0.206	0.102	0.075	0.066	0.118	0.066	0.043	0.035	0.090	0.050	0.043	0.033
Zip-NeRF	0.141	0.064	0.036	0.024	0.188	0.118	0.064	0.061	0.098	0.064	0.043	0.029	0.079	0.050	0.045	0.037
Our Model	0.151	0.069	0.037	0.025	0.187	0.097	0.054	0.036	0.103	0.060	0.041	0.023	0.086	0.048	0.044	0.036