# Self-Camera Pose Estimation Neural Radiance Fields with Periodic Activation Functions for Novel View Synthesis

Thanh-Tung Phan-Nguyen
*Honors Program, Faculty of Information Technology*
*University of Science, Ho Chi Minh City, Vietnam*
*Vietnam National University, Ho Chi Minh City, Vietnam*
19120424@student.hcmus.edu.vn

Khoi-Nguyen Nguyen-Ngoc
*Honors Program, Faculty of Information Technology*
*University of Science, Ho Chi Minh City, Vietnam*
*Vietnam National University, Ho Chi Minh City, Vietnam*
19120106@student.hcmus.edu.vn

Nghia Vu-Huu
*Honors Program, Faculty of Information Technology*
*University of Science, Ho Chi Minh City, Vietnam*
*Vietnam National University, Ho Chi Minh City, Vietnam*
19120028@student.hcmus.edu.vn

***Abstract*—We propose Sinusoidal Neural Radiance Fields (SiNeRF), an end-to-end framework capable of generating novel views from a set of input images without knowing camera poses beforehand. In 2020, many methods representing scenes as Neural Radiance Fields have achieved state-of-the-art results in synthesizing views of real-world scenes. However, these methods still have their limitation, which is the dependency of camera poses for each input image. By using a multi-layer perceptron with periodic activation functions, we indicate that our framework can learn the camera poses along with network parameters during the scene optimization process. The produced poses are more accurate than those from both classical and neural-based methods in non-forward-facing scenes. We also show that using periodic functions as activators can optimize scenes with smoother depth than rectified linear activation, thus can synthesize decent quality views that achieve over 20dB in the PSNR metric even with low-res source images. In conclusion, our contribution is using periodic activation functions in optimizing Neural Radiance Fields to simultaneously estimate the camera poses.**

***Keywords*—Neural radiance field representation, volume rendering, novel view synthesis, camera poses estimation, multi-layer perceptron, sinusoidal function.**

## I. Introduction

The ability to travel around the world without changing geographical location has been achieved by recent advances in Virtual Reality and Augmented Reality. However, to store all scenes of our world needs an enormous amount of data centers, which results in resource waste and environmental pollution. Therefore, creating new realism views from a sparse set of input images is a long-standing problem in Computer Graphics. This problem can be resolved by solving two subproblems: reconstruct the 3D scene to a representation F and, from representation F, generate images from new angles.

Recent works in computer graphics and computer visions have led to a promising approach to represent 3D scenes. They use a fully connected neural network to map each 3D position to color, density and lighting properties from respective direction [1], [2]. Among them, Neural radiance field representation (NeRF) [3] has gained popularity and is considered as *State of the Art* because of its impressive quality. The classical volume rendering technique [4] is used to synthesize novel views in this representation. Despite its remarkable rendered views and simple architecture, NeRF still has some downsides. One of them is the dependence on other techniques such as Structure from Motion (COLMAP) to get access to camera parameters for further computing, and many other NeRF-based works also have this characteristic. The output quality of those methods heavily depends on the accuracy of estimated camera parameters, which leads to failures in cases COLMAP cannot produce camera positions.

A way to remove this feature is to initialize and

optimize camera parameters (both intrinsic and extrinsic) along with representation neural network during training time *(NeRF−−: Neural Radiance Fields without known camera parameters* [5]). This method resolves the dependence on other techniques and achieves comparatively results with the origin NeRF model. However, the optimized model still struggles with local minima and cannot reach the global minima of camera poses in some circumstances, as mentioned in [5]. In this work, we show that replacing the ReLU-based neural network in NeRF−−'s pipeline with a sinusoid-based neural network can overcome this limitation, thus improving output quality. We investigate that using sinusoid activation functions to register camera poses can also provide smoother optimized scenes than the ReLU one.

## II. Related Work

### A. *3D scene representation*

To reconstruct the 3D geometry from 2D images, techniques such as Structured-from Motion (SfM) [6] and Simultaneous Localisation and Mapping (SLAM) jointly solve for the 3D geometry and the associated camera parameters, by establishing feature correspondences (e.g. [7], [8]), or photometric errors, e.g. [9] and LSD-SLAM [10]. However, many of these methods assume diffuse surface texture and do not recover view-dependent appearance, hence resulting in unrealistic novel view rendering.

After that, with the announcement of Multi-view Stereo [11], combined with SfM, very impressive results can be yielded. The output 3D presentation of this technique can be divided into two categories:

- Mesh representation: The most popular and straightforward way to represent a 3D scene is a mesh comprised of many triangles. But for its simplicity, it can only be used to produce an object with a fixed topology.
- Volumetric representation: Volumetric approaches, such as point clouds and voxels, can perform very well in real-world scenes. However, these methods do not scale well as the resolution of the images increases [12].

Neural representation begun to rise in 2019 with various proposals [1], [2], [13]. However, most of them use only 3D location $(x, y, z)$ to produce output, thus result in limitations to the simple scene. And NeRF [3] was the first idea that gets success in this field with the use of a 5D vector (location + direction) to represent a 3D scene. It also uses very small space (1 - 5MB) to store the representation, which also contributes to NeRF's success.

### B. *Neural radiance fields representation and improvement*

Despite being *State of the Art*, NeRF still has many downsides. But with unceasing recent works to overcome those limitations, many impressive NeRF-based models have been announced so far:

- The optimization process of NERF that gains high-quality radiance fields is extremely long. This is expensive and not practical. The method for solving is MVSNeRF [11], which used deep MVS (Multi-view Stereo) and neural rendering, incorporating cost-volume-based scene reasoning into physically-based neural volume rendering. Another solution for this problem is mentioned in [14].
- The stringent sampling requirements and costly neural network queries cause slow rendering in NeRF. This can be improved by using PlenOctree for real-time rendering [15]. A sparse voxel-octree is used, whose leaves store the appearance and density values required to model the radiance at a point in the volume. In addition, the RGB values are represented at a location with spherical harmonics (SH), special functions defined on the surface of the sphere. Coefficients for the SH function are produced by training a network so that the predicted values can be stored within the leaves of the tree. However, this improvement requires large storage and memory footprint for the tree.
- NeRF handles very well with static scenes placed in controlled scenarios. But it's still not capable of rendering many real-world phenomena in uncontrolled images, such as photometric variations or transient objects. NeRF-W (NeRF in the Wild) applies a series of improvements to NeRF to solve these problems. The results of model testing on images of world landmarks show that the results obtained are much better than traditional NeRF. Recreated scenes show light and time closer to the real-world scenes than the prior State of the Art.

### C. *Sinusoidal representation network*

According to Sitzmann et al, 2020 [16], fully-connected networks using Rectified Linear Unit (ReLU) as activation function cannot learn high-frequency textures when representing object (sound, image, video) due to its simplicity, even when it combines with positional encoding [17]. They propose a new network architecture that uses

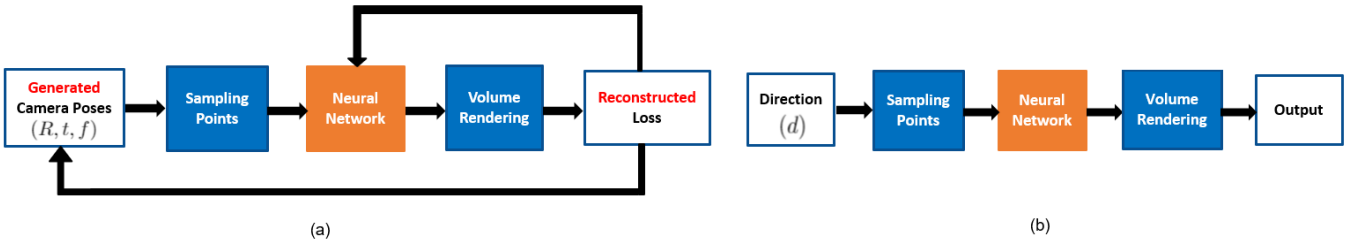


Figure 1: (a) Training stage, (b) Inference stage

sinusoidal activation functions, which can perform better than other activation functions on this problem.

## III. Method

Figure 1 shows our pipeline, which is the same as NeRF's, except for the auto-generated camera poses.

### A. *Camera parameters*

Assuming that all input images of size $W \times H$ are taken from a single pinhole camera, which is also used in original papers [3], [5], then:

The *intrinsic camera matrix* can be represented as:

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

$f_x, f_y$ represent the focal length of the camera in $x$ and $y$ axis, respectively. In a pinhole camera, $f_x$ and $f_y$ share the same value, so we will call that value $f$. The principal point offset $c_x, c_y$ are set to $\frac{W}{2}$ and $\frac{H}{2}$ by default. Hence, for the intrinsic matrix, we only have $f$ as a parameter to optimize. This parameter is shared among the source images.

Since we only care about how to transform a point from camera coordinate system to world coordinate system, *camera pose matrices* are used instead of extrinsic camera matrices, which are their inverse matrix. The *camera pose matrix* for each input image can be expressed by a rigid transformation matrix:

$$\left[\begin{array}{c|c} R & \boldsymbol{t} \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$R$ is a $3 \times 3$ matrix which can be interpreted as the direction of camera coordinate axes in the world coordinate. $t$ is a vector of length 3 representing the position of the camera origin in the world coordinate. This transformation matrix is also called a *camera-to-world matrix*, which we will use later to map each ray from input images to the world coordinate system. However, the rotation matrix $R$ can be simplified as a vector $r$ of length 3 in axis-angle representation:

$$r = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

From the axis-angle rotation vector $r$, we can find the unit vector $(k_x, k_y, k_z)$ and the rotation angle $\theta$, which is the magnitude of $r$. Then we can later convert them to a rotation matrix by using formula 2. As a result, we have $r$ and $t$ as parameters of each image for the optimization stage.

In short, camera parameters consist of the shared focal $f$, an axis-angle rotation matrix $r$, and a translation vector $t$ as the camera-to-world matrix for every input image.

### B. *Sampling points*

Firstly, with each of the provided images, we cast rays from its camera origin to all pixels to get the direction vectors $d$. The direction for a pixel at position $(x, y)$ of an image whose camera focal length is $f$ can be expressed

$$R = \begin{pmatrix} \cos\theta + k_x^2(1-\cos\theta) & k_x k_y(1-\cos\theta) - k_z\sin\theta & k_x k_z(1-\cos\theta) + k_y\sin\theta \\ k_y k_x(1-\cos\theta) + k_z\sin\theta & \cos\theta + k_y^2(1-\cos\theta) & k_y k_z(1-\cos\theta) - k_x\sin\theta \\ k_z k_x(1-\cos\theta) - k_y\sin\theta & k_z k_y(1-\cos\theta) + k_x\sin\theta & \cos\theta + k_z^2(1-\cos\theta) \end{pmatrix}$$

Figure 2: Calculating rotation matrix from unit vector $(k_x, k_y, k_z)$ and angle $\theta$

by the following formula:

$$d = \begin{pmatrix} \left(x - \frac{W}{2}\right) * \frac{1}{f} \\ \left(y - \frac{H}{2}\right) * \frac{1}{f} \\ 1 \end{pmatrix}$$

Then, those vectors are transformed into world coordinate system by multiplying with the rotation part of the corresponding camera-to-world matrix. Therefore, these rays can be represented as $r(t) = o + t.d$, where $o$ is the new origin (the translation part of the camera-to-world matrix) and $d$ is the calculated direction vector. For computing efficiency, we then normalize all rays to fit in a unit sphere. Finally, we sample $N$ points with values of $t$ distributing evenly in the range $[0, 1]$ for each ray and concatenate them with the direction vectors as inputs for the fully connected network.

### C. *Sinusoidal neural network*

We propose an alternative model for NeRF−−'s ReLU-based model. It is also a fully connected network with multiple dense layers, a skip layer from the input layer to the central layer to prevent vanishing gradients. However, we apply the sinusoid function $sin(\omega_0 x)$ ($\omega_0$ is a hyperparameter) as activator instead of Rectified Linear Unit activation. Its output is a 4D vector representing $(R, G, B, \sigma)$ with $R$, $G$, $B$ and $\sigma$ are respectively the color and density of the input sample point. As mentioned in the original NeRF paper [3], the density can be discovered by only the position, while we need to include the direction to determine the output color.

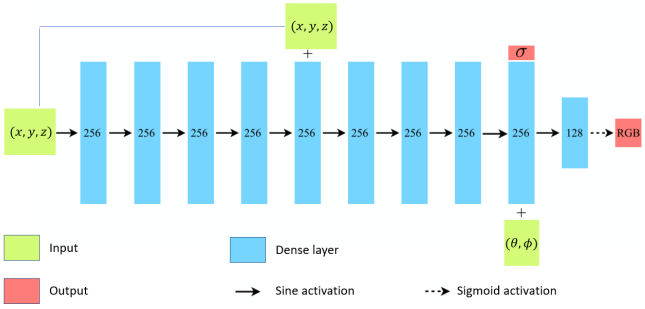


Figure 3: Network architecture

### D. *Volume rendering*

With the output color c$(R, G, B)$ and density $\sigma$ of $N$ points from each ray, we can use the classical volume rendering technique [4], which computes the integral of the density and apply them to the RGB color. Concretely, the output color $C(r)$ of ray $r(t) = o + t.d$ can be determined by:

$$C(\mathbf{r}) = \int_0^1 T(t).\sigma(\mathbf{r}(t)).\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

where $T(t) = \exp\left(-\int_0^t \sigma(\mathbf{r}(s))ds\right)$.
Finally, we optimize all parameters by minimizing the Mean Square Error between all predict pixels and respective ground-truth pixels.

## IV. Experiments

### A. *Setup*

*1) Dataset:* We conduct our experiment on five scenes from three datasets: LLFF (Fortress, Room) [18], Google's RealEstate10k (House), the Nex's shiny dataset (CD, Lab) [19] and one scene (Castle) we manually collect. Each scene contains 20-32 images, whose resolution varies from $480 \times 270$ to $640 \times 360$.

*2) Evaluation:* To evaluate the camera pose estimation, for visualization, we plot the camera poses' translation of each scene on the XY plane. Especially, on non-forward-facing scenes (RealEstate10K and ours), the camera poses are more ambiguous so we add a 3D plot in advance for each of them to make it more intuitive. We call this the refining stage and use all images for training.

We discover that after the first few hundred epochs, all camera parameters become stable and hardly change. Therefore, we freeze them for the novel view synthesis stage and reset all other parameters. We then split the input data into training and test set in proportion 9:1. We use the Peak Signal-to-Noise Ratio (PSNR), Structure Similarity Index Measure (SSIM) and Learned Perceptual Image Patch Similarity (LPIPS) [20] metrics on the test set for evaluation.

*3) Implementation details:*

*Camera parameters:* Because a parameter can be better optimized in range $[-1, 1]$, we initialize the focal $f$, the rotation matrices $r$ and translation matrices $t$ to zeros. As the focal length is usually large (about the height and weight of the taken image), we multiply $f$ by the images' weight $W$ before using it in further steps.

*Neural Radiance Fields model:* Our model follows the network architecture in 3. All hidden layers have a dimension of 128. All weights of our models are initialized in the uniform distribution $\mathcal{U}(-\sqrt{6/d}, \sqrt{6/d})$ where $d$ is the input dimension of their corresponding layers. The hyperparameter $W_0$ for the activation function is 30.0 for the first layer and 1.0 for the following layers. Although the SIREN paper recommends using the same

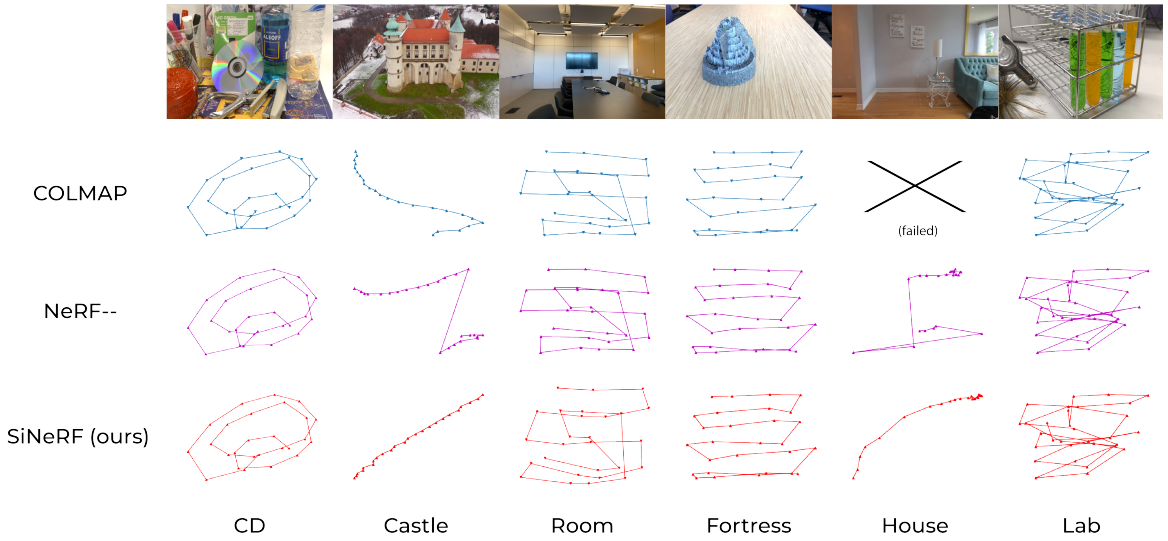


Figure 4: Camera translations on XY plane in world coordinate system (normalized)

value for the following layers, we find that using higher $W_0$ does harm the model. Thus, we decide to keep them at 1.0. For computer efficiency and preventing overfitting, we only cast 1024 random rays from each image in each step to train the models. We also use some optimizations mentioned in [3] and [16]. Instead of optimizing a single network, we train two separate networks concurrently in a single step in the synthesis stage. The first network is trained using $N_c$ points whose $t$ values are spaced evenly. After that, base on their density, we sample other $N_f$ points close to those points with high density to get more information about the scene and use them to train the second network. This technique is not applied to the refining stage because we do not have the correct pose to sample points. Through experiments, it also leads to worse results. Hence, we only use one network in this stage and get the sample points evenly in the range $[0, 1]$.

All parameters are optimized by the Adam optimizer with the initial learning rate at $5.10^{-4}$.

### B. Results

In this section, we compare SiNeRF's results with those produced by our implementation of COLMAP + NeRF [3] and NeRF−− [5]. All models are trained on a Tesla T4 NVIDIA GPU for $7 \sim 8$ hours for optimizing each scene (about 1 hour for the refining stage and $6 \sim 7$ hours for the synthesis stage depending on the number of input images).

*1) On camera poses estimation:* Since COLMAP and both models produce camera poses in different spaces, we normalize their translations in each axis to the range of $[0, 1]$ for easier comparison. In figure 4, we show that our model can reconstruct the positions of all cameras in world coordinate system similar to those produced by COLMAP as well as NeRF−−, except for cases of castle and house. Because those are non-forward-facing scenes, which are different from the others, the camera poses become more ambiguous. In the castle's case, the input images are from a video that travels farther from the top of a castle, which makes the ground truth pose a straight line. In the house's case, the rotations between each image are larger than in forward-facing scenes, which causes the similar features between them much fewer. We also experience that in such rotation-dominant cases, COLMAP fails to reconstruct the camera poses due to not enough similarity between images. In figure 5, we display the 3D plots of those scenes along with the training images for comparison and indicate that our framework can produce accurate and reasonable poses in such ambiguous scenes compared to COLMAP's and NeRF−−'s, which fall into the local minima. About the intrinsic parameter, the focal length $f$ of each scene learned from NeRF-based models and COLMAP are showed in Table 1. Both models produced smaller values than the expected ones from COLMAP, which will affect the depth of the 3D structures. Because the shorter focal length means having a wider view range, the same captured scene will closer to the camera than a camera with a longer focal length.

*2) On novel view synthesis quality:* With learned camera parameters in the refining stage, every tenth image is taken from the dataset for evaluating the synthesis stage.

| Scene | COLMAP | NeRF−− | SiNeRF (ours) |
|---|---|---|---|
| CD | 502.1 | 361.1 | 351.6 |
| Castle | 337.8 | 254.0 | 214.9 |
| Room | 370.3 | 171.1 | 202.5 |
| Fortress | 422.3 | 355.5 | 365.9 |
| House | x | 617.8 | 497.6 |
| Lab | 477.8 | 273.5 | 248.0 |

Table 1: Focal length

Table 2 shows the synthesizing quality between NeRF (with camera parameters from COLMAP), NeRF−− and SiNeRF (ours). We notice that while NeRF dominates the table, our framework takes the lead in specific cases (Castle, House), where the camera poses estimation is more accurate than the others. In forward-facing scenes, SiNeRF's results are on par with those of NeRF−− (Lab, Fortress) but worse in scenes filled with many reflective objects (CD, Room). We discover that the sinusoidal activation function can construct the 3D structure better than ReLU due to its continuously differentiable nature, but cannot represent the illumination equivalent to ReLU. Figure 6 shows that sinusoidal can learn smoother structure while ReLU is prone to checkerboard artifacts.

### C. Limitation

Besides all the advantages our method has over other neural-based methods in scene representation and novel view synthesis, it remains some noticeable downsides. Firstly, despite being able to estimate more precise camera poses, SiNeRF still falls to local minima on the focal length of cameras, which makes its quality inferior to that of the original NeRF. Secondly, our method cannot represent the illumination-dominant cases effectively as ReLU-based frameworks. Lastly, like NeRF−−, our method does not work with 360-inward facing scenes, because a simple Multi-layer Perception cannot learn correctly if two inputs have little to no similarity.

## V. Conclusion and Future Works

In this paper, we propose an end-to-end framework called SiNeRF, which can synthesize novel views from only a set of input images without pre-computed or handcrafted camera parameters. We demonstrate that using periodic activation functions like the sinusoidal function can effectively estimate the camera poses during optimizing scenes for novel view synthesis. Although we have used some optimization techniques like hierarchical sampling with two separate networks and gradient boost by using the multiplier in sinusoidal function, our framework can be further improved by integrating with PlenOctrees [15] to enable real-time rendering. We also investigate using a perceptual model like CLIP [21] to reduce the number of input images needed and help learn the visual concepts more effectively, which will brighten up the future of providing an end-to-end framework for novel view synthesis of real-world scenes.

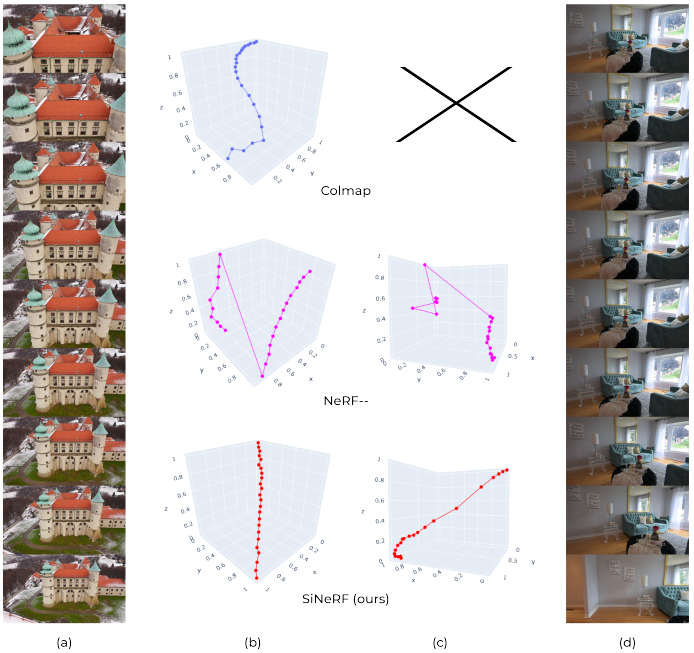


Figure 5: (a) thumbnail of Castle dataset, (b) castle dataset's camera poses in 3D, (c) thumbnail of House dataset, (d) house dataset's camera poses in 3D

| Scene | PSNR↑ | | | SSIM↑ | | | LPIPS↓ | | |
|---|---|---|---|---|---|---|---|---|---|
| | NeRF | NeRF−− | SiNeRF | NeRF | NeRF−− | SiNeRF | NeRF | NeRF−− | SiNeRF |
| CD | **24.94** | 23.10 | 21.12 | **0.87** | 0.66 | 0.60 | **0.28** | 0.34 | 0.39 |
| Castle | 20.20 | 21.75 | **24.33** | **0.78** | 0.60 | 0.70 | 0.45 | 0.44 | **0.35** |
| Room | **27.93** | 23.33 | 20.49 | **0.94** | 0.81 | 0.72 | **0.24** | 0.36 | 0.51 |
| Fortress | **29.27** | 25.08 | 26.32 | **0.88** | 0.64 | 0.66 | **0.26** | 0.37 | 0.41 |
| House | x | 22.77 | **23.57** | x | 0.78 | **0.79** | x | 0.50 | **0.44** |
| Lab | **26.07** | 23.60 | 23.31 | **0.88** | 0.71 | 0.70 | **0.26** | 0.30 | 0.28 |

Table 2: Quantitative comparison between COLMAP + NeRF, NeRF−− and SiNeRF (ours)

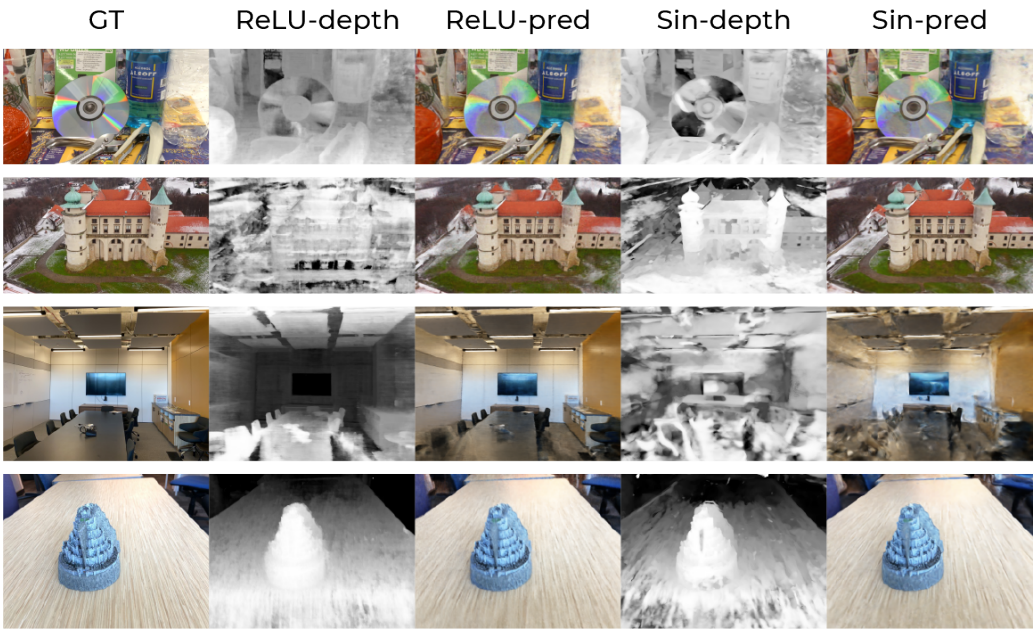


Figure 6: Novel view synthesis qualitative comparison between ReLU and sinusoidal

## REFERENCES


[1] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[2] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," in *Advances in Neural Information Processing Systems*, 2019.

[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[4] Kajiya, J.T., Herzen, and B.P.V., "Ray tracing volume densities. computer graphics," in *SIGGRAPH*, 1984.

[5] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF−−: Neural radiance fields without known camera parameters," *arXiv:2102.07064*, 2021.

[6] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. S. Overbeck, N. Snavely, and R. Tucker, "Deepview: High-quality view synthesis by learned gradient descent," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[8] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.

[9] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, 2011, pp. 2320–2327.

[10] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849.

[11] S. N. Sinha, *Multiview Stereo*. Boston, MA: Springer US, 2014, pp. 516–522.

[12] P. Henzler, V. Rassche, T. Ropinski, and T. Ritschel, "Single-image tomography: 3d volumes from 2d x-rays," *Computer Graphics Forum*, vol. 37, 10 2017.

[13] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," 2020.

[15] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "PlenOctrees for real-time rendering of neural radiance fields," in *arXiv*, 2021.

[16] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Proc. NeurIPS*, 2020.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

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

[19] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn, "Nex: Real-time view synthesis with neural basis expansion," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[20] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.

[21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.