Neural Rendering 3D

Johnny Núñez, Marcos Plaza, Qijun Jin
Department of Computer Science
University of Barcelona
mat.ub.edu

Abstract

The term Neural Volume Rendering refers to a method that generates imagery or video by tracing the path of a ray in a neural network scene, following the path of the ray, and taking an integral during the ray at some time in the future. Typically, a neural network such as this one would be used if, for example, the rays are encoded by a multilayer perceptron, which encodes quantities associated with the rays such as density and color, which in turn are combined to generate an image. Therefore, in this paper, we explain different neural networks to explain the rendering 2D to 3D.

Keywords: Neural Rendering, Geometric Deep Learning, NeRF

1 Introduction

Is it possible to render a 3D object from 2D images and save a lot of time and money with sensors and another kind of hardware? Is it possible to create objects automatically without having to create them manually in graphics engines such as Unreal Engine [1] or other softwares? And if it kept the lighting and shadows of the photograph, is it possible? Many of the problems encountered daily when making movies, real-time rendering, or video games are the cost of implementing such rendering, either with sensory suits for humans, cameras with multiview photography such as Panoptic [2]. To translate the accuracy of the real world to the virtual world requires mechanisms that are fast and inexpensive, where the creation of the new metaverse is rapidly growing thanks to realistic and complex virtual worlds.

These problems lead us to think about the existence of models that can generate hyper-realistic renderings in real time from a set of images or even a single image. To address this problem, what is called Geometric Deep Learning [3] (Johnny, Jin, Marcos) was defined, not only to generate 3D renderings but also to learn about multidimensional data and extract relevant information.

Geometric Deep Learning in Computer Vision is a new field of machine learning that can learn from complex data such as graphs, images, and multidimensional points. It is an attempt at geometric unification of a broad class of machine learning problems from the perspectives of symmetry and invariance. It seeks to apply traditional neural networks to 3D objects, graphs, and varieties. In our case, it is a specific problem, a neural network is applied to multi 2D images to create 3D objects.

The algorithms that appear in this article cannot be explained without first learning about some of the real-time rendering techniques of recent years. These technologies make it possible to generate or calculate images (in two dimensions) from a 2D or 3D model. One of the best known is raytracing [4] (Marcos), based on emulating the physical behaviour of light on landscape elements, by using the concept of rays. Roughly speaking, these rays will be projected into the scene and will be bounced

Submitted to NeurIPS 2021 AI for Science Workshop.

back and forth a number of times to finally decide what colour to paint the pixel. Another technique used in computer graphics is known as Volume Ray Marching or Volume Ray Casting. This method processes volume data and must not be mistaken with ray casting in the sense used in ray tracing, which processes surface data. In the volumetric variant, the computation does not stop at the surface but "pushes through" the object, sampling the object along the ray. Therefore, here is an image to compare both technologies.

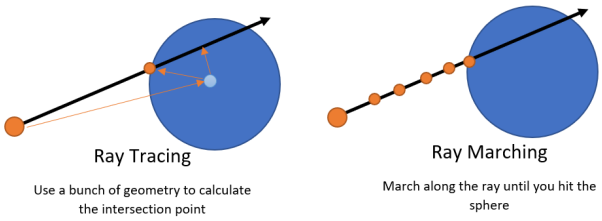

Figure 1: Ray Tracing versus Ray Marching

Given a 3D model, the task of this Volume Ray Casting algorithm [5] (Marcos) is to approximate for each ray which colour each voxel (each cell in three-dimensional space) should be painted in. How is this done? Without going too deep, we will do it through the so-called transfer function ($f(x)$). As we have said, this function takes the numerical data from the dataset to decide what colour and opacity pixels should be painted. Thus, by manipulating this function $f(x)$ we are changing the display of the objects in the scene. Therefore, with this function we can compute the pixel's color along all rays. In essence, this function for mapping the colour of every coordinate in the image, called transfer function $f(x)$ will be the one used by the algorithms explained in this paper. Therefore, that given a new angle or a new observer position, we will be generating a new and accurate visualization simply by predicting every value for every pixel.

2 State of the Art

The focus of this section is to introduce Neural Rendering and its development, as well as the NeRF algorithms and how they are derived.

2.1 Background

First of all, these occupancy networks [6] (Jin) are based on implicit, coordinate-based learning of occupancy. Based on a feature vector and a 3D point, a network of five ResNet blocks predicts binary occupancy. Similarly exists a 6 layer MLP decoder that is used in IM-NET [7] (Marcos) to predict binary occupancy given a feature vector and a 3D coordinate. This algorithm can be used for auto-encoding, shape generation (GAN-style), and single-view reconstruction. DeepSDF [8] (Johnny) is another method for regressing a signed distance function directly from a 3D coordinate, as well as a latent code. There is an 8-layer MPL with skip connections at layer 4, setting a trend. Finally, PIFu [9] (Marcos) demonstrates that it is possible to learn highly detailed implicit models by reprojecting 3D points into pixel-aligned feature representations. This idea will later be reprised to great effect in PixelNeRF.

On the other hand, several other approaches are built on top of the implicit function concept. First, Structured Implicit Functions [10] (Johnny) show that you can combine these implicit representations, e.g., simply by summing them. CVxNet [11] (Jin) takes a pointwise max (in 3D) to combine signed distance functions. Additionally, the paper contains several other elegant methods for reconstructing an object from depth or RGB images. Similarly exists BSP-Net [12] (Johnny) which is in many ways like CvxNet, but it relies on binary space partitioning at its core to generate polygonal meshes natively, rather than by using expensive meshing methods. To represent larger, extended scenes, Deep Local Shapes [13] (Jin) store a DeepSDF latent code in a voxel grid. Scene Representation

Networks [14] (Marcos) or SRN are quite similar to DeepSDF in terms of architecture, but include a differentiable ray marching algorithm for determining the closest point of intersection of an implicit surface and an MLP which regresses color, enabling it to be learned from multiple posed images. Differentiable Volumetric Rendering [15] (Marcos) shows that implicit scene representations can be paired with a differentiable renderer, making the system trainable from images, similar to SRN. Despite the use of the term volumetric renderer, the core contribution is a clever trick to make depth of the implicit surface differentiable: no volume integration is required. It has a more sophisticated surface light field representation and can also refine camera pose learning. Implicit Differentiable Renderer [16] (Marcos) presents a similar technique. The neural articulated shape approximation [17] (Jin) comprises implicit functions to represent articulated objects such as human bodies.

2.2 NeRF Models

In the NeRF [18] (Johnny, Jin, Marcos), they use the DeepSDF architecture, but the regression is not based on a signed distance function, but density and color. In the next step, a numerical integration method (which is easily differentiable) is used to approximate the true volumetric rendering step. NERF++[19] (Jin) proposes a separate NeRF model for handling unbounded scenes in the background. The Nerfies[20] (Johnny) model and its underlying D-NeRF model use a second MLP to apply a deformation to each frame of the video. D-NeRF [21] (Johnny), based on the same acronym, is quite similar to Nerfies. However, it is limited only to translations. Neutral Reflectance Fields [22] (Marcos) extend NeRF by adding a local reflection model in addition to density. The relighting results are impressive, albeit with a single point source. As a conditional variant of NeRF, GRAF [23] (Johnny), "Generative Model for Radiance Fields" is able to add both appearance and shape latent codes, while viewpoint invariance is achieved through the use of GAN-style training. Neural Scene Graphs [24] (Jin) integrate several object-centric NeRF models into a scene graph. A NeRF MLP is used in a pose estimation framework in NeRF [25] (Marcos). The fine-tuning of the poses can even improve view synthesis on standard datasets. However, illumination has not yet been addressed.

3 Methodologies

The scene reconstruction based on neural radiance field consists of 3 phases: Firstly, we need to define a set of 3D points for the camera in the scene. Then, using these 3D points with their corresponding viewing cones to train the neural network to generate the RGB color and its volume density. Finally, the 2D image is reconstructed by using a volume rendering to accumulate both RGB color and density.

By combining those output images, it can give us an impression of a real 3D scene which can be seen from different views.

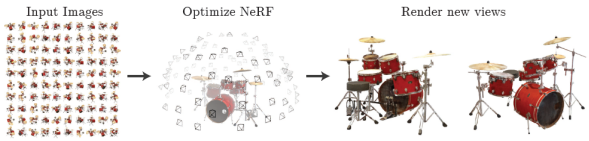

Figure 2: Reconstruction phases with NeRF

3.1 Pipeline

To know how the neural radiance field works, we have to explain the primitive data that involves in the neural network and then look in detail how is the training process.

For the input, we need two elements apart from the image itself: the 3D coordinates (x, y, z) at which it will perform the volume ray marching, and the direction (θ, ϕ) of the viewpoint at which we are looking for. These 5 primitive data are composed to be the continuous 5D scene representation, and

by optimizing these parameters in the neural network process, one can minimize the pixel-level color error between the generated set of images and the original ones.

The output of this neural network is the radiance emitted from every 3D point and viewpoint in the 3D space with a volume density. The radiance emitted is the RGB color composition of the image, which is view-dependent, as the color representation viewed changes from a different viewpoint. The volume density, which is defined for every 3D point, is used to control the radiance saturation on the volume ray marching.

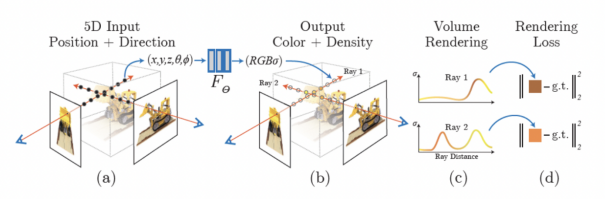

Figure 3: NeRF architecture

Based on the idea that our process is differentiable, the gradient descent method is applied to optimize the model by iteratively minimizing the error between the original image and the rendered image. Since this error is minimized from several viewpoints, the neural network is capable to train a model on a the set of original images, giving both accurate color and density to render.

3.2 Volume rendering

Volume rendering is a set of techniques used to display 2D projections of 3D discretely sampled data sets (usually 3D scalar fields). As seen on the introduction section, particularly on NeRF models, we are using the function known as transfer function $f(x)$, applied in the Volume Ray Marching algorithm. As previously mentioned, it takes a vector with the coordinates, density and angle to finally put a color and opacity in each pixel. The parameters of this function will be fitted by the Neural Network on it's training phase, so that predict the unknown values from a different point of view.

Here, the volume density is defined as the differential probability of a ray terminating at an infinitesimal particle at a location x. Then the color expected $C(r)$ of camera ray is determined by $r(t) = o + td$ in between the near bound t_n and far bound t_f.

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

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$

The function $T(t)$ is the accumulated transmittance along the ray from t_n to t, which is computed as the probability that the ray travels from t_n to t without hitting any other object. Finally, to render a view given a virtual camera is to estimate the integral $C(r)$ of the continuous neural radiance field.

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp\left(-\sigma_i \delta_i\right)\right) \mathbf{c}_i$$

where $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$ and $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples. Therefore, the alpha values is $\alpha_i = 1 - \exp\left(-\sigma_i \delta_i\right)$ for the alpha composition.

3.3 Enhancement

Actually, the basic implementation cannot generate high resolution render for a complex scene because of the low dimensional feature space. However, we can recover the idea of waves in the physics world, allowing us to enlarge the feature space by representing them in a frequency function. This data transformation is called **positional encoding**. Besides, another improvement technique based on the idea of increasing computational efficiency is the **hierarchical procedure**, which allow us to sample the N query points of the ray on demand.

3.3.1 Positional encoding

The positional encoding has emerged due to the fact that deep networks are biased towards learning low frequency functions. For a complex scene, it requires the network to compute a high-frequency variation in color and geometry, therefore the input data should be mapped into a higher dimensional space to preserve this high-frequency variation. The linear mapping γ changes our input dimension from $\mathbb{R}$ to $\mathbb{R}^{2L}$:

$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right)$$

Being L parameter to determine the frequency. This function is applied to both 3D points and the viewing cone, which are normalized in the range [-1, 1].

3.3.2 Hierarchical sampling

The algorithm samples each camera ray as a query of N equal distance points and the evaluation of neural radiance field network on these points is inefficient due to the radiance field presents free space and occlusion which does not provide any information on the rendered image. Therefore, the process to train the network must be selectively. Regarding to this, we need to optimize two networks at different levels of fineness: the coarse level and the fine level.

Firstly, we sample a set of N_c locations to determine the coarse network. Given the output of the coarse network, we apply a weighted sum to all sampled colors c_i along the ray. Then by normalizing the weights as:

$$\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$$

it produces a piecewise constant probability density function along the ray.

The fine level is applied over the coarse level, yet again it has to sample a set of N_f locations to a certain subsets of coarse level and then the color rendered of the ray is computed taking into account the union of both samples $N_c + N_f$. On this fine level, the model is able to allocate more samples to regions that contain visible content.

Considering these enhancements over the basic implementation, at each optimization iteration, the batch of camera rays which is randomly sampled will be processed with the hierarchical sampling to query 2 types of samples: N_c samples from the coarse network and $N_c + N_f$ samples from the fine network. Then the volume rendering will proceed to render a color of each ray from both sets of samples. Indeed, the loss function has to be adapted taking into account both set of renders.

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

where $\mathcal{R}$ is the set of rays in each batch, $C(r)$ is the true pixel color, $\hat{C}_c(r)$ is the coarse level predicted pixel color and $\hat{C}_f(r)$ is the fine level predicted pixel color for ray r.

3.4 NeRF in the Wild

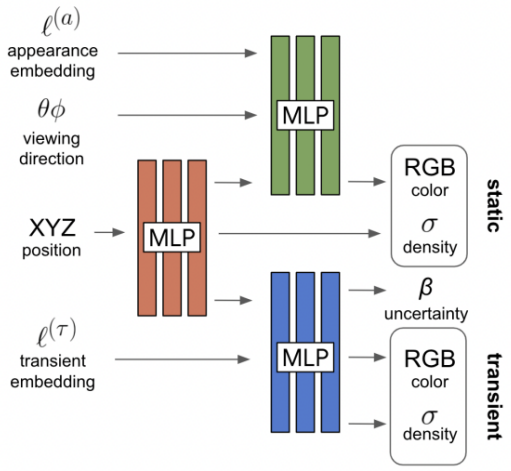

Figure 4: Multimodal embedding in NeRF-W, Image from Original Paper

As part of a low-dimensional latent embedding space, NeRF-W [26] (Johnny, Jin, Marcos) captures the lighting and photometric postprocessing. It is possible to smoothen out the variation in appearance by interpolating between two embeddings without affecting the 3D geometry of the object. A major difference with NeRF is the oriented and photometric postprocessing in the form of a low-dimensional latent embedding space. There is no lighting nor shading information in the vertex data of a mesh, but it smooths out the variations as a result. A smooth interpolation between two embeddings allows you to capture the variation in appearance without altering the geometry of the 3D object. An orientation and photometric postprocessing method in a low-dimensional embedding space using a latent embedding function. We perform the inverse lighting and photometry in the latent space, without having to touch the vertex data. Whenever a photographer takes a series of images, we can be used to compute the following information: orientation of the camera (rotation, 3-D translation), position of the light sources around the scene (3-D translation), photometric correction of the images (relative intensity of each pixel in each image) is particularly useful when capturing photos of static scenes or when giving a consistent rendering of the scene.

4 Results and Discussion

In this section, we present the metrics that use Nerf and his derivations. - Evaluation Metric

- PSNR(Peak Signal-to-Noise Ratio): higher PSNR, lower MSE. Lower MSE implies less difference between the ground truth image and the rendered image. Thus, the higher PSNR, the better model.
- SSIM(Structural Similarity Index): Checks the structural similarity with the ground truth image model. The higher SSIM, the better model.
- LPIPS(Learned Perceptual Image Patch Similarity): Determines the similarity with the view of perception; using VGGNet. The lower LPIPS, the better model.

The results obtained are promising. The Neural Radiance Field model weighs only 5Mb. Neural Radiance Field is capable of producing high-quality images and videos, even in real time. The model is easy to train and does not suffer from overfitting. It can be used for the generation of realistic images and videos for a variety of applications.

Figure 5: Result NeRF

The next step will be to try to improve the model even further. The current model is not capable of generating all types of images. For example, it is not capable of generating images of faces or of realistic textures. The most important direction for further research is to make the model more light-weight and to increase its real-time capabilities that already exist in models such as NeRF++.

Figure 6: Result NeRF 3D

On the other hand, NeRF-W can reconstruct monuments in 3D. This would be very useful to make hyper-realistic maps of buildings with images from the internet, no matter the people or objects that are moving at that moment if we have a large enough dataset.

Figure 7: Result NeRF-W

5 Conclusion

The Neural Radiance Fields seems to us just the beginning of the new reconstruction technique to come out in the market. The benefit of this methods is that the scene reconstructed occupies even less than the single image processed to the input. So there is a big improvement in memory usage in order to compute a new render in a scene.

However, this technique takes within 1 or 2 day to learn all the weights required, which is a lot of time to speed for giving the scene, even though by using more GPU will decrease the computational time. Another counterpart of Neural Radiance Fields is that the object is reconstructed by a set of weights beyond a 3D space (scene) taken a set of the images and the corresponding viewpoints, meaning that the creation does not preserves the 3D coordinates of the object in comparison with this traditional technique of rendering.

To conclude, will NeRF technology be applied in the future on real-time graphics? Clearly we have no answer to this question. From the most objective point of view possible, the computer graphics industry is a very mature and established sector that has seen many advances in both hardware and

software. By means of polygons and meshes, as well as raytracing algorithms among many other technologies, and the actual results obtained is quite awesome. But it is conceivable that in the not too distant future, mixed computer graphics generation techniques will be applied in which machine learning, in this case Neural Radiance Fields, will play an important role.

References

[1] Epic Games. Unreal engine, 2022.

[2] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[3] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velivckovi'c. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *ArXiv*, abs/2104.13478, 2021.

[4] Andrew S Glassner. *An introduction to ray tracing*. Morgan Kaufmann, 1989.

[5] Kun Zhou, Zhong Ren, Stephen Lin, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Real-time smoke rendering using compensated ray marching. In *ACM SIGGRAPH 2008 papers*, pages 1–12. 2008.

[6] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4455–4465, 2019.

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5932–5941, 2019.

[8] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019.

[9] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2304–2314, 2019.

[10] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas A. Funkhouser. Learning shape templates with structured implicit functions. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7153–7163, 2019.

[11] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey E. Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 31–41, 2020.

[12] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 42–51, 2020.

[13] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, S. Lovegrove, and Richard A. Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *ECCV*, 2020.

[14] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *ArXiv*, abs/1906.01618, 2019.

[15] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3512, 2020.

[16] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *arXiv: Computer Vision and Pattern Recognition*, 2020.

[17] Timothy Jeruzalski, Boyang Deng, Mohammad Norouzi, J. P. Lewis, Geo rey E. Hinton, and Andrea Tagliasacchi. Nasa: Neural articulated shape approximation. *ArXiv*, abs/1912.03207, 2020.

[18] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[19] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *ArXiv*, abs/2010.07492, 2020.

[20] Keunhong Park, U. Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. 2020.

[21] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10313–10322, 2021.

[22] Sai Bi, Zexiang Xu, Pratul P. Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Milovs Havsan, Yannick Hold-Geoffroy, David J. Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *ArXiv*, abs/2008.03824, 2020.

[23] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *ArXiv*, abs/2007.02442, 2020.

[24] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2855–2864, 2021.

[25] Yen-Chen Lin, Peter R. Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330, 2021.

[26] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.