NeRF and follow-up works

César Díaz Blanco

February 2024

1 Introduction

Novel view synthesis is the task of synthesizing a target image with an arbitrary target camera pose from given source images and their camera poses.

Recent progress has been successful with the aid of machine learning with neural radiance fields (NeRFs) [3] at the forefront.

This method is of importance to the field of virtual humans as it more accurately represents the identity of human shapes by capturing color and texture in a view-dependent appearance manner.

In this report we provide a brief overview of volume rendering heavily inspired by [7], followed by an explanation of the neural radiance field paper, its limitations, and follow-up works to fix these. Finally, we briefly talk about methods tailored to human image synthesis.

2 Volume Rendering

The process of synthesizing an image using a computer program is called rendering. Volume rendering has its origin in radiative transfer physics [1]. The core idea as in surface rendering, is to cast a camera ray to the scene and keep track of the ray hits which then determine the pixel value.

In the physics interpretation there are three concepts which explain how light interacts with objects: absorption, scattering, and emission. Scattering accounts for light bouncing off instead of taking a straight path and is dropped when doing volume rendering. Absorption happens when a ray is terminated while transmittance, $T(t)$, represents the probability of not being absorbed at point t along the ray. Emission happens when a camera ray hits a surface which is the same as hitting a high density $\sigma(t)$ at point t along the ray.

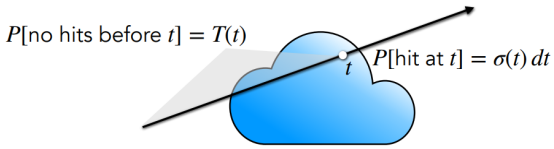


Figure 1: Transmittance and density intuition.

2.1 Derivation of the volume rendering equation

Following the probabilistic notion from figure 2 the transmittance is derived from the density as follows:

$$\begin{aligned} P[\text{no hits before } t+dt] &= P[\text{no hit before } t] \times P[\text{no hit at } t] \\ T(t+dt) &= T(t)(1-\sigma(t)dt) \end{aligned}$$

$T(t+dt)$ is approximated with a first degree Taylor expansion around t:

$$\begin{aligned} T(t) + T'(t)dt &= T(t) - T(t)\sigma(t)dt \\ \frac{T'(t)}{T(t)}dt &= -\sigma(t)dt \\ \log T(t) &= -\int_{t_0}^{t} \sigma(s)ds \\ T(t) &= \exp\left(-\int_{t_0}^{t} \sigma(s)ds\right) \end{aligned} \tag{1}$$

Where $T(t_0) = 1$ as the ray could not possibly hit anything before exiting the camera. The probability that a ray terminates at t is:

$$\begin{aligned} P[\text{first hit at } t] &= P[\text{no hit before } t] \times P[\text{hit at } t] \\ &= T(t)\sigma(t)dt \end{aligned}$$

And derive the volume rendering equation which returns the expected color returned by the ray, which is given by the color, $\mathbf{c}$, at the first hit:

$$\int_{t_0}^{t_f} T(t)\sigma(t)\mathbf{c}(t)dt \tag{2}$$

2.2 Approximating the volume rendering equation

In practice, integrals must be approximated as summations over discrete small segments. The volume rendering equation (2) is approximated by splitting the ray into n segments with endpoints $\{t_1, t_2, ..., t_{n+1}\}$ with lengths $\delta_i = t_{i+1} - t_i$. Density and color are constant in these segments but not transmittance since the density will have a narrow peak inside a segment when a surface is hit.

$$\int_{t_0}^{t_f} T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i dt \tag{3}$$

The transmittance integral (1) for segment i, $t \in [t_i, t_{i+1}]$, is split in: the light blocked by previous segments, and the light blocked by a part of the segment.

$$\begin{aligned} T(t) &= \exp\left(-\int_{t_0}^{t_i} \sigma_i ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i ds\right) \\ &= \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right) \exp(-\sigma_i(t-t_i)) \end{aligned} \tag{4}$$

The segment opacity $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$, rooted in the traditional alpha compositing method, indicates how much light is contributed by ray segment i. Intuitively, if a ray segment highly contributes, then the transmittance after such segment must be low. This is shown when expressing the transmittance up to segment i, T_i, from the corresponding αs:

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{5}$$

The new expressions for transmittance (4) is used to integrate (3):

$$\begin{aligned}
\int_{t_0}^{t_f} T(t)\sigma(t)\mathbf{c}(t)dt &\approx \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp(-\sigma_i(t - t_i))dt \\
&\approx \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \frac{\exp(-\sigma_i(t_{i+1} - t_i)) - 1}{-\sigma_i} \\
&\approx \sum_{i=1}^{n} T_i \mathbf{c}_i (1 - \exp(-\sigma_i \delta_i))) = \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i
\end{aligned} \tag{6}$$

3 NeRF

The main goal of neural radiance fields is to achieve novel view synthesis. It does so by providing the the RGB and density values needed in (6) to get T and α. In the next sections we talk about how this is done.

3.1 MLP

The input to the MLP is a 5D coordinate with the first three coordinates corresponding to a point $\mathbf{x} = (x, y, z)$ along the ray segment i and the last two to the viewing direction $\mathbf{d} = (\theta, \phi)$. Its output is the corresponding volume density and directional emitted color:

$$F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma) \tag{7}$$

The MLP is optimized by following (6) to render a pixel for a view and computing the loss as the total squared error from the true pixel color. The ray is retrieved from the camera pose estimated by COLMAP. This process is described in figure 2

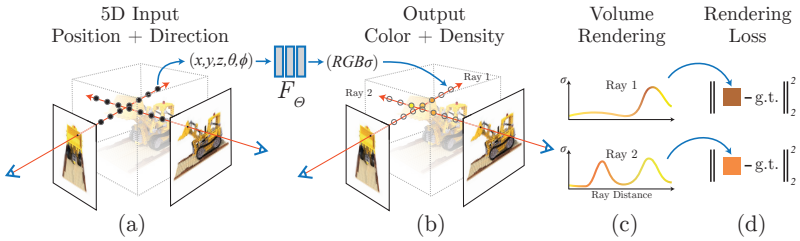
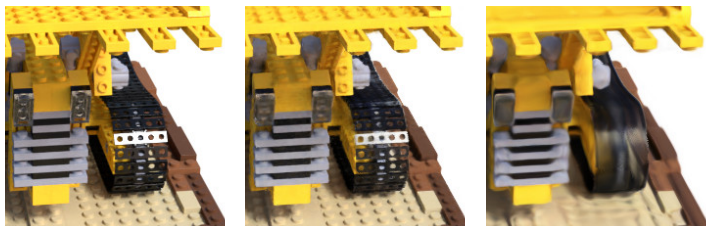


Figure 2: Transmittance and density intuition.

3.2 Positional Encoding

The basic implementation of optimizing a neural radiance field representation for a complex scene does not converge to a sufficiently high resolution representation and is inefficient in the required number of samples per camera ray. These issues are addressed by transforming input 5D coordinates with a positional encoding implemented as a mapping γ which outputs a higher dimensional space $\mathbb{R}^{2L}$:

$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \quad \cos\left(2^0\pi p\right), \quad \cdots, \quad \sin\left(2^{L-1}\pi p\right), \quad \cos\left(2^{L-1}\pi p\right) \right). \tag{8}$$



Ground Truth | Complete Model | No Positional Encoding

Figure 3: Removing the positional encoding drastically decreases the model's ability to represent high frequency geometry and texture, resulting in an oversmoothed appearance.

3.3 Geometry

Even though NeRF only optimizes for rendering quality by comparing corresponding pixels in 2D, it is able to learn the geometry of the scene. This can be easily done by altering (6) so that we calculate the expected depth:

$$\bar{t} = \sum_{i=1}^{n} T_i \alpha_i t_i \tag{9}$$

Note that this is the same as calculating any statistic and thus it is possible to "volume render" any quantity into a 2D image. Figure 4 shows scene editing of higher-level semantic feature maps by using a CLIP similarity score. [8]

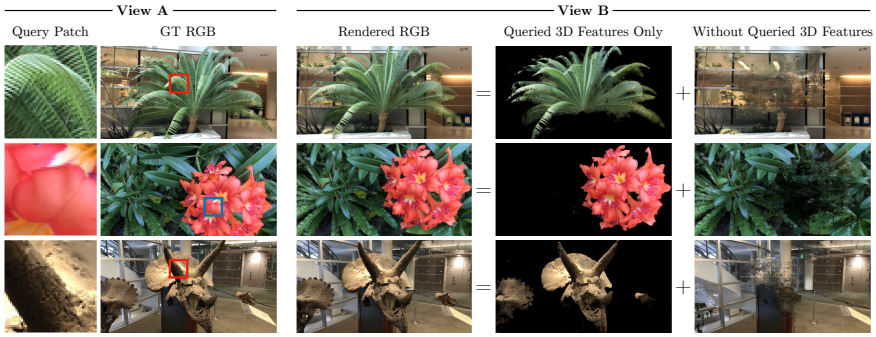


Figure 4: **Scene editing.** NeRF renders an image from an unseen view and separates foreground from background by matching the 3D features to the query patch features.

4 NeRF's limitations and solutions

NeRF's use of an MLP is both its advantage and pitfall. It enables optimization as it is a continuous representation but also limits control of the output geometry, lacks generalization as each NeRF encodes a single scene, and is more expensive to train compared to other traditional methods like photogrammetry. In the next sections we talk about these limitations and the follow-up works that address these.

4.1 Scene specific and static

Given that NeRF takes in a 5D input, the first solution that comes to mind is having a 6D input with time added. This naïve solution does not work as the MLP is designed to learn a single representation and this solutoin would be the same as training as many MLPs as frames we want to capture without exploiting the similarity between scenes.

D-NeRF. [6] proposes to learn a canonical shape and its radiance in order to disentangle the time dependent information from the MLP. The method first transforms the point position to its canonical configuration as $\Psi_t : (\mathbf{x}, t) \rightarrow \Delta\mathbf{x}$. $t = 0$ is chosen as the canonical scene $\Psi_t : (\mathbf{x}, 0) \rightarrow \mathbf{0}$. Then, the assigned emitted color and volume density under viewing direction $\mathbf{d}$ is equal to those in the canonical configuration $\Psi_x : (\mathbf{x} + \Delta\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$. Here Ψ_x is the same as F_Θ in the original NeRF paper. Figure 5 shows the architecture of D-NeRF.

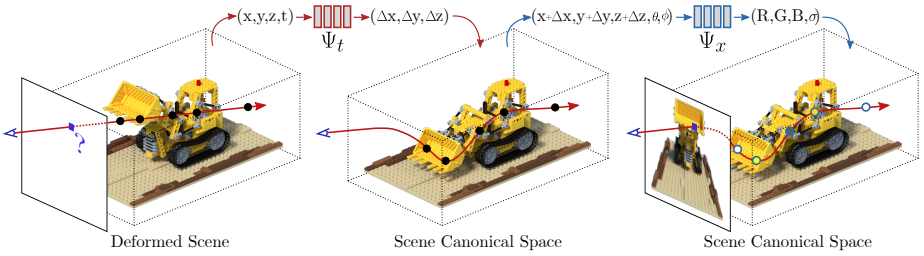


Figure 5: **D-NeRF's Architecture**.

4.2 No editing and no generalization

Since the scene is memorized in the network F_Θ weights, editing is not as straightforward as changing a vertex position in mesh representations. As well, it is not possible to generalize to many scenes with a single MLP.

Control-NeRF. [2] proposes to learn feature volume V_s from different scenes, whose output will be the input to the familiar F_Θ MLP:

$$F_\Theta : (\mathrm{S}(V_s, \mathbf{p})), \gamma(\mathbf{d})) \rightarrow (\mathbf{c}, \sigma) \tag{10}$$

where the feature vector $\mathbf{v}_s$ is obtained by sampling of the feature volume $\mathbf{v}_s = \mathrm{S}(V_s, \mathbf{p})$, where S indicates the trilinear resampling operation. This formulation allows us to optimize the volume V_s for each scene, while simultaneously learning the parameters of F_Θ. After this initial training stage, the parameters Θ of this rendering module are fixed. For every novel scene, we only optimize its feature volume V_s. This will allow us to combine and edit scenes by manipulating their respective feature volumes V_s, and render the result using (6). Figure 10 shows an example of Control-NeRF in action.

Figure 6: Control-NeRF adding objects in a scene.

4.3 Expensive training

The vanilla NeRF was a proof of concept showcasing the advantages of using an MPL as the function returning the density and color values since both the 5D input domain and output are continuous and differentiable. However, it used a fixed positional encoding. Further work explores learning this positional encoding which enables the use of smaller networks which reduces the number of floating point and memory access operations speeding up the training step.

Instant Neural Graphics Primitives. [4] proposes an augmentation by a multiresolution hash table of trainable feature vectors whose values are optimized through stochastic gradient descent. It helps thinking about in reverse starting from the desired input to the MLP. Since we want to reduce F_Θ's size we want to provide it with an already rich representation of $\mathbf{x}$. First, we exploit

the continuous nature of $\mathbf{x}$ by doing linear interpolation of the rich features we are learning. These linear features should be collected in a faster and more effective way than MLPs, such as hash encoding. Lastly, these encoded features should follow the input structure, in our case for 3D coordinates, sampling the corners of 3D voxels. Figure 7 explains this further.

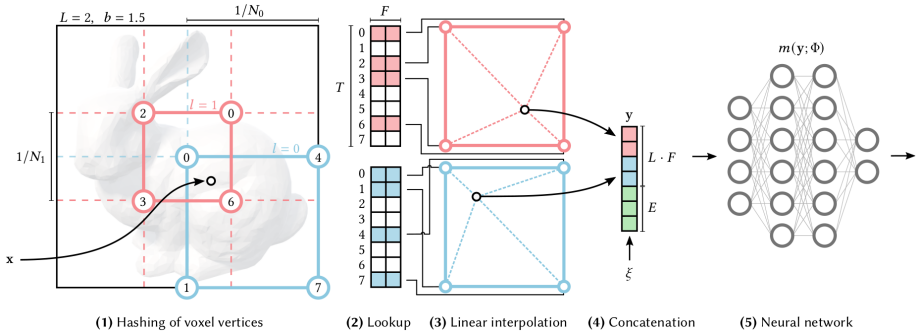


Figure 7: Illustration of the multiresolution hash encoding in 2D. **(1)** for a given input coordinate $\mathbf{x}$, we find the surrounding voxels at L resolution levels and assign indices to their corners by hashing their integer coordinates. **(2)** for all resulting corner indices, we look up the corresponding F-dimensional feature vectors from the hash tables and **(3)** linearly interpolate them according to the relative position of $\mathbf{x}$ within the respective l-th voxel. **(4)** we concatenate the result of each level, as well as auxiliary inputs $\xi \in^{E}$ such as the viewing direction, producing the encoded MLP input $y \in^{LF+E}$, which **(5)** is evaluated last. To train the encoding, loss gradients are backpropagated through the MLP **(5)**, the concatenation **(4)**, the linear interpolation **(3)**, and then accumulated in the looked-up feature vectors.

4.4 Surface extraction

Even though NeRF is able to recover geometry as in (9), this result is far from perfect as a threshold value is needed to distinguish forefront objects from background. Recent methods take the best of surface and volume rendering but, appropriately, constrain their input to scenes without fog or smoke.

UNISURF. [5] assumes only hard surfaces, thus constraining (6) which no longer needs α and uses occupancy $o \in \{0, 1\}$ so that the volume rendering equation is as follows:

$$\sum_{i=1}^{N} o(\mathbf{x}_i) \prod_{j<i} \big(1 - o(\mathbf{x}_j)\big)\, c(\mathbf{x}_i, \mathbf{d}) \tag{11}$$

Then, instead of learning the usual F_Θ, the occupancy field o_Θ and the color field c_Θ are learned, with the former depending only on $\mathbf{x}$ and the latter conditioned on the surface normal $\mathbf{n}$ and feature vector $\mathbf{h}$ of the geometry network as well as the first hit at $\mathbf{x}_s$ which is retrieved with root-finding along the ray.

5 Conclusion

Novel view synthesis is a fast and daring research topic. Neural fields have brought new detail and reconstruction accuracy. However, memory and computation complexity are an issue. Furthermore, as in any method which uses neural networks, interpretability and control is not as straightforward as the information is entangled in the network's weights.

Fortunately, a lot of interest is in the topic and a myriad of works continue to improve on these pitfalls and try to extend this convenient continuous representation. Some of these workarounds and expansions to the original NeRF paper were shown during the latter section of this report.

References

[1] S. Chandrasekhar. *Radiative Transfer*. Dover Books on Physics. Dover Publications, 2013. 1

[2] Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. *arXiv preprint arXiv:2204.10850*, 2022. 6

[3] 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. 1

[4] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 6

[5] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021. 7

[6] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 5

[7] Matt Tancika, Ben Mildenhall, Pratul Srinivasana, Jon Barron, and Angjoo Kanazawa. Eccv 2022 tutorial: Neural volumetric rendering for computer vision. 1

[8] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representations. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2022. 4