CSE291 NeRF Questions

Sambaran Ghosal

December 2022

1 What is a radiance field? What information is included in a radiance field? How is neural radiance field (NeRF) different?

1.1 Answer

A radiance field is defined as a mapping $\mathcal{L} : R^3 \times S^2 \rightarrow R^3$, taking a point (x, y, z) and its corresponding viewing direction in the camera frame (θ, ϕ) to the color space defined by the RGB values at that position and direction. Basically radiance field contains the information of the RGB values at different points in the image space.

A neural radial field is a neural network approximation that takes in as input the point and directions of the image space and outputs the rgb value and color density function at each point.

2 What is ray marching? Given a radiance field, how is each pixel calculated? (This is called the render equation.) Write down your render equation in a concrete math expression with clarified notations.

2.1 Answer

Ray Marching is defined as the process of obtaining the rendered image from the neural radiance field output i.e. rendering the scene based on the rgb values of each position along the ray and the density function along the ray. The volume rendering assigns color to a particular position based on the probability that the ray can travel upto this point without hitting any other particle. We can sample points along the ray using stratified sampling as follows

$$t_i \sim \mathcal{U}[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)] \tag{1}$$

where t_n, t_f are the near and far bound planes of the ray, N is the number of samples we need and $i \in [1, 2, 3, ..., N]$. Let $c_i = (r, g, b)_i$ be the output color from the MLP for i^{th} point along a ray, σ_i be the density function along at this point. We then estimate the color of a position using the quadrature integral approximation as

$$\hat{C}(r) = \sum_{i=1}^{N} e^{-\sum_{j=1}^{i-1} \sigma_j \delta_j} (1 - e^{-\sigma_i \delta_i})) c_i \tag{2}$$

where $\delta_i = t_{i+1} - t_i$ is distance between adjacent samples along the ray and t_i are generated by the stratified sampling algorithm along each ray. Hence we can repeat this process for all points along a ray and get the corresponding color value at that point. Doing this for all rays and all positions will hence generate our image / novel scene.

3 What is positional encoding? What is the purpose of positional encoding in NeRF models? Train your model without positional encoding and compare the results. You need to show at least two pairs of examples.

3.1 Answer

Neural networks despite being called universal function approximators, are not so good at approximating high frequency / varying functions. Deep Learning methods tend to learn low frequency function approximators to the input data. Findings by Rahamanet al. [**?**] shows that mapping the inputs to first a higher dimensional space using frequency functions like sin, cos before feeding them to the neural networks helps better fit the data with high frequency variation.

If the MLP is represented as $_\theta$ as a mapping from $xyz\theta\phi$ to color space as $F_\theta : xyz\theta\phi \rightarrow c$, then the positional encoding can be defined as first passing the input $xyz\theta\phi$ to a frequency encoder $\gamma(p)$ which in our case is a harmonic embedding defined by

$$\gamma(p) = (sin(2^0 p), cos(2^0 p), sin(2^1 p), cos(2^1 p), ..., sin(2^{L-1} p), cos(2^{L-1} p) \tag{3}$$

where L is the number of frequency embeddings desired. The MLP is now defined as $F_\theta := F_\theta \; o \; \gamma(p)$ where the positional encoding is applied to each component of position and viewing direction vector.

The performance of two networks, one with positional encoder and one without positional encoder is shown below in Figure(**??**)

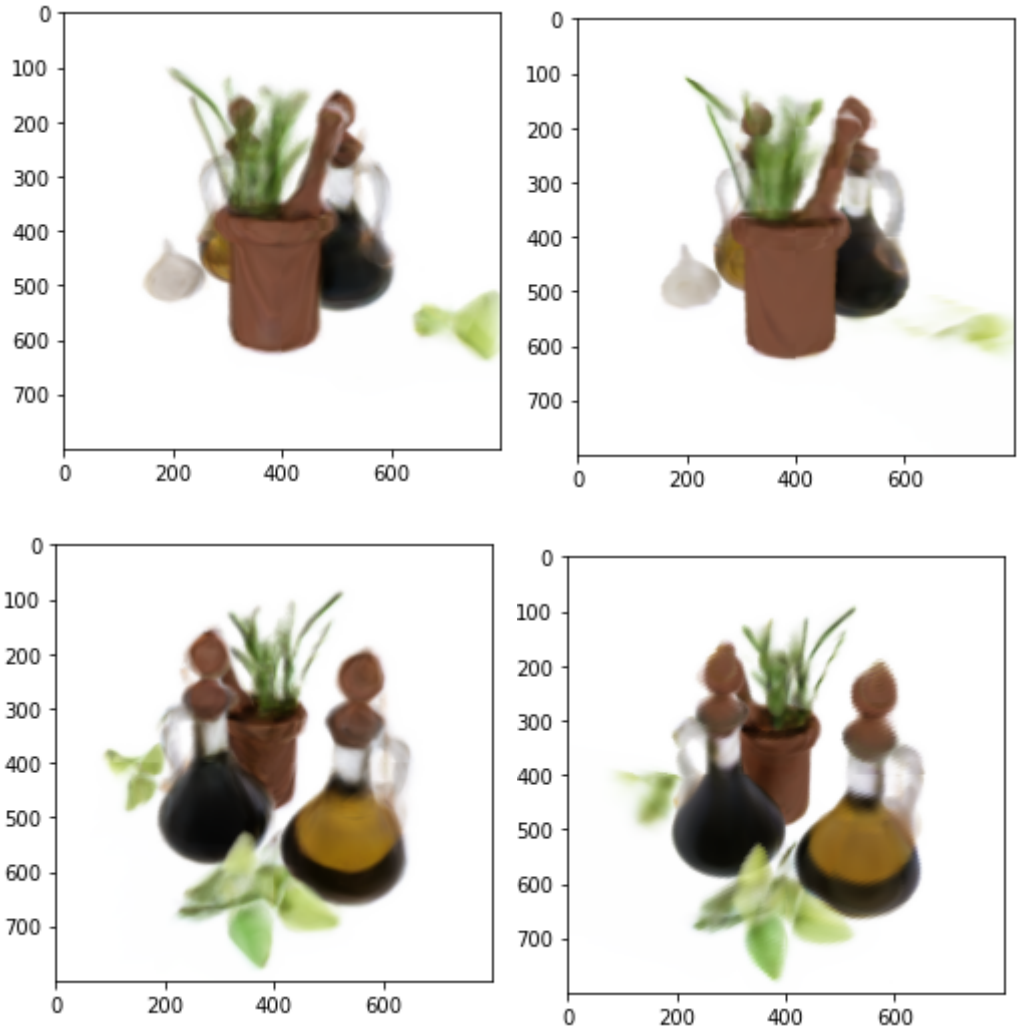

Figure 1: Generate image with no positional encoder (left) and with encoding (right)

4 Is it possible to extract scene geometry (e.g. depth) information from a trained NeRF model? Describe your method in detail. Implement your method and show two depth maps generated by your method.

4.1 answer

Yes, it is possible to produce depth map from a trained NeRF model. From NeRF, we have the sampled points along rays. Let $z_i = t_i - t_{i-1}$ be the distance between adjacent sampled points. From the NeRF MLP, we have the density function at each sampled point. Intuitively, density at each point somehow related to what is the probability that the ray travels upto a given point without being obstructed by other points. Mathematically, the term $e^{-\sum_{j=1}^{i-1}\sigma_j\delta_j}(1 - e^{-\sigma_i\delta_i})$) exactly represents the probability that the ray travels upto the i^{th} point in the ray without being obstructed i.e. lower this term is for a given point, higher chance that the ray terminates at this point. So if the term becomes 0 at a point, this indicates that the ray terminates at this point and hence the depth of this point in the image is a weighted sum of the distance between different samples along the ray weighted by the probability that the ray did not terminated at those points upto this point i.e. depth of a ray can be given by

$$d = \sum_{i=1}^{N} e^{-\sum_{j=1}^{i-1}\sigma_j\delta_j}(1 - e^{-\sigma_i\delta_i}))z_i \tag{4}$$

Computing this for each ray in the image hence gives us the depth map of the given scene.

Below in Figure(2) are attached two depth maps for two different scenes.

5 What are the major issues you find when using NeRF? List at least 2 drawbacks. For each of them, propose a possible improvement. You are encouraged to check follow-up papers of NeRF, but you should cite these works if borrowing their ideas.

5.1 Answer

- The rendering procedure used by neural radiance fields (NeRF) samples a scene with a single ray per pixel and may therefore produce renderings that are excessively blurred or aliased when training or testing images

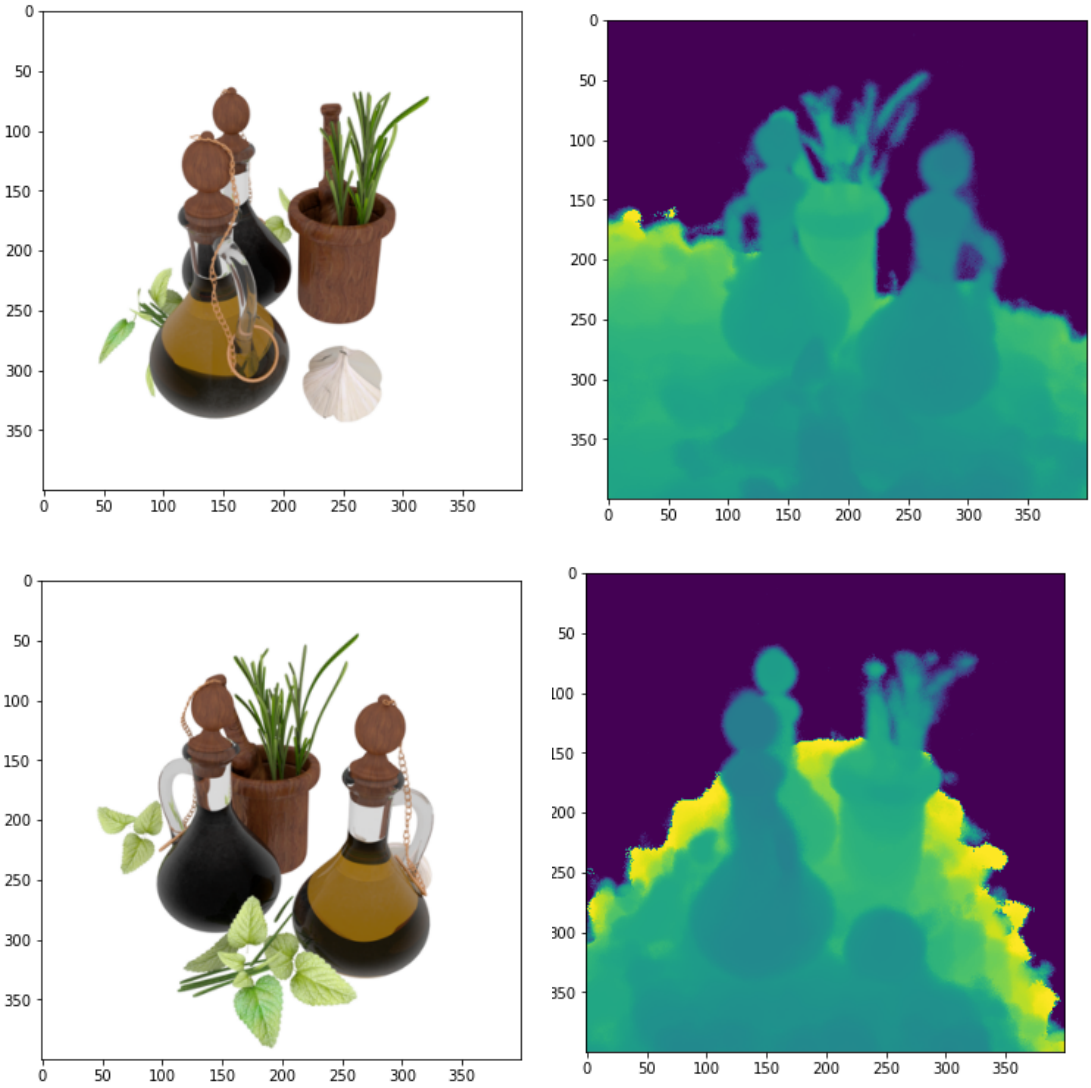

Figure 2: Ground truth and corresponding depth map of image scene

observe scene content at different resolutions. One may try to solve this problem by using multiple rays per pixel. But then we would have to query the MLP for the color space and density output for each ray and this increases the computational complexity a lot.

Solution Proposed Typical positional encoding as used in Neural Radiance Fields maps a single point in space to a feature vector, where each element is generated by a harmonic embedding with an exponentially increasing frequency.

Barron et al. [**?**] proposed their integrated positional encoding that considers Gaussian regions of space, rather than infinitesimal points to the positional encoder. This provides a natural way to input a "region" of space as query to a coordinate-based neural network, allowing the network to reason about sampling and aliasing. The expected value of each positional encoding component has a simple closed form:

$$E_{x\sim\mathcal{N}(\mu,\sigma^2)}\gamma_\omega(p) = sin(\omega\mu)e^{-\frac{(\omega\sigma)^2}{2}} \tag{5}$$

Rather than casting an infinitesimal ray through each pixel, they instead cast a full 3D cone. For each queried point along a ray, they consider its associated 3D conical frustum. Two different cameras viewing the same point in space may result in vastly different conical frustums. In order to pass this information through the NeRF network, they fit a multivariate Gaussian to the conical frustum and use the integrated positional encoding described above to create the input feature vector to the network.

This work is called **Mip-NeRF** and is described here : `https://jonbarron.info/mipnerf/`

- Optimizing a coordinate-based network from randomly initialized weights for each new signal is highly inefficient. The network weights θ are typically optimized via gradient descent to produce the desired image. However, finding good parameters can be computationally expensive, and the full optimization process must be repeated for each new target.

  Solution Tancik et al. [**?**] propose a meta-learning based weight initialisation that makes it easier to optimize the query network function weights. The simple-NeRF formulation relies on multi-view consistency for supervision and therefore fails if naively applied to the task of single view reconstruction. However, if the model is trained starting from meta-learned initial weights, it is able to recover 3D geometry. The MV Meta initialization has access to multiple views per object during meta-learning, whereas the SV Meta initialization only has access to a single view per object during meta-learning.

  This work is called **Learned Initializations for Optimizing Coordinate-Based Neural Representations** and is described here : `https://www.matthewtancik.com/learnit`