Depth Supervision for Neural Radiance Fields

1st Shounak Naik
Robotics Department
Worcester Polytechnic Institute
Worcester, MA
ssnaik@wpi.edu

2nd Ziming Zhang
ECE Department
Worcester Polytechnic Institute
Worcester, MA
zzhang15@wpi.edu

***Abstract*—Neural Radiance Field (NeRFs) is the state of the art technique for 3D Reconstruction. NeRFs learn an implicit represenatation of the scene in their weights. Our work focuses on inducing a depth loss into the NeRF training. NeRFs are notoriously slow to train. We propose on estimating a point cloud of the scene beforehand from the set on input images and then use the geometric prior information and pass it onto the NeRF via a depth loss. We utilize a NeRF transformer framework for this work.**

***Index Terms*—Neural Radiance Fields, Colmap, Scale, Transformation**

I. Introduction

NeRFs give a RGB image output given a camera pose. The Neural Network learns the radiance field of the scene. When we render images from a variety of poses, we get a output which is equivalent to a dense 3D reconstruction. Although, they have the capacity of rendering detailed scenes, they can be very slow to train taking multiple days on a single GPU. Thus there is a huge effort towards speeding up NeRF training. NeRFs are trained on a set of sparse input views (RGB images and pose of that camera). The NeRF is supposed to learn a general mapping between the poses and the RGB image. When the NeRF is learning the mapping between the poses and the RGB image, it is actually learning the geometric information of the scene. We can get an idea of the geometry of the scene by running COLMAP. Our idea is to use the output of the COLMAP which is a point cloud during the training of the NeRF. This effectively would be equivalent to giving the NeRF model a geometric prior of the scene.

II. Related Works

Depth Supervision for NeRF was first put forth in [1]. This is the main inspiration of our work. In this work they use COLMAP to generate a point cloud of the object given various input views of the object. They compare the ray termination depth with the point cloud depth from each input/training pose of the scene and thus they have devised a depth loss. The way they have formulated the ray termination is the first sample on a ray where the σ or the transmittance sees a sudden spike. In NeRFs each 3D point in the scene has a learnable parameter of transmittance which indirectly conveys information on whether or not there is a material at that 3D point. They show that this transmittance - material intuition works well for NeRF scenes by plotting a transmittance vs ray depth graph along various rays. They always see a single sharp spike in the transmittance and at that depth they consider to be some material present. Since the COLMAP relied depth would be noisy, they have formulated a KL Divergence loss which measures the distance between the COLMAP depth distribution and the ray termination depth distribution. They show using this method helps the classical NeRF to train faster and also show that a NeRF can be trained with sparser set of input views. This work triggered a lot of subsequent work in the Depth Supervision for NeRF training domain.

The framework used in this work is inspired from [2]. This work scraps out the deterministic volume rendering and replaces it by a ray transformer. Each sample along the ray is treated as a token to for this network. This network would eventually learn a good attention mechanism to weight each sample and then finally render a RGB value for a given ray. NeRFs training and inferences are specific for a particular scene and are not generalizable to another scene. To tackle this problem, this work has introduced transformers into their implementation. They argue that removing classical volume rendering and replacing it with a feed forward network with attention has the capacity to generalize NeRFs to a greater extent. They show the same in their results as well. Further details about this work would be discussed in the Methodology since this framework is essential to our work.

Since training NeRFs can be notoriously slow, there is a lot of work which tries to reduce the training time. There is a huge effort on optimizing computation relating to the sampling along the ray and the summation of the renders of each sample along the ray, which translates to optimizing cuda kernels for this specific operations. While these works are mostly are mostly engineering, there is a interesting mathematical formulation given in [3] which significantly reduces the training time. TensoRF uses a 4D tensor to represent a 3D voxel grid with per-voxel multi-channel features. The article also details how TensoRF factorizes the 4D scene tensor into multiple compact low-rank tensor components. This method improves over previous methods by reducing the memory footprint.

III. Methodology

In this section, we would talk about the detailed approach that we took for this work.

Code for this work can be found here

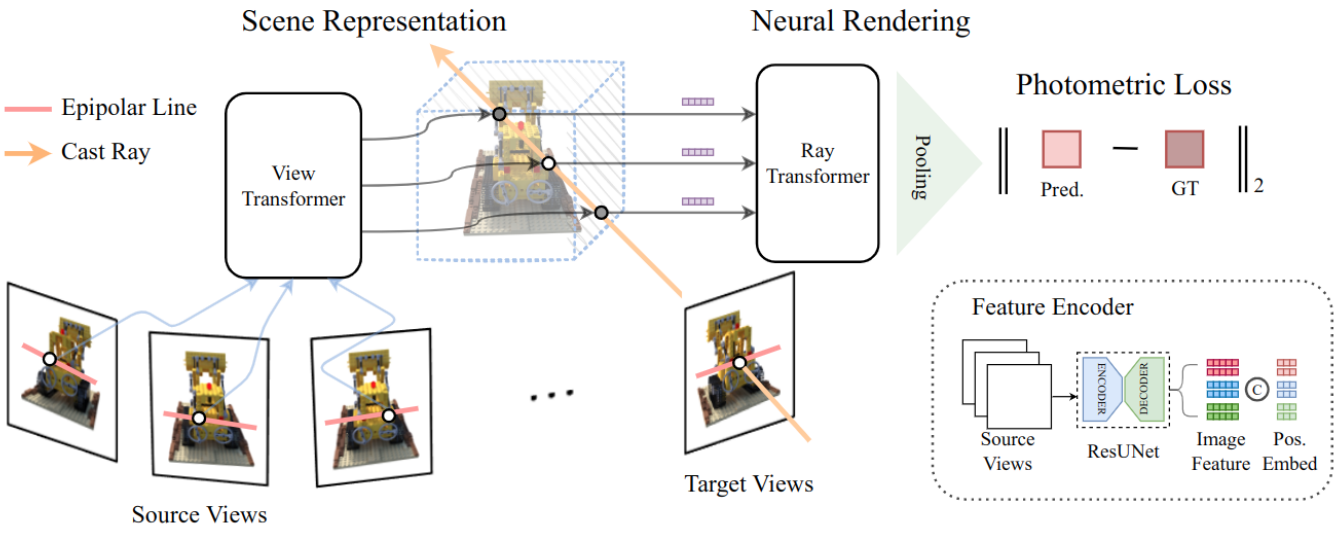


Fig. 1: Overview of Generalizable NeRF Transformer technique

A. *Generalizable NeRF Transformer*

There is a set of input views given to the framework. This framework then converts this data such that for each pose there is a set of Source Views. When training for rendering the Target View, we take the help of the Source Views. The overview of this framework is illustrated in Figure 1. A ray is marched along a pixel from a target pose and samples are taken along this ray. Each sample point is a 3D point which is then projected onto the source views. The white sample is projected onto the three source view in the figure. The RGB pixels of these projected points is given to a feature extractor and these embeddings are then given to the View Transformer. The job of this view transformer is to map a relationship between the embeddings from various views and output a single embedding which should summarize the feature details of the sample point as seen in the source views.

The Ray Transformer replaces the classical volume rendering step into a learnable feed forward step with attention. Each of the sample points would have a embedding at this stage of the network. The Ray transformer learns the attention that should be given for each of these embeddings. The output from this transformer is then fed onto a short MLP which reconstructs the RGB image which should be seen from the Target View/Pose. A photometric loss is used to estimate the correctness of the reconstructed image. This loss is back propagated through the entire network to optimize the network to get the best outputs.

A metric called PSNR is used primarily to evaluate NeRF networks. The PSNR (Peak Signal-to-Noise Ratio) is calculated using the following equation:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right) \tag{1}$$

where:

- MAX is the maximum possible pixel value of the image (e.g., 255 for an 8-bit image).
- MSE is the Mean Squared Error between the original and reconstructed images.

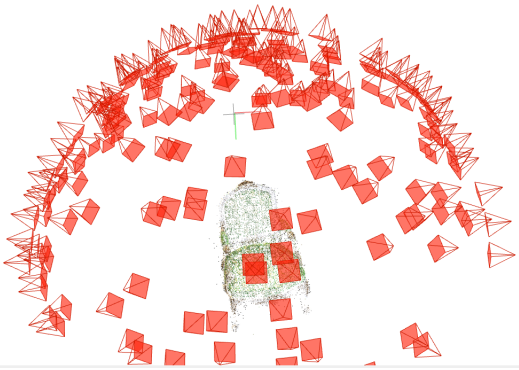


Fig. 2: COLMAP Point Cloud of the Chair

B. *COLMAP and Depth*

COLMAP is software that can generate a point cloud given a set of input rgb images. COLMAP runs SfM (Structure from Motion) underneath which is a photogammetry method which involves Triangulation, Perspective -n-Point and Bundle Adjustment. We chose the chair dataset from the *nerf_synthetic* dataset and generated a point cloud from it. This point cloud can be seen in Figure 2. From this generated point cloud, we can find out the depth of the point cloud in various poses. From the transformer NeRF we can find out the depth by getting a weighted average of the 3D samples (t_i) from the

ray along with the weights which are considered to be the learned attention. Thus, the intuition is to pass the COLMAP depths as the ground truth and then formulating a depth loss based on the comparison between the COLMAP depth and the learned attention based depth.

$$\text{depth} = \frac{\sum_{i=1}^{n} attention_i \cdot t_i}{\sum_{i=1}^{n} attention_i} \tag{2}$$

C. Connecting COLMAP and NeRF frames

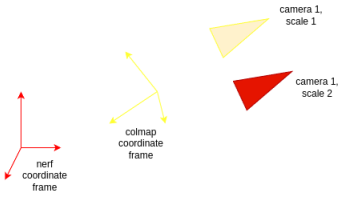


Fig. 3: COLMAP and NeRF coordinates frames at a different scale

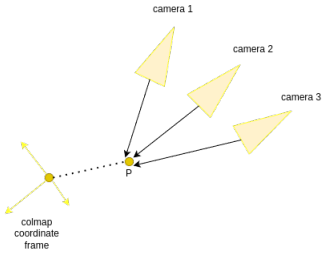


Fig. 4: Equalling scale of COLMAP and NeRF

We want the 3D point cloud and the poses in the same coordinate frame for our problem. The poses that we get from COLMAP can be noisy as they are estimated by SfM, while the poses from the *nerf_synthetic* are exact. Thus we want to find a transformation between the NeRF coordinate frame and the COLMAP coordinate frame. This problem is illustrated in Figure 3. We see that from Figure 2 the COLMAP coordinate frame is set a random floating point. We also don't know the scale at which COLMAP has estimated the points and the poses. NeRF synthetic dataset has a special property that coordinate frame is set at center of the object, in this case a chair. Additionally, its a well known fact the average distance between the origin and the cameras is set to 4.0 and all cameras are pointed towards the coordinate frame origin. We use this fact to register the connect the two frames. In the COLMAP coordinate frame, we find the center of attention P by finding the intersection of all rays coming from the various cameras. This procedure is illustrated in Figure 4. We shift the origin to point P. We also want to rotate the coordinate frame such that z axis points up as we seen in nerf coordinate space. For this we add the y components from the camera_to_world poses. These components describe the image plane y in colmap coordinate frame. If we add these up across cameras, we would get a vector pointing downwards since we have a 360 view. Comparing this vector with $[0, 0, 1]$, we can get a rotation

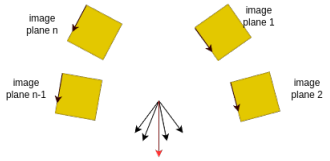


Fig. 5: Rotation so that z axis points up

matrix to get the required coordinate frame. This method is illustrated in Figure 5. We rotate the poses and points as well. Now we measure the average distance of the cameras from this new coordinate frame and scale it back to 4.0 which is the scale that we want to achieve. Similarly the points in the point clouds are also scaled back with the same scaling ratio.

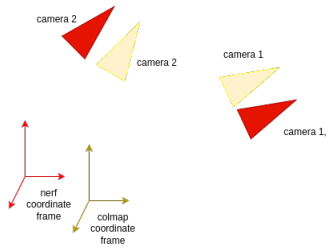


Fig. 6: NeRF compliant COLMAP frame

At this point we have COLMAP coordinate frame which is compliant with the *nerf_synthetic* dataset coordinate system. NeRFs train better when the dataset is compliant and thus having this coordinate frame is advantegous. At this point, we could theoretically connect the two coordinate frames as $T_{\text{nerf}}^{\text{colmap}} = T_{\text{nerf}}^{\text{camera}} * T_{\text{camera}}^{\text{colmap}}$. But getting an estimate of this transformation was highly noisy as the COLMAP poses are noisy. We saw emperically that using this transformation does not give a good result when this transformation is used on the 3D point cloud. This problem is illustrated in Figure 6.

Thus we resort to using the colmap coordinate frame and the corresponding poses and the 3D point cloud for further work. This decision was taken after referring to a similar approach taken in [1]. We visualize the entire scene in this coordinate frame and it is seen in Figure 7. The blue points are the camera poses.

In this frame, we register the depth of this point cloud in each of the camera poses. The depth map for each of the poses is generated by considering for outliers, we only keep the depth values between the 10^{th} and 90^{th} percentile. This helps with filtering out noisy point estimates during the SfM process. Additionally, we keep the mimimum of the depth values at every pixel in the image plane. For 100 camera poses, COLMAP took 5 minutes to run and the depth image registration took another 4 minutes.

D. MSE Depth Loss

The simplest Loss function is used to compare the attention derived depth and the colmap depth, we use a mean-squared-

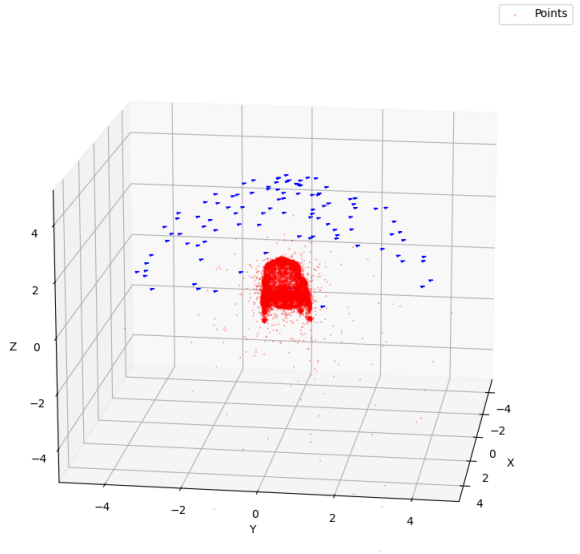


Fig. 7: NeRF compliant COLMAP coordinate frame and chair object

error loss on this.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(attention\_depth - colmap\_depth)^2 \quad (3)$$

E. KL Divergence Loss

Since the 3D point cloud can be a noisy estimate of the actual geometry, MSE loss can be numerically unstable. Instead, we can convert the depths into probablity distributions and then find the distance between these 2 probabilty distributions. KL divergence, formally denoted as $D_{KL}(P||Q)$, captures the informational difference between two probability distributions, P and Q. It tells us how much extra information (measured in bits) we need on average, to represent samples drawn from P using code designed for Q. Intuitively, if P and Q are identical, no additional information is required, resulting in a KL divergence of zero. Conversely, larger KL divergences indicate a significant difference between the distributions, highlighting the inefficiency of using Q's code for P's data. The formulation for our problem would be as follows.

$$D_{KL}(P||Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{Q(x)}{P(x)}\right) \quad (4)$$

Here, P is colmap_depth since this is the ground truth distribution we want our attention_depth to match. Thus Q is attention_depth.

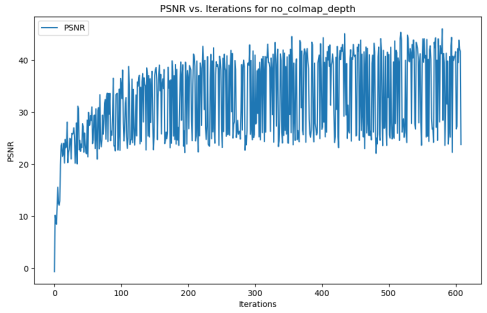
IV. Experiments and Results

The dataset we worked on had 100 different camera poses. Thus each epoch had 100 different poses. At a time, we sampled 512 rays from these camera poses and had 64 3D point sampples along these rays. We trained our framework for 600 epochs with an initial learning rate of 1×10^{-4}.

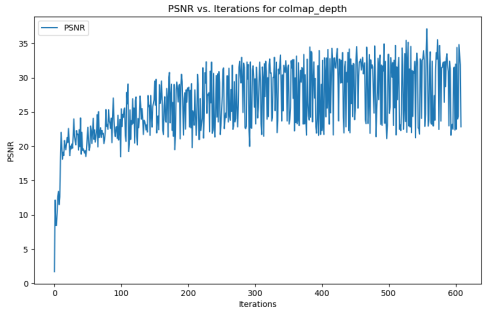
We use either the KL Divergence Loss or the MSE Loss along with the RGB Loss. We suspect this would decrease training time.

$$\text{Total Loss} = \text{RGB Loss} + \lambda * \text{Depth Loss} \quad (5)$$

Using the KL Divergence loss actually decreased the PSNR values. This can be seen in Figure 8. We suspect this to happen due to the nature of the batching during neural network training. We use batches of 512 rays, we get the COLMAP depth and the attention depth of these 512 rays and then convert these arrays into a probability function to calculate the KL Divergence loss. Since the rays are randomly sampled, the pseudo ground truth depth distribution keeps on changing, essentially making this a moving target problem for our neural network. This suspicion is clearly illustrated by the noisy nature of the KL Divergence Loss seen in Figure 9



(a) PSNR for Default Case



(b) PSNR with KL Divergence Loss

Fig. 8: PSNR with and without KL Divergence Loss

Because of our earlier observations, we resort to the MSE loss. The MSE Loss converges but unfortunately we don't see any significant improvement over the default training method. We see that the PSNR values are improving with respect to the KL Divergence case but there isn't a significant improvement over the default case. This can be seen in Figure 11.

The visual results for the MSE case can be seen in Figure 12. We suspect that with a sparser set of input views, we would see a significant improvement in PSNR settling time with the

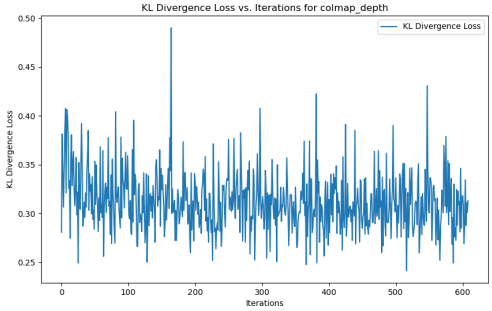


Fig. 9: KL Divergence Loss

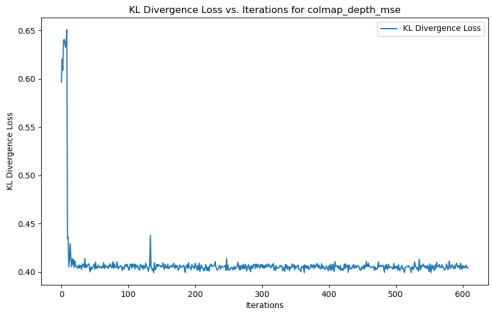


Fig. 10: MSE Depth Loss

proposed depth loss. Unfortunately, even with 25 poses (75% decrease), we see very similar results to the ones shown above.

V. Conclusions and Future Work

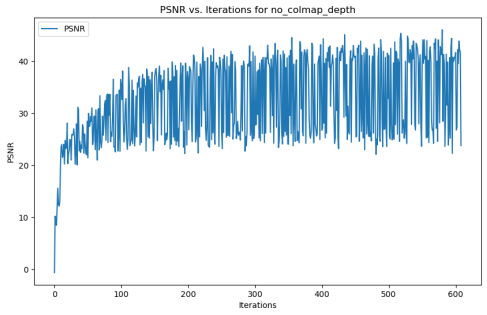
We have proposed inducing a geometric prior which has the potential of speeding up NeRF training. It can also train high quality NeRFs with a sparser set of input views. The results in this work points us to the conclusion that the View Transformer and the Ray transformer already bridges the gap which was supposed to be completed by the depth loss. This again, points to the power of the attention mechanism when used with the right formultation. We also observe that the transformer helps significantly with the training time. With just a 100 epochs, we start to see the general shape of the scene. In the future, it is imperative to think of a way to use a probabilistic loss function across depths since the ground truth can be noisy as well. In the future, it can be great to systematically analyze these suspected advantages of using a transformer backend for the NeRF frameworks.

References

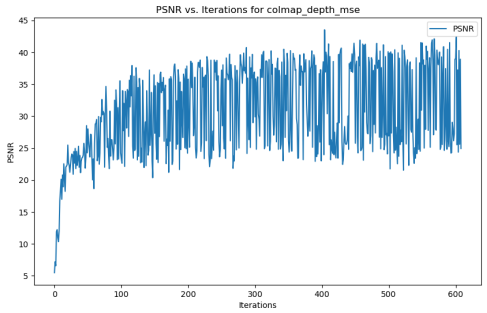
[1] Deng, K., Liu, A., Zhu, J., & Ramanan, D. (2022). Depth-supervised NERF: fewer views and faster training for free. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr52688.2022.01254

[2] T, M. V., Wang, P., Chen, X., Chen, T., Venugopalan, S., & Wang, Z. (2022, July 27). Is attention all that NERF needs? arXiv.org. https://arxiv.org/abs/2207.13298

[3] Chen, A., Xu, Z., Geiger, A., Yu, J., & Su, H. (2022, March 17). TeNSORF: Tensorial Radiance Fields. arXiv.org. https://arxiv.org/abs/2203.09517



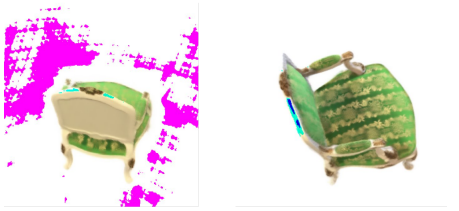
(a) PSNR for Default Case



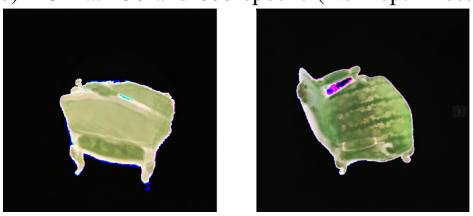
(b) PSNR with MSE Loss

Fig. 11: PSNR with and without MSE Loss

(a) Ground Truth RGB



(b) RGB at 150 and 600 epochs (No Depth Loss)



(c) RGB at 150 and 600 epochs (MSE Depth Loss)

Fig. 12: RGB comparisons