<Final Report: EEE4610-01>

Tensor Radiance Field Reconstruction without Camera Prior

Jiho Park, Jung Yi, Hyeryung Jeon

School of Electrical and Electronic Engineering

College of Engineering

Yonsei University

<Final Report: EEE4610-01>

Tensor Radiance Field Reconstruction without Camera Prior

Report Advisor: Sangyoun Lee

December 2023

Jiho Park, Jung Yi, Hyeryung Jeon
School of Electrical and Electronic Engineering
College of Engineering
Yonsei University

Contents

ABSTRACT

Tensor Radiance Field Reconstruction without Camera Prior

The emergence of NeRF (Neural Radiance Field) [1] has enabled photorealistic novel view synthesis, leading to significant advancements in 3D reconstruction. To overcome the limitations of traditional implicit neural representations, diverse grid-based hybrid representations(e.g. Plenoxel, InstantNGP, TensoRF [2]) have been proposed, resulting in dramatically faster training and rendering speeds. However, these methods still rely on camera parameters obtained through Structure-from-Motion(SfM), which is a non-differentiable camera calibration process. This often becomes a bottleneck in end-to-end training for deep learning-based 2D to 3D reconstruction. Recently, there exist researches focusing on deep learning-based camera calibration as an alternative to SfM. In this study, we are presenting the exploration of factors that can improve the training of TensoRF [2] (one of the state-of-the-art grid-based representation) in environments without camera information. The explored factors are depth information which has strong geometry prior and point cloud back-projection which can be utilized by 3D-matching loss and 2D photometric loss.

Key words : 3D Reconstruction, NeRF, Camera Pose Estimation, Depth Prior

1. Introduction

The demand for 3D digital content has surged across various domains including entertainment, medicine, architecture, archaeology, and robotics, to name a few. Therefore, in today's digitally driven world, the ability to create a comprehensive three-dimensional (3D) representation of real-world objects and scenes has become an invaluable tool for a multitude of applications. This process, known as 3D reconstruction, seeks to capture the geometry, texture, and appearance of a subject from a series of two-dimensional (2D) images or other types of input data. However, the journey from 2D imagery to a fully-realized 3D model is filled with complexities. The algorithms and techniques that underpin this transformation must address issues related to perspective, scale, occlusion, and illumination.

This research field is experiencing a revival since the appearance of the Neural Radiance Field(NeRF). NeRF uses a neural network as an implicit function of the 3D scene. Nowadays, the hybrid representation, which composed of both explicit(e.g. hash-grid, tri-plane) and implicit(e.g. neural network) functions, has become the state-of-the-art method in both speed and quality. These methods require camera parameters of input views. Consequently, most of the 3D reconstructions rely on Structure-from-Motion(SfM) to get camera parameters. Thus most of NeRF [1] methods are vulnerable to the failure of SfM(e.g. low-textured scene). For this reason, research on camera pose estimation is also being actively explored. In this study, we are mainly focusing on applying the pose estimation methods to the efficient 3D representations which is Tensor Radiance Field(TensoRF) [2].

First of all, we replaced the camera extrinsic with trainable parameters, and simultaneously optimize them while training. We mainly applied two different concepts for efficient camera optimization. First is the depth guidance from pre-trained monocular depth estimation network. Since the depth map has strong geometry prior, it helps to optimize unknown camera poses. Due to this property, depth maps are well used in few-shot reconstruction or other camera pose estimation tasks. Second is the back-projected point cloud matching. This technique is mainly proposed by Wenjing et al [5]. We explored how these concepts work on TensoRF [2].

2. Related Work

2.1. 3D Representations

2.1.1. NeRF: Neural Radiance Field [1]

(1) Introduction

The task this paper seeks to address is 'novel view synthesis', which involves generating images according to new viewing directions for a fixed scene. The neural network used in NeRF takes the viewing direction and 3D coordinates as inputs and produces color and volume density as outputs. By utilizing these two outputs and going through the volume rendering process, 2D images can be generated based on the viewing direction. At this time, since volume rendering is composed of a differentiable function, the neural network can be trained through traditional backpropagation methods. The images used for training are 2D images from various viewing directions of a single scene.

(2) Neural Radiance Field (NeRF)

Neural Radiance Fields(NeRF) is a fully connected neural network, which is an implicit function of the 3D scene. In this setup, the volume density is determined solely by the coordinates, while the color structure takes into consideration both the coordinates and the viewing direction. Initially, it outputs the volume density according to the 3D coordinates, and then takes the remaining representation along with the viewing direction as inputs to output the RGB color.

(3) Volume Rendering

Volume Rendering is a rendering process that generates a 2D image for a predetermined viewing direction, and it doesn't require training. Firstly, it determines the color that each ray (path of light) will depict. In this process, the color C(r) that the ray will depict is calculated considering the color c(r(t), d) and the volume density σ(r(t)) at every point along the ray's path. If the volume density from the starting point of the view to a specific point is high, the color at that point will be obscured and not visible. Taking this into consideration, the cumulative transmittance T(t) is calculated and incorporated as described below.

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

$$\text{where } T(t) = \exp\left(-\int_{t_n}^{t}\sigma\big(r(s)\big)d\,s\right), \qquad r(t) = \mathrm{o} + t\mathrm{d}$$

The Volume Rendering equation derived from NeRF is not integrable, so it is approximated using numerical integration. In this process, instead of using the coordinates of the divided sections directly, random sampling within the sections is utilized. This is done to prevent learning from fixed discontinuous points only and to facilitate learning of a continuous scene representation. In the final integration equation, traditional alpha compositing is applied, and C(r) is calculated as follows.

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

$$\hat{C}(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))c_i\,, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right)$$

Below is the figure of application of NeRF and volume rendering.

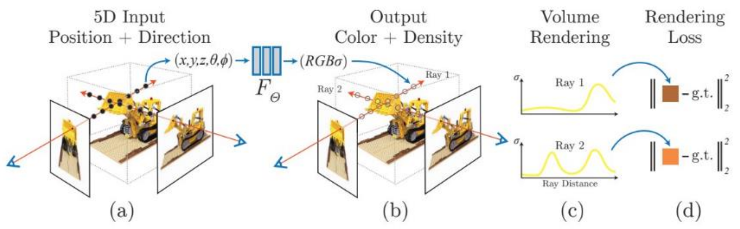


Figure 1. NeRF Representation & Volume Rendering Overview.

(4) Optimizing NeRF

In this paper, two techniques are applied for performance improvement, namely, NeRF optimization: Positional Encoding and Hierarchical Sampling.

A. Positional Encoding

Using only the traditional 3D orthogonal coordinate system can lead to the loss of coordinate information as it passes through a deep neural network. Therefore, by utilizing positional encoding to increase the dimensions as shown below, the performance has been improved. At this time, positional encoding is applied not only to the 3D coordinates but also to the viewing direction, and the final structure of the neural network is as follows.

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

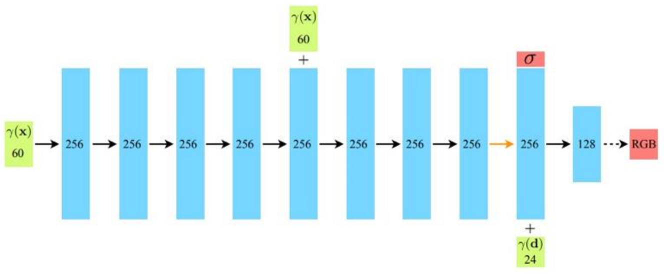


Figure 2. MLP Architecture in NeRF.

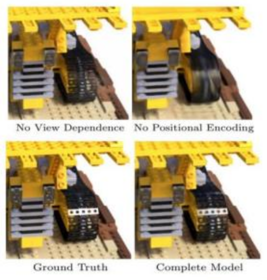
B. Hierarchical Sampling

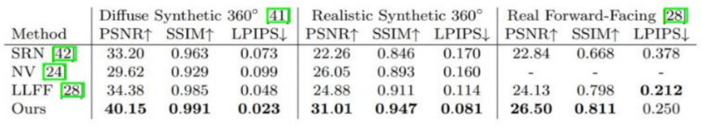
Hierarchical sampling is implemented for efficient sampling. To achieve this, course and fine networks are utilized. The course network learns the scene on a general level. In the fine network, additional sampling is applied in sections where the volume density is high based on the rendering results from the course network. Through this process, the fine network can learn more finely in areas where objects are present. Ultimately, in the final network used, sampling points from both course and fine stages are utilized.

$$\mathcal{L} = \sum_{r \in \mathcal{R}} \left[|\widehat{C_c}(r) - C(r)|_2^2 + |\widehat{C_f}(r) - C(r)|_2^2 \right]$$

(5) Experiment

First, experiments were conducted to determine the effectiveness of the methodology that considers the direction of view when outputting color, and of Positional Encoding. Through the qualitative and quantitative results below, it can be confirmed that utilizing both of the aforementioned methods improves performance. Then, it can be confirmed that it exhibits state-of-the-art (SOTA) performance, surpassing the performance of existing methods.





Method	Diffuse Synthetic 360° [41] PSNR↑	SSIM↑	LPIPS↓	Realistic Synthetic 360° PSNR↑	SSIM↑	LPIPS↓	Real Forward-Facing [28] PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

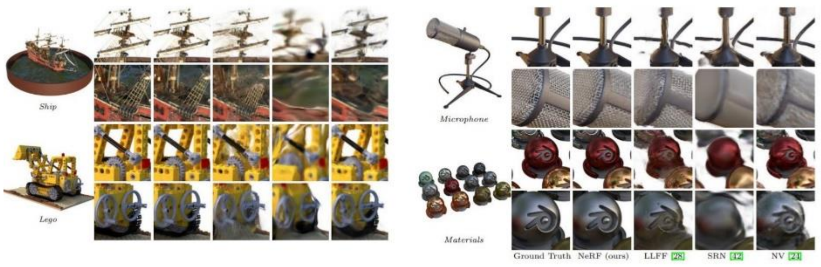


Figure 3. Test-Set Scene View Comparisons.

2.1.2. TensoRF: Tensorial Radiance Field [2]

The "TensoRF" technique takes a distinct path from the Neural Radiance Fields (NeRF) approach, which predominantly uses Multilayer Perceptrons (MLPs). In contrast, TensoRF depicts a scene's radiance field using a 4-dimensional tensor. This design essentially integrates a 3D voxel grid with individual multi-channel attributes for each voxel. The standout feature of TensoRF is its ability to break down this 4D tensor into several reduced-rank tensor elements, using both the CANDECOMP/PARAFAC (CP) method and innovative Vector-Matrix (VM) decomposition strategies.

(1) CP and VM Decomposition

CP Decomposition factorizes a 3D tensor into a sum of outer products of vectors. Mathematically, a tensor T is decomposed into vectors v_r^1, v_r^2, v_r^3 across its three modes (I, J, K). Each element in the tensor is a sum of scalar products formed by these factorized vectors. VM Decomposition takes a more flexible approach by allowing tensors to be factorized into vectors as well as matrices. Unlike CP Decomposition, which uses pure vector factors, VM employs matrix factors for two out of the three modes, making it more versatile. This increases the parameterization capability, enabling the model to represent more complex high-dimensional data.

(2) Tensorial Radiance Field Representation

A. Factorization of Radiance Field

The geometry grid g_σ is a 3D tensor, while the appearance grid g_c is a 4D tensor. Both are factorized using Vector-Matrix (VM) decomposition. The vector and matrix elements capture the spatial layout of scene structure and look, while the appearance feature-mode vectors highlight universal appearance connections. All appearance feature-mode vectors are stacked together to form a matrix B, serving as a global appearance dictionary.

B. Efficient Feature Evaluation

The approach allows for direct and efficient evaluation of the density σ and appearance c values for any voxel using the decomposed factors. For g_σ, a single voxel's density value can be directly calculated using its factorized form. For g_c, a full P-channel feature vector is computed using a single matrix operation involving B.

(3) Rendering and reconstruction

A. Volume Rendering

For the rendering process, a ray traverses and collects Q shading points along its path to derive each pixel. The pixel color C is then calculated using:

$$\sigma, c = \sum_r \sum_m \mathrm{A}_{\sigma,r}^{m}(\mathrm{x}), S\left(\mathrm{B}\left(\oplus\left[\mathrm{A}_{c,r}^{m}(\mathrm{x})\right]_{m,r}\right), d\right)$$

$$C = \sum_{q=1}^{Q} \tau_q (1 - \exp(-\sigma_q \Delta_q)) c_q, \ \tau_q = \exp\left(-\sum_{p=1}^{q-1} \sigma_p \Delta_p\right)$$

In this context, σ_q represents the density and c_q denotes the color at the sampled points x_q. Meanwhile, Δ_q indicates the step size, and τ_q stands for the transmittance.

B. Reconstruction

The radiance field is refined using gradient descent to reduce the L2 rendering discrepancy compared to the actual pixel colors. Tensor factorization techniques are applied, using global vectors and matrices for correlation and regularization.

2.2. Camer Pose Estimation in NeRF

2.2.1. NeRF—[3]

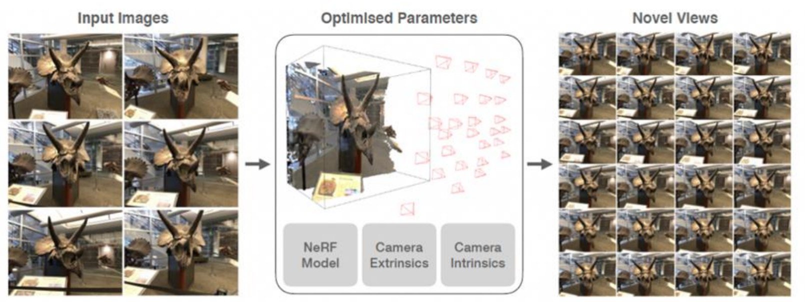


Figure 4. NeRF—optimizes camera extrinsic & intrinsic

NeRF-- presents a framework where camera pose and intrinsics are regarded as learnable parameters. Instead of treating them as fixed inputs, they are optimized alongside the 3D scene representation, first by estimating the camera parameters for every image, and then by refining the volumetric scene depiction through a photometric reconstruction loss.

This framework simultaneously optimizes both the camera parameters and the scene representation from a set of input RGB images, denoted as G. This approach is built on the understanding that every image is captured in a forward-facing setup with a certain degree of rotation and translation, and that all images are taken using identical intrinsic parameters.

$$\Theta^*, \Pi^* = \arg\min_{\Theta,\Pi} \mathcal{L}(\hat{\mathcal{I}}, \hat{\Pi}|\mathcal{I}),$$

where P denotes the camera parameters, encompassing both its intrinsic values and its 6 DOF positions.

(1) Camera parameters

A. Camera Intrinsic:

Camera intrinsic is characterized by the focal length f and the principal points cx and cy relating to the pinhole camera model. Without sacrificing generality, it can approximate cx as W/2 and cy as H/2, where H and W represent the image's height and width, respectively. In this context, estimating the camera's internal parameters essentially translates to determining the focal length f.

B. Camera poses:

Camera poses can be expressed in SE(3) through the camera-to-world transformation matrix T_wc = [R | t]. Specifically, optimizing the translation vector t, defined in Euclidean space, is straightforward when setting it as trainable parameters. Conversely, for the camera rotation, it adopts the axis-angle representation denoted as f := aw in R^3, where w symbolizes the normalized rotation axis and a indicates the rotation angle. The rotation matrix R can be derived using Rodrigues' formula, wherein ()^ acts as a skew operator, transforming the vector f into a skew matrix.

$$\mathbf{R} = \mathbf{I} + \frac{\sin(\alpha)}{\alpha}\phi^\wedge + \frac{1-\cos(\alpha)}{\alpha^2}(\phi^\wedge)^2,$$

(2) Joint Optimization

It co-optimizes the camera parameters with the NeRF for a sparse set of forward-facing views. To produce a pixel p from a given image I_i, the model projects a ray $\widehat{r_{1,p}}(h) = \widehat{o_1} + h\widehat{d_{1,p}}$ through the pixel towards the radiance field from the camera. This uses the current camera parameters $\widehat{\pi_1} = (\hat{f}, \widehat{\phi_1}, \widehat{t_1})$ and the direction of the radiance field:

$$\hat{\mathbf{d}}_{i,p} = \hat{\mathbf{R}}_i \begin{pmatrix} (u - W/2)/\hat{f} \\ -(v - H/2)/\hat{f} \\ -1 \end{pmatrix},$$

It's noteworthy that $\widehat{o_1} = \widehat{t_1}$ is equivalent to $\widehat{R_1}$ and can be derived from $\widehat{\phi_1}$ using Rodrigues' formula. As it traces along the ray, it samples 3D points {x_j} and evaluate both {c_j} and {s_i}. During the training phase, it renders M pixels randomly for each input image and strive to minimize the reconstruction loss between the colors and their actual hues. A crucial point to highlight is that its pipeline is differentiable, which facilitates co-optimization of NeRF and the camera parameters by targeting the minimization of the reconstruction loss.

(3) Blender Forward Facing Dataset

It introduces a dataset called BLEFF. This dataset encompasses 14 scene paths that have been rendered with varying degrees of rotation and translation perturbations. Detailing the methodology behind constructing the dataset: It begins with a perfect forward-facing camera trajectory, meaning all cameras are oriented forward and move within the xy-plane at z=0. To this impeccable trajectory, it introduces 5/5/6 levels of rotation/translation/full-6DoF perturbation, yielding 16 rotation trajectory for each scene.

2.2.2. SCNeRF: Self-Calibrating Neural Representation Field [4]

(1) Introduction

To learn the geometry of a scene from multi-view 2D images, it is necessary to have camera parameters for each image. In NeRF (Neural Radiance Fields), these camera parameters are set using COLMAP. The authors propose setting parameters such as the pinhole camera model, non-linear noise and radial distortion as learnable parameters, enabling self-calibration. While traditional self-calibration methods rely on geometric constraints, this study utilizes both geometric and photometric consistency in tandem. The experimental results showed that it was possible to learn the camera parameters even without COLMAP initialization, and the proposed method could determine more accurate camera parameters, improving upon the performance of the existing baseline models.

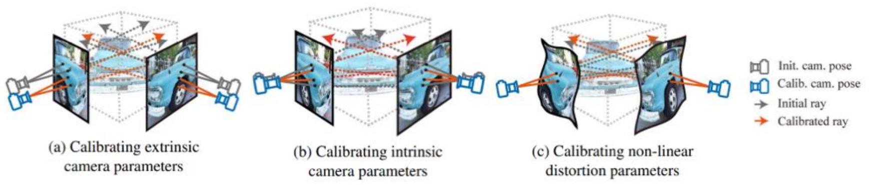

Init. cam. pose
Calib. cam. pose
Initial ray
Calibrated ray
(a) Calibrating extrinsic camera parameters
(b) Calibrating intrinsic camera parameters
(c) Calibrating non-linear distortion parameters

Figure 5. Calibration process of three different camera parameters

(2) Preliminary

To test the proposed self-calibration method, experiments were conducted using NeRF (Neural Radiance Fields) and NeRF++ as the baseline models. Therefore, the volume rendering methodology utilizing opacity was commonly applied in both cases. Moreover, in NeRF++, to achieve good performance even in unbounded spaces, the model learns by distinguishing the background.

$$\hat{C}(r) \approx \sum_{i}^{N} \left(\prod_{j=1}^{i-1} \alpha\left(r(t_j), \Delta_j\right) \right) \left(1 - \alpha\left(t_i, \Delta_i\right)\right) c(r(t_i), v)$$

(3) Differentiable Self-calibrating Cameras

A. Intrinsic Matrix(K) and Rotation(Extrinsic) Matrix(R)

For the intrinsic matrix K, it is parameterized by dividing it into $K = K_0 + \Delta K$.

$$K = \begin{bmatrix} f_x + \Delta f_x & 0 & c_x + \Delta c_x \\ 0 & f_y + \Delta f_y & c_y + \Delta c_y \\ 0 & 0 & 1 \end{bmatrix}$$

For the rotation matrix, a 6-vector representation is applied to maintain the orthogonality of the matrix.

$$f\left(\begin{bmatrix} | & | \\ \mathbf{a}_1 & \mathbf{a}_2 \\ | & | \end{bmatrix}\right) = \begin{bmatrix} | & | & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \\ | & | & | \end{bmatrix},$$

$$R = f(\mathbf{a}_0 + \Delta\mathbf{a}), \quad \mathbf{t} = \mathbf{t}_0 + \Delta\mathbf{t}.$$

Therefore, the mapping from pixel to ray can be represented as follows. In this case, K0, R0, and t0 are not optimized.

$$\mathbf{r}_d = RK^{-1}\mathbf{p}, \quad \mathbf{r}_o = \mathbf{t}.$$

Because cameras use a circular lens, radial distortion, or circular distortion, occurs. To correct this, the fourth-order distortion model from COLMAP is utilized, which is known as the fish-eye model.

$$k = \left(k_1 + z_{k_1}, k_2 + z_{k_2}\right)$$
$$n = \left((p_x - c_x)/c_x, \left(p_y - c_y\right)/c_y, 1\right)$$
$$d = \left(1 + k_1 n_x^2 + k_2 n_x^4, 1 + k_1 n_y^2 + k_2 n_y^4\right)$$
$$p' = \left(p_x d_x, p_y d_y, 1\right)$$
$$r_d = RK^{-1}p', \qquad r_o = t$$

B. Generic Non-linear Ray Distortion

In actual lenses, distortions occur that are impossible to represent with parametric camera models. To correct this kind of noise, a non-linear raxel parameter is introduced. Therefore, the residual between the distortion and the calculated ray is established as a parameter, and the values in the continuous space are extracted using bilinear interpolation.

$$r_d' = r_d + z_d, \qquad r_o' = r_o + z_o$$

To summarize, the camera parameters for Self-Calibration are computed through the following process.

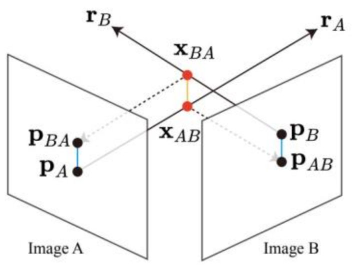
$$\begin{tikzcd}
 & & z_k \arrow[d] & & \mathbf{p} \arrow[d] & & & \\
 & k \arrow[r] & + \arrow[r] & \mathbf{k} \arrow[r] & \text{Radial Distortion} \arrow[d] & & & \\
 & z_K \arrow[d] & & & \mathbf{p}' \arrow[d] & & z_{\mathbf{r}_d} \arrow[d] & \\
K \arrow[r] & + \arrow[r] & K \arrow[rr] & & R \cdot K^{-1} \cdot \mathbf{p}' \arrow[r] & \mathbf{r}_d \arrow[r] & + \arrow[r] & \mathbf{r}_d \\
R|t \arrow[r] & + \arrow[r] & R|t \arrow[rr] \arrow[urr] & & t \arrow[r] & \mathbf{r}_o \arrow[r] & + \arrow[r] & \mathbf{r}_o \\
 & z_{R|t} \arrow[u] & & & & & z_{\mathbf{r}_o} \arrow[u] &
\end{tikzcd}$$

Figure 6. Computation graph of camera parameters and noise

(4) Geometric and Photometric Consistency

Geometric Consistency: Projected Ray Distance

In a 3D space, when a point is projected onto images as p_A and p_B through different camera parameters, each point is referred to as a corresponding point. If the camera calibration is perfect, the rays r_A and r_B corresponding to each pixel p_A and p_B would intersect at one point. Therefore, the paper proposes a loss term that minimizes the distance between the two rays. At this point, as the distance from each image A and B increases, the distance between the rays can also increase. Ultimately, instead of directly utilizing the spatial distance corresponding to the shortest distance between the two rays at coordinates x_AB and x_BA, the average of the distances projected onto each image is used as the loss term.



$$d_\pi = \frac{\|\pi_A(\mathbf{x}_B) - \mathbf{p}_A\| + \|\pi_B(\mathbf{x}_A) - \mathbf{p}_B\|}{2}$$

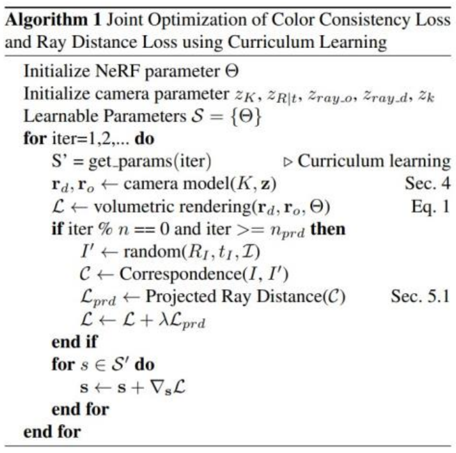
Figure 7. An illustration of the proposed Projected Ray Distance (PRD).

Photometric Consistency

Similar with NeRF, Photometric Loss is used to learn Geometry and Appearance. Simultaneously, the camera parameters are learned from the ray.

(5) Optimization

If the geometry is unknown or the camera's self-calibration is coarse, it is not possible to properly learn the camera parameters. Therefore, the geometry and linear-camera model are learned first, followed by the learning of the non-linear camera parameters. In the context below, Curriculum Learning refers to the addition of distortion and noise correction parameters to the camera parameters.



Algorithm 1 Joint Optimization of Color Consistency Loss and Ray Distance Loss using Curriculum Learning

```
Initialize NeRF parameter $\Theta$
Initialize camera parameter $z_K$, $z_{R|t}$, $z_{ray\_o}$, $z_{ray\_d}$, $z_k$
Learnable Parameters $\mathcal{S} = \{\Theta\}$
for iter=1,2,... do
    S' = get_params(iter)                              ▷ Curriculum learning
    $\mathbf{r}_d, \mathbf{r}_o \leftarrow$ camera model($K, \mathbf{z}$)                    Sec. 4
    $\mathcal{L} \leftarrow$ volumetric rendering($\mathbf{r}_d, \mathbf{r}_o, \Theta$)          Eq. 1
    if iter $\% \ n$ == 0 and iter $>= n_{prd}$ then
        $I' \leftarrow$ random($R_I, t_I, \mathcal{I}$)
        $\mathcal{C} \leftarrow$ Correspondence($I$, $I'$)
        $\mathcal{L}_{prd} \leftarrow$ Projected Ray Distance($\mathcal{C}$)         Sec. 5.1
        $\mathcal{L} \leftarrow \mathcal{L} + \lambda\mathcal{L}_{prd}$
    end if
    for $s \in \mathcal{S}'$ do
        $\mathbf{s} \leftarrow \mathbf{s} + \nabla_{\mathbf{s}}\mathcal{L}$
    end for
end for
```

(6) Experiments

Experiments were conducted using two outdoor scenes, the LLFF dataset, and the Tanks and Temples dataset. The performance of the proposed camera self-calibration was found to be higher than the initialization using COLMAP. Moreover, according to the ablation study results regarding the distortion and noise correction parameters, a higher quality view was generated when more correction parameters were applied.

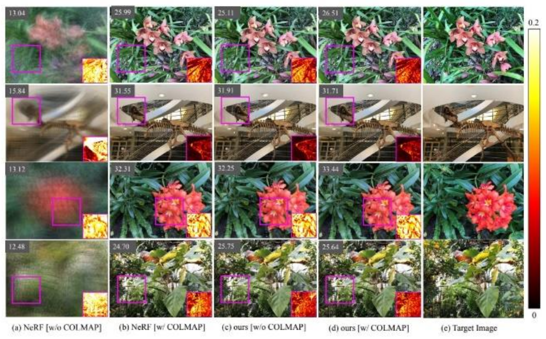


Figure 8. Comparison of NeRF and SCNeRF approach using LLFF dataset

2.2.3. NoPe-NeRF: Optimising Neural Radiance Field With No Pose Prior [5]

(1) Introduction

The existing unposed-NeRF models (e.g., NeRFmm, BARF, SC-NeRF) have the issue of being able to handle only forward-facing scenes. There are two reasons causing this. The first is that they do not consider the relative poses of the images, treating each image as an independent input. The second is that the radiance field itself suffers from shape-radiance ambiguity, meaning that predicting the camera parameters can potentially exacerbate the ambiguity. In this paper, authors use mono-depth effectively to address these problems. There are three main contributions: 1) Integrating mono depth to unposed-NeRF 2) Supplying relative poses to camera-NeRF optimization 3) Regularising ‘relative pose estimation’ using depth-based surface rendering.

(2) Undistortion of Monocular Depth

Mono-depth estimation provides a powerful geometry prior, reducing shape-radiance ambiguity. However, since this depth estimation occurs independently for each individual image, it lacks multi-view consistency. Therefore, instead of using the predicted depth map, an undistorted depth map is obtained to be used as the ground truth depth map. Here, the undistorted depth map is D*, which has applied shape and shift distortion parameters alpha and beta. Both alpha and beta are trainable parameters. Additionally, authors argue that jointly optimizing NeRF with alpha and beta enhances multi-view consistency (from the below loss).

$$D_i^* = \alpha_i D_i + \beta_i,$$

$$\mathcal{L}_{depth} = \sum_i^N \left\| D_i^* - \hat{D}_i \right\|,$$

where

$$\hat{D}_i(\mathbf{r}) = \int_{h_n}^{h_f} T(h)\sigma(\mathbf{r}(h))dh$$

(3) Relative Pose Constraint: Point Cloud Loss, Surface based Photometric Loss

Point Cloud Loss

For consecutive images, the approach is that the point clouds formed should be similar since the observed objects are similar. Firstly, depth maps are generated from the images taken by consecutive cameras. Through back-projection, a point cloud for each depth map is constructed. Since the poses of the two point clouds are different, their poses are aligned. Ultimately, the difference between the two constructed point clouds is taken as a loss. At this point, the Chamfer Distance metric is utilized.

$$\mathcal{L}_{pc} = \sum_{(i,j)} l_{cd}(P_j^*, \mathbf{T}_{ji} P_i^*),$$

where $\mathbf{T}_{ji} = \mathbf{T}_j \mathbf{T}_i^{-1}$

$$l_{cd}(P_i, P_j) = \sum_{p_i \in P_i} \min_{p_j \in P_j} \|p_i - p_j\|_2 + \sum_{p_j \in P_j} \min_{p_i \in P_i} \|p_i - p_j\|_2.$$

The point cloud loss provides supervision through 3D-3D matching. Going further, to also apply photometric consistency, the appearance difference of associated pixels is taken as a loss. In other words, the point cloud is projected back onto the image, and the difference concerning the projected pixels is calculated. Here, '<>' operator denotes the sampling operation in the image.

$$\mathcal{L}_{rgb-s} = \sum_{(i,j)} \| I_i \langle \mathbf{K}_i P_i^* \rangle - I_j \langle \mathbf{K}_j \mathbf{T}_j \mathbf{T}_i^{-1} P_i^* \rangle \|,$$

Below is the final loss.

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda_1 \mathcal{L}_{depth} + \lambda_2 \mathcal{L}_{pc} + \lambda_3 \mathcal{L}_{rgb-s},$$

3. Method

Not only to enhance the reconstruction quality, but also to investigate each effect of the losses, we accumulately applied the losses. There are 4 different losses, rendering loss(rgb loss), monocular depth loss, 3D point cloud matching loss and 2D point cloud reprojection loss.

3.1. Camera Pose Optimization

Fundamentally, to optimize the camera extrinsic matrix should be set as trainable parameters. Generally, camera extrinsic matrix is 4x4 matrix, but it's degree of freedom isn't 16. The extrinsic of the camera could be determined by 3-dimensional rotation and translation. In this case, we used the SO(3) group representation for rotation which requires 3 dimensions, and world translation vector for translation. As a result, while training with the original rendering loss, those 6 parameters will be optimized.

3.2. Depth Guidance

To provide rich geometric information, pre-trained monocular depth estimator model DPT [6] is adopted. DPT [6] model is used for ground truth depth map. For the efficiency of memory of the gpu, we've processed the depth map in advance and save it before training. Since there might exist bias on the predicted depth map, the depth guidance might derive confusion for the reconstruction. Therefore, we applied the distortion parameter for ground truth depth as in [5]. This parameter enables the shift and scale on depth map(Sec. 2.2.3.). The ground truth depth is also used for following back-projected point cloud loss.

3.3. Back-projected Point Cloud Guidance

Back-projected rays from matched pixels are often used for camera estimation [4]. However, to utilize the back-projected point cloud, depths of each pixels are required. Thanks to the existence of the ground truth depth, we can easily back project the given pixels. This concept, exploiting depth information to the fullest extent, was proposed by Wenjing et al [5]. By comparing back projected point clouds between two adjacent frames, more geometric information could be injected.

First is the back-projected point cloud loss from adjacent images. Point cloud could be easily generated with optimized camera extrinsic and the depth value. Then, Chamfer Distance is applied to every point clouds, measuring the distance from the nearest point cloud from adjacent image.

Second is the surface-based photometric loss. This loss measures difference in appearance between associated pixels by reprojecting the point cloud into two adjacent cameras. Wenjing et al [5] argue that this loss can alleviate the mismatch between back-projected point clouds. Overall loss equations are elaborated on Sec 2.2.3.

4. Experiments and Results

Train settings were selected by which was affordable with our resources. The experiment was done with Fern and Flower datasets from LLFF [7], which is the real-world forward-facing dataset. The train-evaluation split ratio was set to 1/8. For the rendering evaluation, we measured the train and eval PSNR values. For pose evaluation, we set the camera pose values extracted from Colmap(SfM) as ground truth. For pose evaluation metric, Absolute Trajectory Error(ATE) and Relative Pose Error (RPE) were used. The quantitative results are shown in Tab. 1 and Tab. 2.

Losses	Train PSNR ↑	Eval PSNR↑	ATE ↓	RPE_r ↓ (rotation)	RPE_t ↓ (transltion)
RGB	15.57	10.11	NaN	NaN	NaN
RGB + Depth	**21.18**	15.75	0.108	92.78	7.53
RGB + Depth + Point Cloud Back-projection	19.91	15.69	0.068	**3.07**	5.55
RGB + Depth + Point Cloud Back-projection & Reprojection	20.55	**15.86**	**0.037**	**3.07**	**2.37**

Table 1. Experiment Result on Fern dataset

Losses	Train PSNR ↑	Eval PSNR↑	ATE ↓	RPE_r ↓ (rotation)	RPE_t ↓ (translation)
RGB	14.32	3.96	NaN	NaN	NaN
RGB + Depth	**22.88**	**17.53**	0.045	52.11	1.924
RGB + Depth + Point Cloud Back-projection	20.04	15.27	0.041	2.83	1.844
RGB + Depth + Point Cloud Back-projection & Reprojection	21.93	17.19	**0.021**	**0.50**	**0.587**

Table 2. Experiment Result on Flower dataset

4.1. RGB Loss Only

Since there are no other techniques, pure optimization of the parameterized camera didn't work well. It's loss relatively didn't converged. Also, camera pose metrics were not able to be calculated due to the large error (We presume that the metric values have gone to infinity). Below figure is the qualitative result. Through the results, we can confirm that additional prior information is required to reconstruct the Tensor Radiance Field.

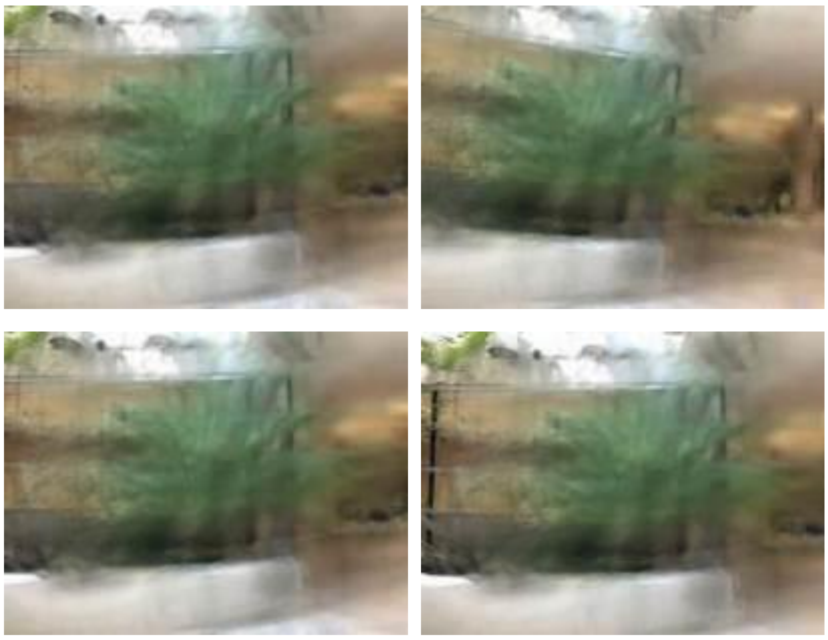


Figure 9. Rendering Results for fern with RGB loss.

Above two are from trainset and below two are from testset.

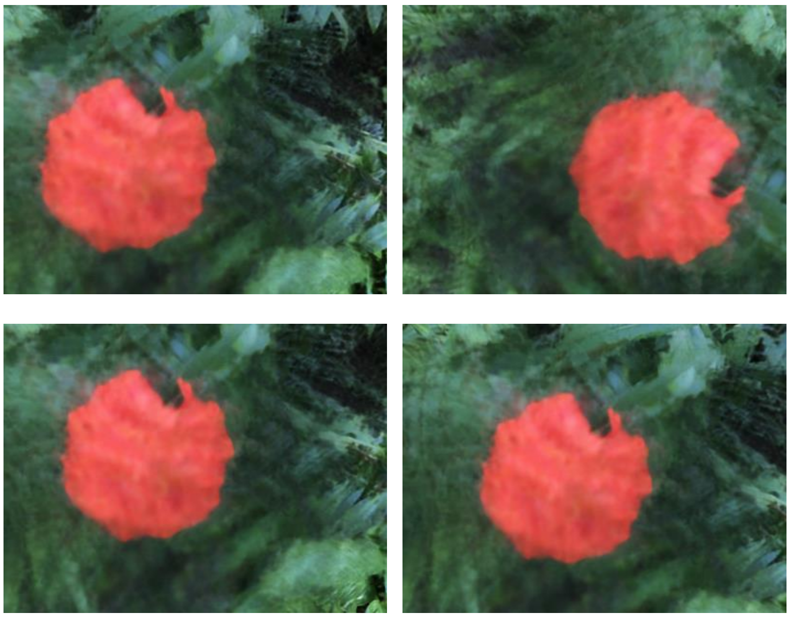


Figure 10. Rendering Results for flower with RGB loss.

Above two are from trainset and below two are from testset

4.2. Depth Guidance

The addition of the depth loss has performed best on PSNR metrics in both flower(Tab. 1) and fern(Tab. 2) dataset. However, we can see that its' camera rotation and translation errors(RPE_r, RPE_t) are much higher than others (especially the rotation). This is the case that have fallen into local minimum of loss. Although the evaluation PSNR is high, model has found out completely incorrect geometric solution which somehow satisfies the given train and test views. We argue that this could occur because the LLFF [7] data, especially the fern data, is a slightly sparse dataset. The qualitative result (Fig. 11, Fig. 12) also shows that the qualities of the image are not even, although some shows notable performance. As a result, although depth loss achieved high PSNR, we consider the result as a failure in reconstruction.

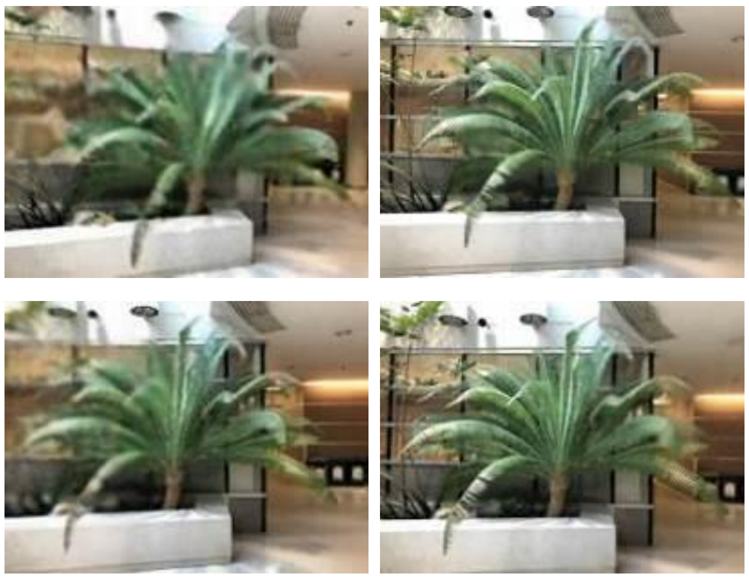


Figure 11. Qualitative Result of Depth Guidance Experiment on Fern Data.

Above two are from trainset and below two are from testset.

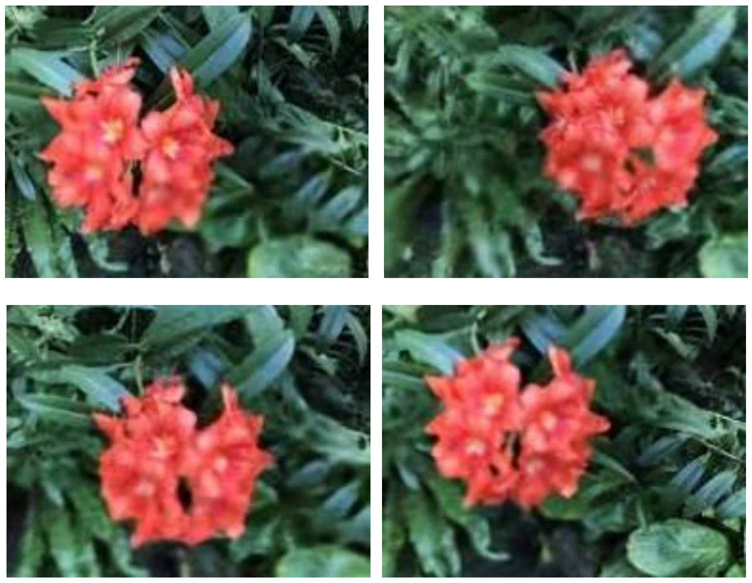


Figure 12. Qualitative Result of Depth Guidance Experiment on Flower Data.

Above two are from trainset and below two are from testset.

4.3. Point Cloud Reprojection Guidance

To compensate for the camera pose error which occurred by the depth loss, we experimented two losses derived from the back-projected point cloud. First, back-projected point cloud matching loss provides additional geometric information. Second, rendering loss from the reprojection of the point cloud is applied sequentially. (To remind, this method was proposed by Wenjing et al [5]).

In both fern(Tab. 1) and flower(Tab. 2) dataset, application of back-projected point cloud matching loss remarkably improves the camera pose estimation. However, the PSNR value drops a lot while adopting the loss. The rendering quality decrement could be observed in both qualitative(Fig. 13, Fig. 14) and quantitative results.

Finally, the implementation of the reprojected point cloud loss stabilized the training, while estimating accurate camera parameters. Qualitative results are shown in Fig. 15, Fig. 16. Some results (especially for fern dataset) show worse quality compared to depth guidance results. However, the model precisely estimated the camera poses, while remaining the rendering quality.

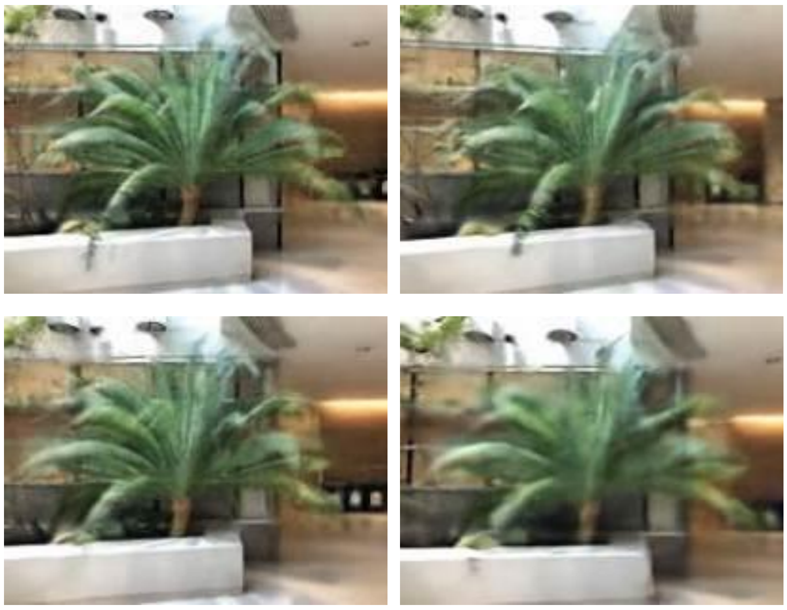


Figure 13. Qualitative Result of Back-projected Point Cloud Guidance on Fern data.

Above two are from trainset and below two are from testset.

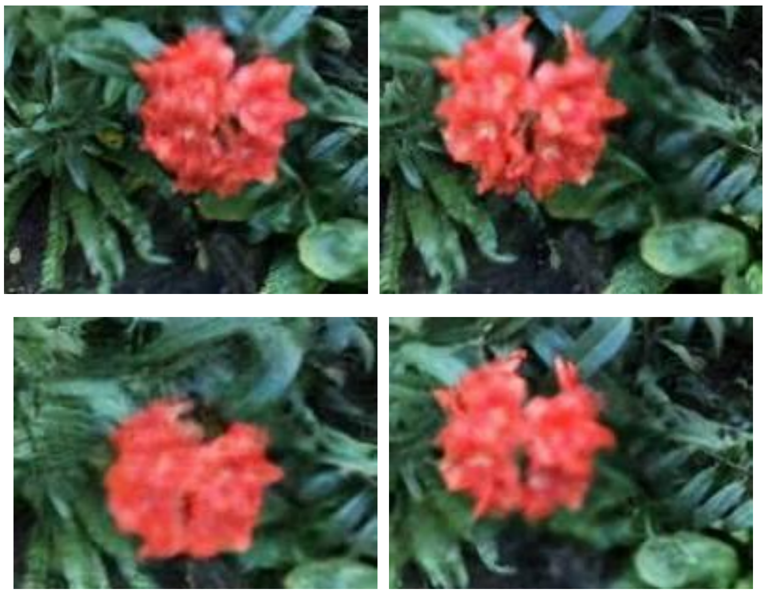


Figure 14. Qualitative Result of Back-projected Point Cloud Guidance on Flower data.

Above two are from trainset and below two are from testset.

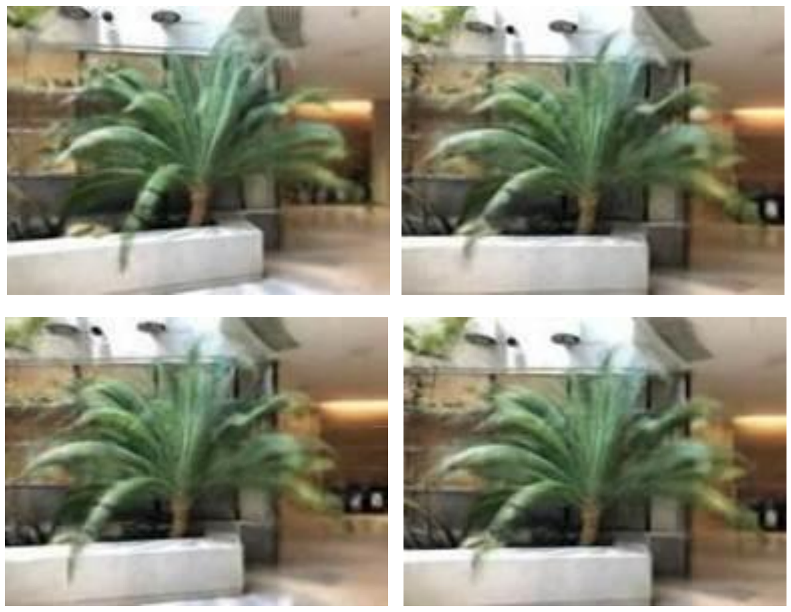


Figure 15. Qualitative Result of Point Cloud Reprojection Guidance on Fern data.

Above two are from trainset and below two are from testset.

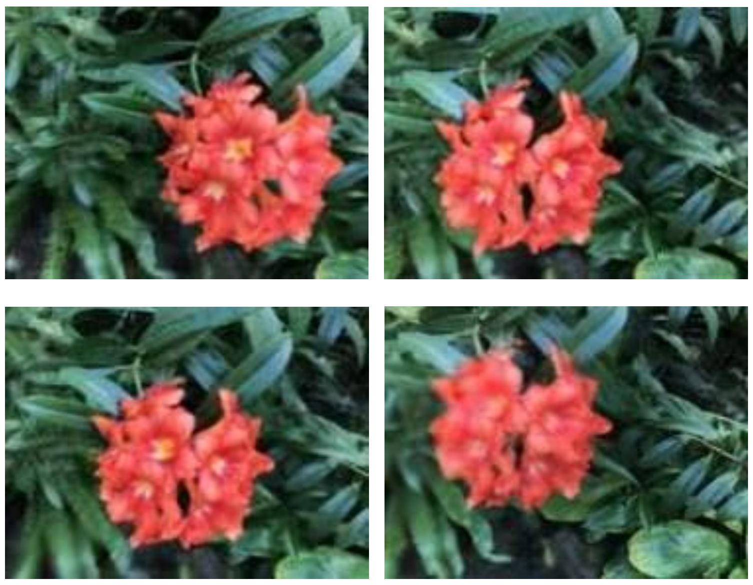


Figure 16. Qualitative Result of Point Cloud Reprojection Guidance on Flower data.

Above two are from trainset and below two are from testset.

5. Discussion

Depth Guidance on TensoRF: Surprising thing about the depth guidance experiment is that even it had totally incorrect camera pose, both train and test PSNR were high. We argue that this phenomenon is because of the sparsity of LLFF [7]. Due to the sparsity of the train images, there exists various camera solutions which only satisfies the given views. Also, another presumable possibility is the property of TensoRF [2]. Unlike NeRF [1], explicit grid allows more direct manner of optimization. The depth loss might have caused a larger gradient flow to the features on grid, which led to notable increment in rendering quality.

Limitations and Future Work: Due to the lack of resources, we've compared the results before complete convergence. Fern dataset was evaluated at the iteration of 250k and Flower dataset was evaluated at the iteration of 350k. Although it was enough to confirm the effect of each losses, the final performance might be improved with more iterations. Moreover, the experiment on other dataset might show different results. We would leave those and the verification of our arguments about the depth guidance, as future work.

6. Conclusion

Grid-based representations for 3D reconstruction have advantages in rendering speed and training speed, which makes them state-of-the art. However, they mostly rely on the SfM for camera calibration. In this study, we've explored the camera optimization process on Tensor Radiance Field.

We've sequentially enhanced the Tensor Radiance Field reconstruction without pose prior. The effect of three guidance, depth guidance, back-projected point cloud matching guidance and reprojected point cloud guidance were investigated. First, depth guidance provides strong geometry prior which highly enhances the rendering quality without the camera prior. On the other hand, it lacks information which eventually leads to completely incorrect camera pose. Second, back-projected point cloud guidance, which is the exploitation of the given depth information, helps the model to achieve accurate camera pose. However, it gives up the rendering quality in large extent. Finally, reprojected point cloud guidance enables the model to confirm the photometric quality of the back-projected point cloud. As a result, the model has achieved accurate estimation of camera pose and sustained reconstruction quality.

Reference

[1] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”, *In European conference on computer vision (ECCV)*, pages 405-421, 2020.

[2] Chen, Anpei, et al. "Tensorf: Tensorial radiance fields.", *European Conference on Computer Vision (ECCV)*, pages 333-350, 2022.

[3] Wang, Zirui, et al. "NeRF--: Neural radiance fields without known camera parameters." *arXiv:2102.07064*. 2021

[4] Jeong, Yoonwoo, et al. “Self-Calibrating Neural Radiance Fields”, *In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5846-5854, 2021.

[5] Wenjing Bian et al. “NoPe-NeRF: Optimising Neural Radiance Field With No Pose Prior”, *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4160-4169, 2023.

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[7] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.