

MC-NeRF: Muti-Camera Neural Radiance Fields for Muti-Camera Image Acquisition Systems

Yu Gao^{1,2}, Lutong Su^{1,2}, Hao Liang^{1,2}, Yufeng Yue^{1,2}, Yi Yang^{1,2*}, Mengyin Fu^{1,2}

arXiv:2309.07846v1 [cs.CV] 14 Sep 2023

Abstract—Neural Radiance Fields (NeRF) employ multi-view images for 3D scene representation and have shown remarkable performance. As one of the primary sources of multi-view images, multi-camera systems encounter challenges such as varying intrinsic parameters and frequent pose changes. Most previous NeRF-based methods often assume a global unique camera and seldom consider scenarios with multiple cameras. Besides, some pose-robust methods still remain susceptible to suboptimal solutions when poses are poor initialized. In this paper, we propose MC-NeRF, a method can jointly optimize both intrinsic and extrinsic parameters for bundle-adjusting Neural Radiance Fields. Firstly, we conduct a theoretical analysis to tackle the degenerate case and coupling issue that arise from the joint optimization between intrinsic and extrinsic parameters. Secondly, based on the proposed solutions, we introduce an efficient calibration image acquisition scheme for multi-camera systems, including the design of calibration object. Lastly, we present a global end-to-end network with training sequence that enables the regression of intrinsic and extrinsic parameters, along with the rendering network. Moreover, most existing datasets are designed for unique camera, we create a new dataset that includes four different styles of multi-camera acquisition systems, allowing readers to generate custom datasets. Experiments confirm the effectiveness of our method when each image corresponds to different camera parameters. Specifically, we adopt up to 110 images with 110 different intrinsic and extrinsic parameters, to achieve 3D scene representation without providing initial poses. The Code and supplementary materials are available at <https://in2-viaun.github.io/MC-NeRF/>

Index Terms—NeRF, Muti-camera system, intrinsic parameters regression, poses regression, 3D scene representation, volume rendering.

1 Introduction

Visual 3D reconstruction assists autonomous driving, multi-robot navigation, and security surveillance in understanding their environment. These scenes often contain a large number of cameras, which can provide abundant visual information, especially in the form of multi-view perspectives. Recently, Neural Radiance Field (NeRF) [1] demonstrated high quality 3D scene representation capabilities. The essence of NeRF is to learn and reconstruct spatial information by utilizing cross light constraints generated from multi-view images. This information is stored within a neural network to achieve the implicit representation of the scene.

*This work was partly supported by National Natural Science Foundation of China (Grant No. NSFC 62233002, 61973034, U1913203 and CJSP Q2018229)

¹School of Automation, Beijing Institute of Technology, Beijing, China

²State Key Laboratory of Intelligent Control and Decision of Complex System, Beijing Institute of Technology, Beijing, China

*Corresponding author: Y. Yang Email: yang yi@bit.edu.cn

Muti-Camera Images Without Intrinsic and Extrinsic Parameters

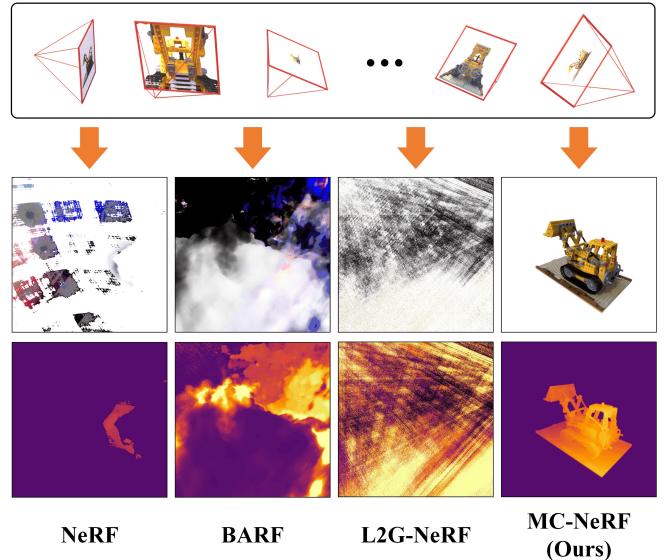


Fig. 1: We propose MC-NeRF, which can jointly optimize both intrinsic and extrinsic parameters for bundle-adjusting Neural Radiance Fields. The challenge introduced by multi-camera images is that each image corresponds to different intrinsic and extrinsic parameters, which breaks the original NeRF assumption of a globally unique camera.

An important precondition for NeRF series methods is the acquisition of images from different views. These images often need to be captured using global unique cameras, ensuring uniform camera intrinsic parameters across all images. Specifically, widely utilized NeRF datasets, like Synthesis [1], LLFF [2], NSVF [3], and Mip-NeRF 360 [4], employ unique camera intrinsic parameters for each scene, ensuring the ray distribution model is fixed in rendering process. Furthermore, some classical 3D reconstruction approaches also adopted the global unique camera assumption. For instance, Structure from Motion (SFM) [5] assumes the same intrinsic parameters for each image to estimate the poses of individual images. The DTU dataset [6] utilized by MVSNet [7] employs a unique camera for each distinct scene. Based on this unique camera condition, MVSNet constructs a differentiable homography matrix to create a cost volume.

However, in practical scenarios, ensuring that the multi-view images are captured by global unique cameras is not always feasible. A typical application is the multi-camera image acquisition system. We believe the following con-

cerns related to multi-camera acquisition systems: Firstly, each image in the multi-view data captured by the system corresponds to different intrinsic and extrinsic parameters. To establish the ray distribution model within the NeRF series methods, accurate parameters for each image are required. But the calibration process for multi-camera systems can be notably time-consuming. Secondly, the multiple intrinsic matrices invalidates the assumption of a global unique camera, which affects the reliability of methods such as COLMAP [5] used for estimating extrinsic parameters. Besides, frequent adjustments to cameras, like poses and quantities, even camera lens, are common operations in multi-camera acquisition systems, which is accompanied with system vibrations and random camera shifts. These further increase the frequency of system recalibration. In summary, exploring 3D reconstruction methods suitable for multi-camera systems is necessary.

Focusing on the above problems, we propose Muti-Cam NeRF (MC-NeRF), a 3D reconstruction method designed for multi-camera acquisition systems. The method has the capability to directly perform reconstruction using mixed images captured by different cameras, and each image can correspond to a completely different camera intrinsic parameters without requiring any camera extrinsic parameters information. In addition, MC-NeRF achieves end-to-end training and global optimization. During the training process, the intrinsic and extrinsic parameters of each camera and the final NeRF model can be obtained simultaneously. The optimized intrinsic and extrinsic parameters can be stored as calibration parameters, and in this case we can directly use the NeRF series methods for reconstruction, skipping the optimization process of camera parameters. When cameras in the system change (including intrinsic and extrinsic parameters), it only needs to perform global optimization again to obtain new system calibration parameters.

It's worth noting that our method also has some limitations. Because camera intrinsic parameters can reflect the true scale of images, it is difficult to acquire individual intrinsic parameters for each camera without a standard scale reference. This requires importing additional information for scale calibration. Additionally, during the intrinsic parameter regression process, we encountered a degenerate issue, which prevented us from obtaining valid parameters. This implies that the additional information mentioned above needs to meet certain conditions to avoid degeneration. Moreover, we also found that joint regression of camera parameters leads to parameter coupling, hindering the acquisition of both intrinsic and extrinsic parameters separately. Due to these reasons, our method requires auxiliary calibration images during the optimization process. We also explored the base requirements for calibration and presented a efficient scheme for obtaining these auxiliary images.

As previously discussed, the majority of existing NeRF datasets are generated based on global unique camera, which can not satisfy the requirement of randomized camera parameters with mixed cameras. This has motivated us to propose our own dataset. The MC dataset provides the reader with the flexibility to tailor camera parameters and the number of cameras, allowing for free combinations.

In conclusion, the contributions of this paper are as follows:

- We address the degenerated cases in intrinsic parameter regression and the coupling problem within joint optimization of intrinsic and extrinsic parameters, and also presented corresponding solutions for these challenges.
- We propose a joint optimization strategy that can simultaneously optimize the intrinsic parameters, extrinsic parameters and neural radiation fields.
- We design a calibration cube for multi-camera image acquisition systems and give an efficient calibration image acquisition scheme.
- We propose a new dataset for multi-camera acquisition systems and provide the source code for dataset generation, enabling readers to create their own datasets freely.

2 RELATED WORK

Camera Calibration The camera model demonstrates how points in camera coordinates are projected into pixel coordinates, and camera calibration aims to determine the parameters within this model. Earlier calibration methods observe calibration objects whose geometry in 3D space is known with high precision to calculate the camera parameters, and this efficient method requires only a small number of images to achieve high-performance results [8]. Considering the precision of manufacturing and the cost of calibration objects, Tsai et al. [9] captured images of a 2D planar calibration object from various poses to estimate camera parameters. This method requires providing accurate pose information for each calibration plane. Another widely used approach adopting 2D planar calibration is the chessboard method proposed by Zhang [10]. This technique involves capturing images including a chessboard pattern from different positions to determine the camera parameters. An important precondition of Zhang's method is that multiple chessboard images need to be captured to ensure the accuracy of the solution. Compared with Tsai's method, Zhang's approach eliminates the need to provide the calibrator's position. In addition to employing known calibration points, there are also some methods relying on detecting specific image features to achieve calibration, like vanishing points [11], [12] and coplanar circles [13].

In recent years, deep learning has provided new inspirations for camera calibration. DeepCalib [14] utilizes a CNN network trained with a vast collection of scene images matched with corresponding camera parameters, and established the relationship between image features and camera parameters. DeepCalib can directly predict calibration results based on the input images. An advantage of this method is that there is no need to provide typical calibration objects. Similar to DeepCalib, Hold-Geoffroy et al. [15] also attempted to predict camera parameters directly from a single image, and they explored the boundaries of human tolerance towards image distortion, and further designed a new perceptual measure for the calibration errors.

Apart from methods designed explicitly to obtain calibration results, some works can indirectly acquire camera parameters. Some learning methods [16]–[23] rely on classical multi-view geometry theories, like epipolar constraints, PnP (Perspective-n-Point) algorithms, feature points matching and triangulation,

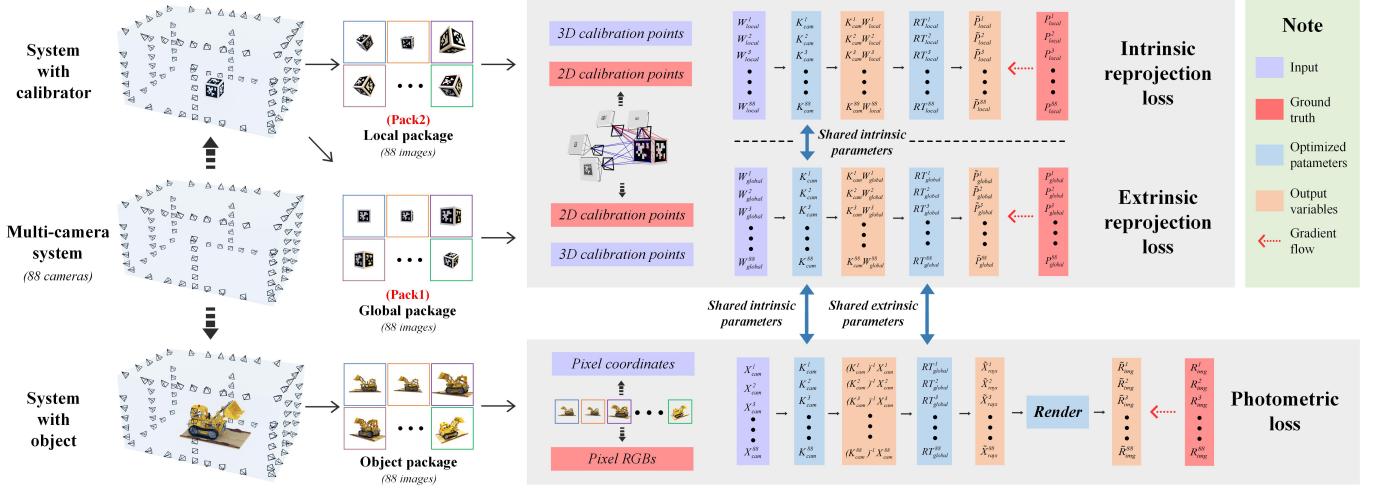


Fig. 2: Overall branches of proposed framework. 1) The left side illustrates data acquisition process using a multi-camera system (Room Style). Each camera in the system is fixed in position, and our method requires two packages of calibration images, and one package of the object. 2) The right side displays the loss function branches of the network, consisting of three components. Firstly, it refines the coarse camera intrinsic and extrinsic parameters by utilizing *Pack1* and *Pack2*. This optimization is executed by the Intrinsic and Extrinsic Reprojection Loss branch. Secondly, with the coarse parameters, the network continues rendering training, incorporating the Photometric Loss branch. This joint optimization can obtain accurate camera parameters and refine the rendering network.

to achieve 3D reconstruction or visual odometry. Although these applications may not design for obtaining camera parameters, they still has the capability to retrieve poses, and the mentioned theories always serve as the foundation for model block design and loss functions design. Additionally, camera pose estimation methods based on NeRF, such as iNeRF [24], have also emerged.

Camera Preconditions of NeRF Acquiring accurate camera parameters can often be challenging, and in some cases, certain parameters might even be unreachable. When camera parameters are unreliable, adopting classic NeRF-series methods [25]–[33] to describe the scene becomes difficult. Some researches [34]–[36] employ input images to estimate camera poses. Li et al. [26] used COLMAP to obtain the initial camera poses, which were subsequently used as precondition for NeRF to reconstruct the scene. Instead of using existing methods directly, some other methods [37]–[40] can leverage prior knowledge to establish the loss function by introducing geometric constraints with photometric consistency. Specifically, Wang et al. [39] jointly optimized camera parameters and NeRF through photometric reconstruction. Jeong et al [41] design a mixed camera model to achieve the self-calibration, and the process is constrained by geometric and photometric consistency for NeRF training. Lin et al [38] employed a coarse-to-fine auxiliary positional embedding [42]–[44] to jointly recover the radiance field and camera pose. NoPe-NeRF [45] uses monocular depth priors to constrain the scene as well as relative pose estimates. Ray-to-ray correspondence losses also can be leveraged to impose constraints on the relative pose estimates by utilizing either keypoint matches or dense correspondences [41], [46]. DBARF [47] adopts low-frequency feature maps to guide the bundle adjustment for

generalizable NeRFs [48], [49]. GNeRF [50] introduces a pose estimator to directly estimate camera poses from images. Essentially, these methods combine the training process with camera parameters prediction.

TABLE 1: Description of symbols

Symbol	Description
K_{cam}^i	intrinsic parameters for each camera.
RT_{local}^i	extrinsic parameters for each camera in <i>Pack2</i>
RT_{global}^i	extrinsic parameters for each camera in <i>Pack1</i> (pose for each camera in the multi-camera system)
<i>Render</i>	neural radiance field
W_{local}^i	world coordinates of Apriltag corners in <i>Pack2</i> (ground truth)
W_{global}^i	world coordinates of Apriltag corners in <i>Pack1</i> (ground truth)
P_{local}^i	pixel coordinates of Apriltag corners in <i>Pack2</i> (ground truth)
P_{global}^i	pixel coordinates of Apriltag corners in <i>Pack1</i> (ground truth)
\tilde{P}_{local}^i	pixel coordinates of Apriltag corners in <i>Pack2</i> (predicted)
\tilde{P}_{global}^i	pixel coordinates of Apriltag corners in <i>Pack1</i> (predicted)

In summary, NeRF series methods have been extensively explored and significantly improved, yet certain limitations remain. Firstly, previous works often require inputs collected from the same camera or cameras with identical performance, which is difficult to ensure in a multi-camera image acquisition system. Secondly, different intrinsic parameters are rarely

considered in prior researches, implying that existing methods aim to standardize all images to a single camera model with uniform parameters.

3 METHOD

3.1 Overview

The main goal of our work is to explore an efficient 3D representation method suitable for multi-camera image acquisition systems. In comparison with previous NeRF series methods, our challenge arises from the unknown intrinsic and extrinsic parameters for each image. The proposed method is illustrated in Fig.2, consisting of three loss functions and can be divided into two branches. In the training stage, we start by jointly training the intrinsic and extrinsic reprojection losses, while masking the rendering branch, to obtain initial intrinsic and extrinsic parameters. Then we employ the initialized parameters to perform optimization of all parameters. The symbol definitions are provided in Table 1.

The reason for employing this strategy is that the inputs for rendering are provided by the intrinsic and extrinsic parameters, as demonstrated in the photometric loss branch. As a downstream component in the sequential flow, the rendering network can be easily influenced by fluctuations in upstream inputs. This implies that training the rendering network may become meaningless when the camera parameters undergo drastic changes. Furthermore, the rendering network adopt coarse to fine structure, which contains a significantly larger number of parameters compared to the intrinsic and extrinsic reprojection loss branches, leading to substantial training time. Therefore, obtaining initialized camera parameters first and subsequently refining them with the rendering network training is an efficient strategy.

Additionally, we employ both the photometric loss branch and the intrinsic reprojection loss branch for joint optimization, with the intrinsic reprojection loss branch serving the purpose of decoupling intrinsic and extrinsic parameters. the issue is detailed discussed in Section 3.4.

3.2 Multi-Camera Intrinsic Parameters Regression

In this section, we first present the theoretical formulas for computing intrinsic parameters. Then, we show how we design learnable parameters. Next, we analyze the degenerate cases during intrinsic parameter optimization, where, in such cases, it becomes challenging to obtain accurate camera intrinsic parameters. Lastly, based on the degenerate cases, we discuss the design scheme and usage of the calibration object.

3.2.1 Theoretical method for intrinsic parameters

The process of computing the camera intrinsic parameters is called calibration. The problem can be described as: given n calibration points $P_i (i = 0, 1, 2, \dots, n)$ in world coordinate system with corresponding pixel coordinates labelled p_i , determining the intrinsic matrix K of the pinhole camera. For the same camera, the projection relationship between P_i and p_i is as follows:

$$sp_i = K [R | T] P_i \quad \text{with } K = \begin{bmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

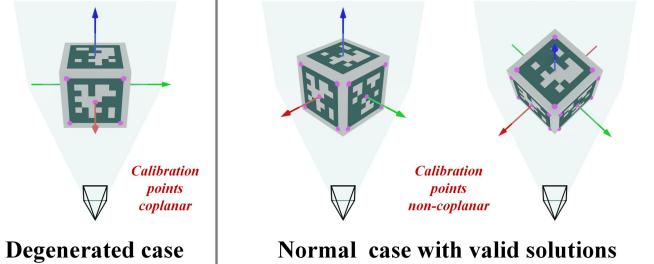


Fig. 3: Degenerate case in intrinsic parameters regression. When a single Apriltag is present in the calibration image, obtaining a valid solution is not feasible. However, at least two Apriltags ensures the acquisition of camera intrinsic parameters.

where s is an arbitrary scale factor, $[R | T]$ are the extrinsic parameters, and K is defined as the camera intrinsic matrix. (u_0, v_0) in K are the coordinates of the principal point, α and β are called scale factors in image u and v axes respectively, and c describes the skewness of the two image axes. We define a homography H with size 3×4 , which is also called camera matrix, and we have:

$$sp_i = HP_i \quad \text{with } H = K [R | T] \quad (2)$$

with the condition of $n \geq 6$ and $\|H\|_F = 1$, where F is Frobenius norm, H can be solved by SVD decomposition. When H is obtained, K and R can be solved by the first three columns of H . Considering that K and R are upper triangular matrix and orthogonal matrix respectively, we can adopt RQ decomposition to get the solutions. Generally, we define the camera coordinate system as same as it defined in OpenCV or Colmap, and the diagonal elements of K are all positive, then we have:

$$K = KD \quad \text{with } D = \begin{bmatrix} f(K_{11}) & 0 & 0 \\ 0 & f(K_{12}) & 0 \\ 0 & 0 & f(K_{13}) \end{bmatrix} \quad (3)$$

where $f()$ is the sign function $sign()$. Besides, we set the condition of $\|H\|_F = 1$, which is not the real scale of K . The final camera intrinsic matrix is defined as $K = K/K_{33}$.

3.2.2 Training Parameters and Degenerated Cases

The core of the network architecture is to reproduce the computing process of Eq.(1). The equation satisfies both the projection mapping and the inverse projection mapping. Projection entails mapping image coordinates p_i to 3D coordinates P_i , while the inverse projection maps 3D coordinates P_i back to image coordinates p_i . We choose the inverse projection mapping as the reproduction computation flow because it doesn't require depth information for precise 2D coordinate retrieval, and the loss function can be implemented in pixel coordinate system.

Assume that the multi-camera system contains m cameras with unknown parameters, and $K_j (j = 0, 1, 2, \dots, m)$ denotes the intrinsic matrix of each camera. We decompose K_j into the initialization K_{j0} and the adjustable weights ΔK_{j0} , and

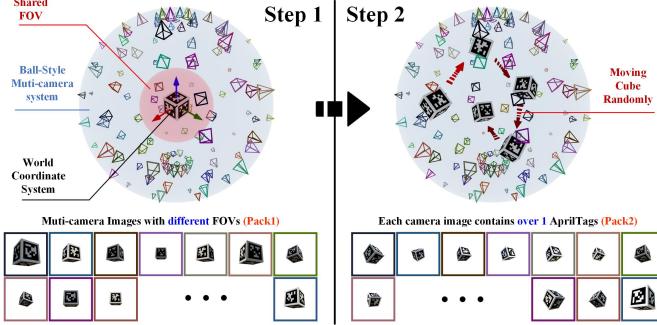


Fig. 4: Details of calibration data Acquisition

also assume the axes of the image coordinate system are all perpendicular to each other, which means $c = 0$. The matrix:

$$K_j = \begin{bmatrix} f_{xj} \cdot \Delta f_{xj} & 0 & u_{0j} \cdot \Delta u_{0j} \\ 0 & f_{yj} \cdot \Delta f_{yj} & v_{0j} \cdot \Delta v_{0j} \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

We define the adjustable weights of extrinsic parameters in the $se(3)$ space and then convert back to the $SE(3)$ space. This approach is taken because representing extrinsic matrix requires only 6 variables in $se(3)$, while it demands 12 variables to regression of R and T in the $SE(3)$ space directly. Let $[\alpha_{j0}, \alpha_{j1}, \alpha_{j2}, \alpha_{j3}, \alpha_{j4}, \alpha_{j5}]$ ($j = 0, 1, 2, \dots, m$) be 6 adjustable weights in the $se(3)$ and the corresponding matrix in the $SE(3)$ is denotes as $[R_j | T_j]$ with size 3×4 . For each camera in the system, we have:

$$pd_p_{ij} = K_j [R_j | T_j] P_{ij} \quad (5)$$

where P_{ij} is the 3D point used as input and generated by the calibration object, pd_p_{ij} is the predicted pixel coordinate. We define the loss function for the intrinsic parameters regression stage as:

$$loss_{intr} = \sum_{i=0, j=0}^{i=n, j=m} \left\| \frac{gt_p_{ij}}{\sqrt{h^2 + w^2}} - \frac{pd_p_{ij}}{\sqrt{h^2 + w^2}} \right\|_2^2 \quad (6)$$

where h, w represent the height and width of the image, and gt_p_{ij} denotes the ground-truth coordinate of p_{ij} , which can be detected in the images including the calibration object.

It should be noted that the above optimization can not get the correct solutions sometimes, because of the degenerated cases. Specifically, from Eq.(2), we have:

$$[u_i \ v_i \ 1]^T = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} [X_i \ Y_i \ Z_i \ 1]^T \quad (7)$$

where h_i has dimensions 4×1 , and we convert Eq.(7) to:

$$\begin{cases} u_i h_3^T P_i - h_1^T P_i = 0 \\ v_i h_3^T P_i - h_2^T P_i = 0 \\ \Rightarrow \begin{cases} -P_i^T h_1 + 0 + u_i P_i^T h_3 = 0 \\ 0 - P_i^T h_2 + v_i P_i^T h_3 = 0 \end{cases} \end{cases} \quad (8)$$

Eq.(8) is a homogeneous system of linear equation consisting of h_i . As mentioned earlier, solving this equation requires the condition $n \geq 6$, then we can convert Eq.(8) to:

$$Ah = 0, \text{ with } A = \begin{bmatrix} -P_0^T & 0 & u_i P_0^T \\ 0 & -P_0^T & v_i P_0^T \\ \vdots & \vdots & \vdots \\ -P_n^T & 0 & u_i P_n^T \\ 0 & -P_n^T & v_i P_n^T \end{bmatrix} \quad (9)$$

Besides, in Eq.(9), we also have $h = [h_1 \ h_2 \ h_3]^T$ and $\|h\| = 1$. When all the calibration points lie in the same plane, for example, when all the feature points are in the plane at $Z = 0$, we have A in Eq.(9):

$$A = \begin{bmatrix} -X_0 & 0 & \dots \\ -Y_0 & 0 & \dots \\ 0 & 0 & \dots \\ 1 & 0 & \dots \\ 0 & -X_0 & \dots \\ 0 & -Y_0 & \dots \\ 0 & 0 & \dots \\ 0 & 1 & \dots \\ -u_0 X_0 & v_0 X_0 & \dots \\ -u_0 Y_0 & v_0 Y_0 & \dots \\ 0 & 0 & \dots \\ u_0 & v_0 & \dots \end{bmatrix}^T \quad (10)$$

In Eq.(10), columns 3, 7 and 11 are always equal to 0, which implies that there is no unique solution for A and also for H . This prevents available solution for the camera intrinsic matrix K_{cam}^i . The specific degenerated case and normal case are shown in Fig.3.

3.2.3 Calibration cube design and instruction

Compared to single-camera calibration, multi-camera calibration has the following limitations: First, the cameras of multi-camera systems are fixed in space and cannot move freely, like camera array and camera ball. Second, it is time-consuming for multi-camera systems to capture images with the calibrator in the ideal position. Hence, it is necessary for each camera to use as few images as possible to complete the calibration. Third, multi-camera systems need to be calibrated frequently because of the changing of camera position and lens. Besides, calibrator should be designed for ease of manufacturing while maintaining high machining precision to ensure the accuracy of calibration. As mentioned in section 3.2.2, degenerated cases lead the invalid solutions when all calibration points are coplanar. This also require that the calibrator must have the capability of providing non-coplanar three-dimensional calibration points.

To meet these requirements, we designed the calibrator as a cube. Each side of it is spray-painted with a different Apriltag label from the 36H11 family. Apriltag can provide five calibration points per plane, including four corners and a central point. Detecting more than two Apriltag patterns in each image can provide at least 10 points. These points also fulfill the condition of being non-coplanar, which avoids the degenerated cases mentioned in Section 3.2.2. Besides, the cube is easy to machine, with low cost but high machining

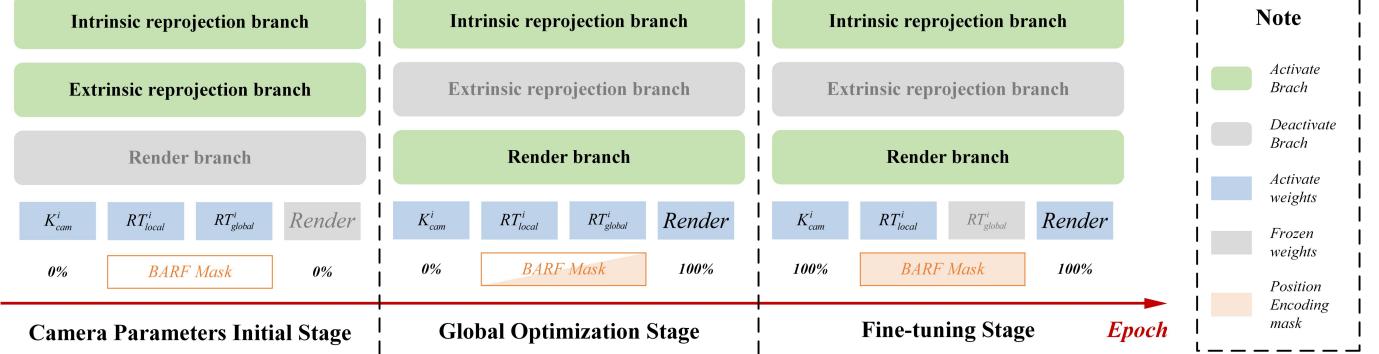


Fig. 5: Training sequential diagram: Our training process consists of three stages: 1) Camera Parameter Initialization Stage: The goal in this stage is to obtain coarse intrinsic and extrinsic parameters. Rendering training is not conducted in this stage. 2) Global Optimization Stage: Due to the coupling issue during joint optimization, we retain the Intrinsic Reprojection branch as a decoupling constraint. We also employ the progressive alignment method proposed in BARF to achieve extrinsic parameters with higher accuracy than those obtained in the first stage. 3) Fine-tuning Stage: In this stage, continuing to optimize extrinsic parameters without the progressive alignment constraint can lead to divergence. We freeze the extrinsic parameters and only adjust the intrinsic parameters.

accuracy. As for calibration points detection, Apriltag supports an open source algorithm, which is stable and easy to deploy.

Fig.4 illustrates the procedure for acquiring calibration data. Firstly, the cube is captured by all cameras in the shared field of view, with the center of the cube defined as the origin of the world coordinate system. The X-axis towards to number 1, the Y-axis towards to number 2 and the Z-axis towards to number 4. We define the set of images captured during this step is *Pack1*. Secondly, the cube is moved randomly to ensure that each camera captures an image containing at least two Apriltags, and the set of images captured in this phase is defined as *Pack2*. Until now, we have finished the calibration data collection. *Pack1* is utilized for training the coarse extrinsic parameters, while *Pack2* is used for calibrating coarse intrinsic parameters K^i_{cam} of each camera.

3.3 Multi-Camera Extrinsic Parameters Regression

Once the coarse intrinsic parameters are obtained, the regression for extrinsic parameters is converted into a PnP problem which is Perspective-n-point. In the computation flow outlined in Section 3.2, the form of the loss function in Eq.(6) is extremely similar to that of the Bundle Adjustment (BA) method, which is widely employed to address Perspective-n-point (PnP) problems. This means that we can use the reprojection process to optimize the extrinsic parameters again.

3.3.1 World coordinate system

In Section 3.2, we collected two sets of auxiliary data, named *Pack1* and *Pack2*, and the world coordinate system definitions in these two sets are entirely different. It's important to note that the world coordinate system definition of the multi-camera acquisition system corresponds to that of *Pack1*. When using the *Pack2* data to obtain camera intrinsic parameters, the matching extrinsic parameters can also be obtained. The origin of the world coordinate system for these extrinsic parameters remains defined at the center of the cube calibrator within their images. However, due to the random movement of the cube, the world coordinate systems

for each image are not consistent with others. As a result, these extrinsic parameters are irrelevant to the multi-camera acquisition system and carry no practical significance. As illustrated in Fig.2, the RT^i_{local} from Intrinsic reprojection loss branch is not reused in the subsequent optimization.

In order to obtain the accurate position of each camera in multi-camera acquisition system, the PnP optimization needs to adopt the Apritag data in *Pack1* to regenerate extrinsic parameters.

3.3.2 PnP with bundle adjustment

The Perspective-n-point (PnP) problem is defined as follows: given a set of 3D points in world coordinate system, and knowing their corresponding coordinates in pixel coordinate system, solving the camera extrinsic parameters with the condition of intrinsic parameters provided. As a typical method to solve PnP, Bundle Adjustment (BA) can be described as to optimize camera poses and scene points by minimizing the difference between the projected 3D scene points in the images and their corresponding 2D image feature points. The loss function in BA is defined as:

$$loss_{BA} = \frac{1}{2} \sum_{i=0, j=0}^{i=n, j=m} \left\| \frac{gt_p_{ij}}{\sqrt{h^2 + w^2}} - \frac{pd_p_{ij}}{\sqrt{h^2 + w^2}} \right\|_2^2 \quad (11)$$

Eq.(11) and Eq.(6) differ only in their coefficients, and the computation flow can be reused to regress the extrinsic parameters. The optimized coarse extrinsic parameters are denoted as RT^i_{global} .

During the training process, the two branches optimize simultaneously, sharing camera intrinsic parameters. Since their computation flows are almost the same, their convergence rates are quite similar, intrinsic and extrinsic reprojection losses can be directly added.

3.4 Neural Radiance Fields and Global Optimization

We unfold this section by explaining why NeRF is necessary for the global end-to-end network architecture first. Then

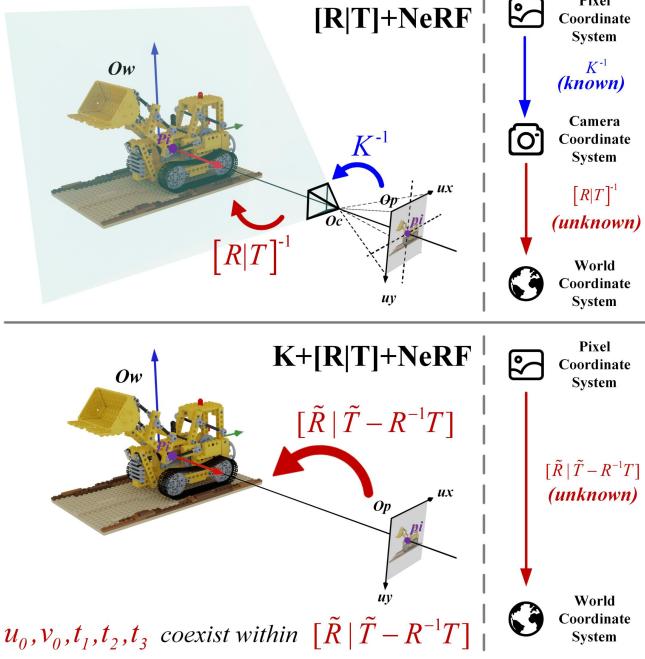


Fig. 6: Coupling issue between intrinsic and extrinsic parameters. 1) The first row illustrates the joint optimization for poses and NeRF. In methods such as BARF or L2G-NeRF, where the intrinsic parameters are known, mitigating the issue of camera parameters coupling. 2) The second row showcases the joint optimization of all parameters, as outlined in Eq.13. This procedure effectively represents a global transformation involving scaling, rotation, and translation. u_0 and v_0 from intrinsic parameters, along with t_1, t_2 and t_3 from extrinsic parameters, coexist within $\tilde{T} - R^{-1}T$ and cannot be disentangled.

we discuss the coupling problem during joint training of intrinsic and extrinsic parameters. Following this, we validate the proposed method within the 2D image alignment space, which is similar to previous works such as BARF and L2G. We also explain the differences between jointly optimizing all camera parameters and individually optimizing extrinsic parameters in this part. Finally, we present the details of how we design the entire network and transition it from the 2D image alignment space to the 3D space.

3.4.1 Necessity of neural radiance fields

Neural radiance fields (NeRF) employ Multi-layer perceptron (MLP) to represent 3D scenes. The network processes 5D information, including position and direction, and generates pixel color and volume density from a given viewpoint.

An important precondition of NeRF is the accurate correspondence between image pixels and 3D space rays. On the one hand, we can use the camera intrinsic and extrinsic parameters to establish the above connection for 3D scene representation. On the other hand, we can also use this relationship as a constraint to optimize the camera parameters. Previous works, like SC-NeRF and BARF, have demonstrated the capability for jointly optimizing NeRF alongside camera parameters, which implies that it is feasible to implement an

end-to-end joint optimization.

Our goal is to achieve a 3D representation of an object within the shared FOV of a multi-camera acquisition system, aligning with the inherent capabilities of NeRF. Additionally, it is an efficient and convenient method that can simultaneously acquire parameters for each camera during the scene representation training stage, with the ability to store and reuse them. This process seamlessly integrates calibration with scene reconstruction, offering significant benefits for multi-camera systems, particularly those experiencing frequent variations in intrinsic and extrinsic parameters.

In summary, the introduction of NeRF makes the end-to-end network structure design feasible, enabling simultaneous camera parameters regression and scene representation. These make NeRF necessary both in design and target.

3.4.2 Coupling in camera parameters

As mentioned in Eq.(1), our goal is to obtain the values of K and $[R|T]$. In the NeRF method, rays are defined by their direction vectors rather than specific spatial point coordinates, which implies that we can disregard the scale variable s . Given pixel coordinates as inputs and $c = 0$ provided in Eq.(4), Eq.(1) can be rewritten as follows:

$$P_i = [R|T]^{-1}K^{-1}p_i$$

$$\text{with } K = \begin{bmatrix} 1/f_x & 0 & -u_0/f_x \\ 0 & 1/f_y & -v_0/f_y \\ 0 & 0 & 1 \end{bmatrix} \quad (12)$$

Specifically, Eq.(12) can be modified as:

$$\begin{aligned} P_i &= R^{-1}K^{-1}p_i - R^{-1}T \\ \Rightarrow P_i &= \tilde{R}p_i + (\tilde{T} - R^{-1}T) \end{aligned} \quad (13)$$

where \tilde{R} and \tilde{T} are as follows:

$$\tilde{R} = \begin{bmatrix} r_{11}/f_x & r_{21}/f_y & r_{31} \\ r_{12}/f_x & r_{22}/f_y & r_{32} \\ r_{13}/f_x & r_{23}/f_y & r_{33} \end{bmatrix}, \quad (14)$$

$$\tilde{T} = \begin{bmatrix} -u_0r_{11}/f_x & -v_0r_{21}/f_y & r_{31} \\ -u_0r_{12}/f_x & -v_0r_{22}/f_y & r_{32} \\ -u_0r_{13}/f_x & -v_0r_{23}/f_y & r_{33} \end{bmatrix}$$

we also have:

$$\begin{aligned} \tilde{T} - R^{-1}T &= \begin{bmatrix} \tilde{t}_1r_{11} + \tilde{t}_2r_{21} + \tilde{t}_3r_{31} \\ \tilde{t}_1r_{12} + \tilde{t}_2r_{22} + \tilde{t}_3r_{32} \\ \tilde{t}_1r_{13} + \tilde{t}_2r_{23} + \tilde{t}_3r_{33} \end{bmatrix} \\ \text{with } \tilde{t}_1 &= -u_0/f_x - t_1 \\ \tilde{t}_2 &= -v_0/f_y - t_2 \\ \tilde{t}_3 &= 1 - t_3 \end{aligned} \quad (15)$$

In Eq.(14) and Eq.(15), r_{ij} represents element of the rotation matrix R in 3D space, and t_i represents the element of the translation matrix T in 3D space. From Eq.(15), we can observe that u_0, v_0 and t_1, t_2, t_3 are strongly coupled. While we can obtain result for $\tilde{T} - R^{-1}T$ through optimization, distinguishing between u_0, v_0 and T remains challenging.

A straightforward solution to address the coupling problem in joint optimization is by imposing additional constraints on the coupled variables. We introduce a constraint for the camera intrinsic parameters to separate intrinsic and extrinsic parameters from each other. In the collected calibration data, *Pack2* provides comprehensive constraints on the camera intrinsic parameters, while *Pack1* only constrains the world coordinate system with extrinsic parameters (the images in *Pack1* cannot guarantee at least two Apriltags captured in each image). Moreover, the extrinsic parameters RT_{local}^i generated by *Pack2* are independent of the multi-camera acquisition system's extrinsic parameters RT_{global}^i , which eliminates the impact of RT_{global}^i while constraining the intrinsic parameters. The described procedure is shown in Fig.6.

In summary, as mentioned in Section 3.1, we require the intrinsic reprojection loss as a constraint to achieve the independent regression of coupled variables u_0, v_0 and t_1, t_2, t_3 .

3.4.3 Neural Image Alignment (2D)

As mentioned in BARF and L2G-NeRF, 2D neural image alignment problem can be described as: let x be the 2D pixel coordinates, we aim to optimize a 2D neural field parameterized as the weights of a multilayer perceptron (MLP) f_{mlp} :

$$RGB(x) = f_{mlp}(Tx; w) \quad (16)$$

while also solving for geometric transformation parameters as $T = [R|T]$, and $[R|T]$ denotes the rigid rotation and translation in 2D space.

Compared to the previously introduced 2D image alignment problem, we have increased its complexity even further, resulting in the updated problem formulation:

$$RGB(x) = f_{mlp}(TLx; w) \quad (17)$$

Where L represents a linear transformation, simulating the computation flow of projection from pixel coordinates to camera coordinates. The form of L is consistent with that of the intrinsic matrix K and is defined as follows:

$$L = \begin{bmatrix} a_1 & 0 & b_1 \\ 0 & a_2 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \quad (18)$$

Based on Eq.(17) and Eq.(18), the regression for the intrinsic parameter K can be equivalently represented as the regression for the linear transformation L . It's important to note that in the experiments conducted by BARF and L2G, the term T in Eq.(16) is defined as a homography matrix. However, since the transformation requirements for 3D space in the NeRF series methods involve only rotation and translation, we redefine T as $[R|T]$ instead. This adjustment reduces the number of regression parameters for each cropped patch from 8 ($T \in SL(3)$) to 3 ($T \in SE(2)$). Nonetheless, the inclusion of the linear transformation introduces an additional 4 parameters, resulting in a total of 7 regression parameters.

3.4.4 NeRF with Bundle Adjustment

Lin et al. [38] first proposed Bundle-Adjusting Neural Radiance Fields (BARF), which addresses the problem of achieving NeRF training without precise camera extrinsic parameters. We believe one of the key advantages of BARF is its independence from requiring extra information. It only needs simple adjustments to the coarse-to-fine NeRF architecture to achieve the joint optimization for camera pose and NeRF. The authors validated the proposed progressive alignment method on the image alignment task in 2D space, and extended its application to the 3D NeRF space.

We attempted to extend this method again by incorporating the regression of camera intrinsic parameters into the BARF framework. The training sequence map of our method is illustrated in Fig.5. During the camera parameters initialization stage, coarse camera parameters are obtained using the calibration data *Pack1* and *Pack2*. In the global optimization stage, we employ the intrinsic reprojection loss branch to impose constraints on the principal point coordinates of the camera. This strategy effectively addresses the parameter coupling problem outlined in Section 3.4.2. Simultaneously, we leverage the progressive alignment method from BARF to optimize the extrinsic parameters of the multi-camera acquisition system. In the fine-tuning stage, we freeze the extrinsic parameters and only refine the intrinsic parameters to obtain the final results.

4 EXPERIMENTS

The goal of the experiments is to answer the following questions:

- 1) Does NeRF still work effectively when each image corresponds to different intrinsic parameters? In comparison to the globally unique camera assumption in the original NeRF, what challenges do the multiple intrinsic parameters present to the NeRF series methods?
- 2) Does the Intrinsic Reprojection Loss branch proposed in Section 3.2 successfully yield accurate camera intrinsic parameters? Section 3.2 highlights that if the calibration points lie on the same plane, a degenerate case could occur, resulting in inaccurate estimation. Does this challenge still remain within the proposed network architecture?
- 3) Is the extrinsic regression computation introduced in Section 3.3 necessary? How does the performance of models such as BARF and L2G-NeRF, which can directly regress extrinsic parameters with accurate intrinsic parameters provided?
- 4) Compared to previous works, how does the proposed method perform in the 2D neural image alignment problem?
- 5) Does the approach introduced in Section 3.4 achieve global optimization of the network? Having obtained the intrinsic and extrinsic parameters of different cameras as initial values in Sections 3.2 and 3.3, how does the performance compare when directly optimizing NeRF using these initial parameters, without implementing global optimization?

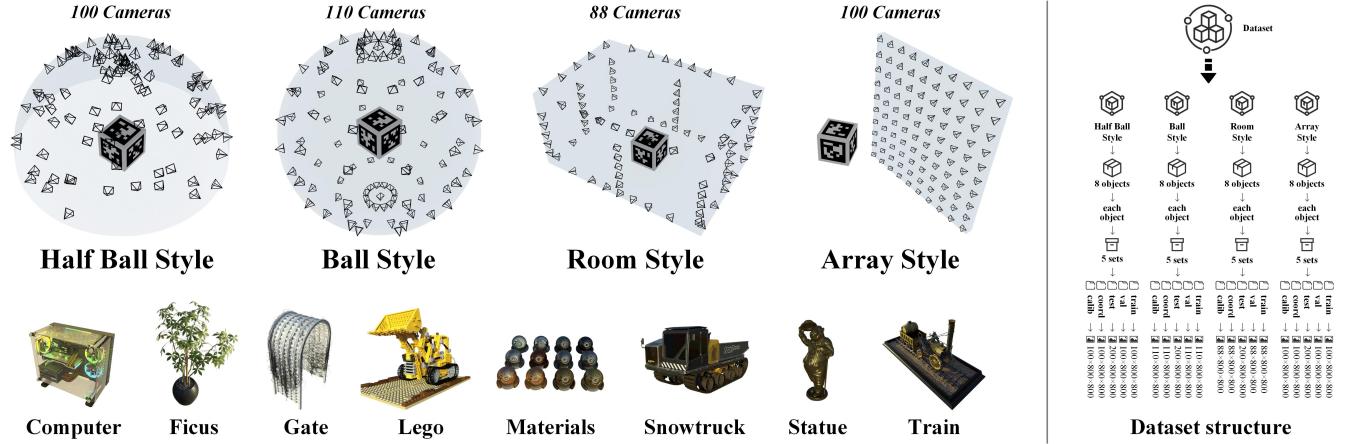


Fig. 7: Scene styles and dataset structure. The left side shows the camera distribution for the different multi-camera systems and the eight objects of interest in our dataset. The right side illustrates the composition of our dataset. Within each style, there are eight objects correspond to five sets of data, including two sets of calibration (*Pack1* and *Pack2*), as well as the training, validation, and test sets.

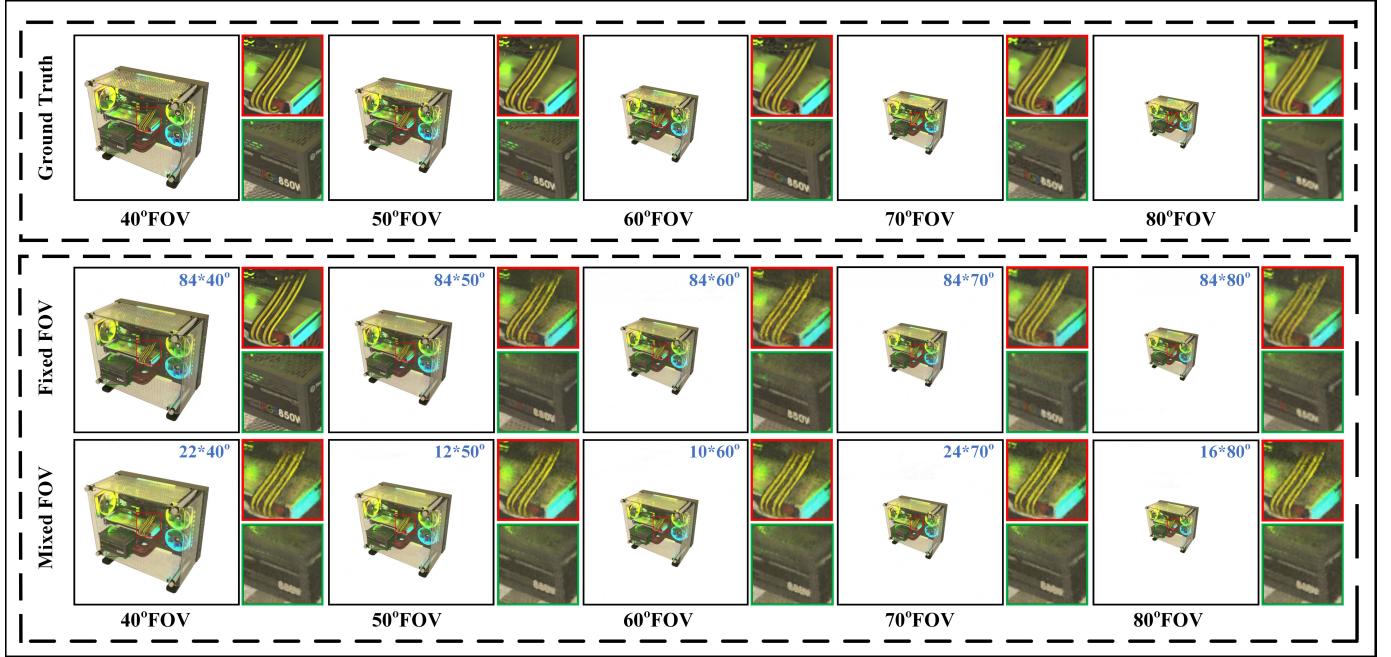


Fig. 8: Comparison of rendering performance in NeRF with mixed intrinsic dataset and single intrinsic dataset.

4.1 Dataset and Hardware Platform

We designed four datasets based on commonly used multi-camera image acquisition systems. The camera distribution in these datasets includes halfball style (as employed by the original NeRF), ball style, room style, and array style. The datasets contain a total of 32 groups, with 8 different objects for each dataset style. The different sampling styles involve varying numbers of cameras, leading to different numbers of training images. For each style, the number of validation images is equal to the number of training images. The details of the datasets are illustrated on the right side of Fig. 7.

It's worth noting that due to the images being collected

from cameras with different intrinsic parameters, determining which camera to use for generating the test data is challenging. To showcase the performance of the proposed method across various camera intrinsic parameters, we generate the 200 test images in each group by continuously varying the intrinsic parameters along the camera's motion trajectory. Detailed sampling procedures and parameter variations have been provided on the project website.

The datasets are generated using Blender 3.3.3, and the intrinsic parameters for each camera within each style are different. For example, in the HalfBall Style acquisition system, there are a total of 100 cameras with varying intrinsic parameters. The training data is collected using these cameras

according to their depicted positions. Similarly, the validation data is also collected using these cameras, but each camera’s position differs from that of the training set. The resolution of each image is 800×800 . All of our experiments were conducted on an NVIDIA RTX 3090 with 24GB, and PyTorch version is 1.12.1.

TABLE 2: Quantitative results of *Exp.1*.

	40°	50°	60°	70°	80°
PSNR↑	Fixed FOV	25.440	27.872	29.896	32.935
	Mixed FOV	24.169	27.333	29.949	32.138
	Distance	1.271	0.539	0.053	0.797
SSIM↑	Fixed FOV	0.858	0.913	0.942	0.969
	Mixed FOV	0.819	0.899	0.939	0.963
	Distance	0.039	0.014	0.003	0.006
LPIPS↓	Fixed FOV	0.108	0.056	0.040	0.017
	Mixed FOV	0.147	0.069	0.037	0.026
	Distance	0.039	0.013	0.003	0.009

4.2 Exp.1 Unique-Camera VS Multi-Cameras

In this experiment, our goal is to explore whether NeRF has the capability to utilize multiple intrinsic parameters simultaneously, which represents a foundational prerequisite for our method. We select cameras with FOVs of 40, 50, 60, 70, and 80 degrees to generate a mixed dataset. NeRF is then trained on this dataset with provided ground-truth intrinsic and extrinsic parameters. Simultaneously, we also conduct training of NeRF with each FOV data independently and compare the rendering performance with NeRF trained on the mixed dataset. Hence, this experiment contains a total of 6 datasets: 1 mixed dataset and 5 independent datasets. Each dataset has 84 images, and the extrinsic parameters are the same totally across all datasets.

The results are presented in Fig.8 where the top row shows ground-truth images captured at varying FOVs, indicated below each image. The second row displays results produced by the NeRF model trained on each fixed FOV dataset, and the third row exhibits results from the NeRF model trained on a mixed dataset. The numbers in the top-right corner of each result represent the number of images in the dataset with the current FOV. In the mixed dataset, there are 22, 12, 10, 24, and 16 images for FOVs of 40, 50, 60, 70, and 80 degrees, respectively.

The comparison results can be observed in Table 2. The *Distance* item indicates the absolute difference between their respective indicators. As evident from both Table 2 and Fig.8, the models trained with different datasets exhibit nearly equivalent performance (in each column of both Fig.8 and Table 2). This observation has shown the capability of NeRF that can effectively process multi-camera images, when accurate camera intrinsic and extrinsic parameters are provided.

In summary, with accurate camera parameters provided, NeRF can work efficiently when each image corresponds to different intrinsic parameters. We believe that the challenge posed by multiple intrinsic parameters to NeRF is how to obtain accurate intrinsic parameters for each individual camera. The function of intrinsic and extrinsic parameters is to

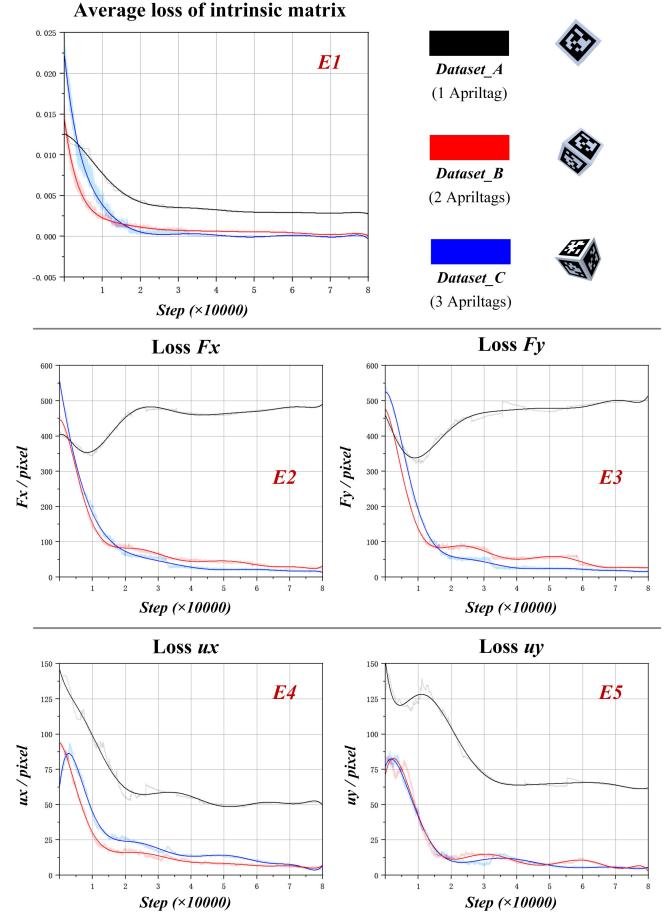


Fig. 9: Comparison results of regressing intrinsic parameters with different number of Apriltag in calibration data. With at least two Apriltags included in the calibration data, we can achieve the expected results. However, when including only one Apriltag, degenerate case is showed up, leading to poor results. Specifically, the black curve in *E1* fails to converge to zero, while the blue and red curves both converge near zero.

establish a mapping between image pixels and spatial rays. The accuracy and correctness of this mapping serve as vital prerequisites for NeRF, irrespective of whether the camera intrinsics are the same or equal.

4.3 Exp.2 Intrinsic Parameters Regression

In this experiment, our goal is to verify the effectiveness of the proposed intrinsic parameters regression method. We conduct experiments using AprilTag images collected by 70 cameras with different FOVs. The dataset consists of three groups: the images in the first group contain only one AprilTag in each image, denoted as *Dataset_A*. This group ensures that the calibration points lie in the same plane. The images in the second group contain two AprilTags, denoted as *Dataset_B*. According to the conclusion in Section 3.2, theoretically, the results trained by this dataset can successfully obtain the intrinsic parameters. The images in the third group contain three AprilTags, denoted as *Dataset_C*. We believe that the results trained with this group will have better accuracy. Here,

TABLE 3: Quantitative results of *Exp.2*.

	Train Step	10000	20000	40000	60000	80000
<i>Loss_Fx</i>	Dataset A	370.06	458.82	460.30	474.59	487.35
	Dataset B	153.53	88.98	53.87	32.47	26.64
	Dataset C	198.08	69.21	26.73	19.41	17.41
<i>Loss_Fy</i>	Dataset A	343.69	429.86	462.75	470.78	511.29
	Dataset B	144.23	92.43	55.34	32.47	26.64
	Dataset C	197.32	57.01	25.65	21.34	18.91
<i>Loss_Ux</i>	Dataset A	99.42	62.91	58.71	51.43	51.61
	Dataset B	23.17	15.83	12.85	9.27	7.75
	Dataset C	30.73	19.57	11.37	9.77	7.10
<i>Loss_Uy</i>	Dataset A	129.51	105.26	64.49	68.22	61.94
	Dataset B	44.52	14.98	13.056	12.48	9.73
	Dataset C	43.17	13.65	14.21	8.31	7.19

Fx and *Fy* are the focal lengths of the camera in the horizontal and vertical directions, while *ux* and *uy* are the coordinates of the principal point.

The regression results are presented in Fig. 9 and Table 3. Average loss of intrinsic parameters refers to the average absolute error between predicted intrinsic parameters and ground truth intrinsic parameters. In this scenario, the images contain only 1 AprilTag (*Dataset_A*) and the intrinsic parameters cannot be obtained. This case exhibits similar performance to the degenerate case, indicating the persistence of degeneracy in the proposed method. Specifically, as illustrated by the black curves in *E2* and *E3*, *Fx* and *Fy* fail to converge, and the losses in *E4* and *E5* exhibit significant deviations from 0. Moreover, Table 3 reveals that the accuracy of the regression results is significantly inferior to the other two cases.

When the images we used contain more than two AprilTags (*Dataset_B*, *Dataset_C*), the regression can be guaranteed to converge. It can be observed from the quantified data in the table that the parameters trained using the dataset containing three AprilTags achieve higher accuracy, as demonstrated in *Fx*, *Fy*, *ux*, and *uy*. In comparison, the parameters trained with the dataset containing 2 AprilTags converge more rapidly.

In summary, camera intrinsic parameters can be obtained when the Intrinsic Reprojection Loss branch is trained with dataset containing at least two Apriltags. This demonstrates the effectiveness of the proposed method. If the provided calibration points lie on the same plane, the degenerated case will still occur, resulting in the inability to obtain camera intrinsic parameters. The degenerated case is reproducible in our method.

4.4 Exp.3 Extrinsic Parameters Regression

In this experiment, we aim to verify the necessity of extrinsic parameters regression. While BARF and L2G-NeRF address the joint optimization of NeRF and camera poses, these methods are limited by the initial extrinsic parameters. Specifically, both of the above methods add perturbations to the ground truth extrinsic parameters, leading to camera poses drift, which is then attempted to be rectified during the joint optimization. However, the initial pose of each camera in practical applications still requires huge preparation work, especially for the multi-camera acquisition systems with a

TABLE 4: Comparison of poses regression performance between BARF and L2G-NeRF with the initialization poses obtained from calibration cube.

Style		Inherited Value	L2G-NeRF	BARF
Ball	<i>Loss_R</i>	0.0392	0.0054	0.0050
	<i>Loss_T</i>	0.1355	0.0463	0.0509
HalfBall	<i>Loss_R</i>	0.0601	0.0183	0.0114
	<i>Loss_T</i>	0.1891	0.0224	0.0321
Room	<i>Loss_R</i>	0.0473	0.0048	0.0027
	<i>Loss_T</i>	0.1844	0.0366	0.0284
Array	<i>Loss_R</i>	0.0561	0.0727	0.0915
	<i>Loss_T</i>	0.2154	0.0149	0.0287

large number of cameras. Therefore, in this experiment we randomly provide the initial poses of all cameras and attempt to perform poses regression on four styles of scenes using the above mentioned two methods. This is done to assess the performance of the previous methods under the condition where no potential poses information is provided.

The results are presented in Fig. 10. The first row illustrates the randomly initialized camera poses of the two methods across the four scenarios, while the second row depicts the stabilized camera poses after training. When the camera poses are initialized randomly, both methods struggle to acquire accurate camera extrinsic parameters. As our proposed method employs the progressive alignment technique introduced by BARF in the joint optimization, it becomes evident that an effective initialization of the camera pose is essential. Further details regarding the convergence process are provided in our supplementary project webpage.

It is worth noting that the camera initialization pose obtained from the calibration cube can satisfy both L2G-NeRF and BARF. The reasons for adopting the BARF approach are as follows: Firstly, as outlined in the original BARF, it makes only minor modifications to the classic Coarse-to-Fine architecture of NeRF, which makes it easily extensible. Secondly, in comparison to L2G-NeRF, these two methods exhibit similar performance under the aforementioned extrinsic initialization conditions. The comparisons are presented in Table 4. Inherited Value refers to the initial poses obtained by calibration cube, *Loss_R* and *Loss_T* denote the losses associated with the ground truth rotation and translation data. These losses are defined as follows:

$$\begin{cases} \text{Loss}_R = \sum_{i=0}^n \text{mean}(\|\hat{R}_i - R_i\|_2^2)/n \\ \text{Loss}_T = \sum_{i=0}^n \text{mean}(\|\hat{T}_i - T_i\|_2^2)/n \end{cases} \quad (19)$$

\hat{R}_i and R_i represent the ground truth and predicted rotation matrices respectively, while \hat{T}_i and T_i represent the ground truth and predicted translation matrices. $\text{mean}()$ represents the mean function. n represents the total number of poses, which is equal to the number of images in the dataset. It can be observed that when using the initial pose provided by the calibration cube, there is almost no difference in performance between these methods across the four different styles of scenes.

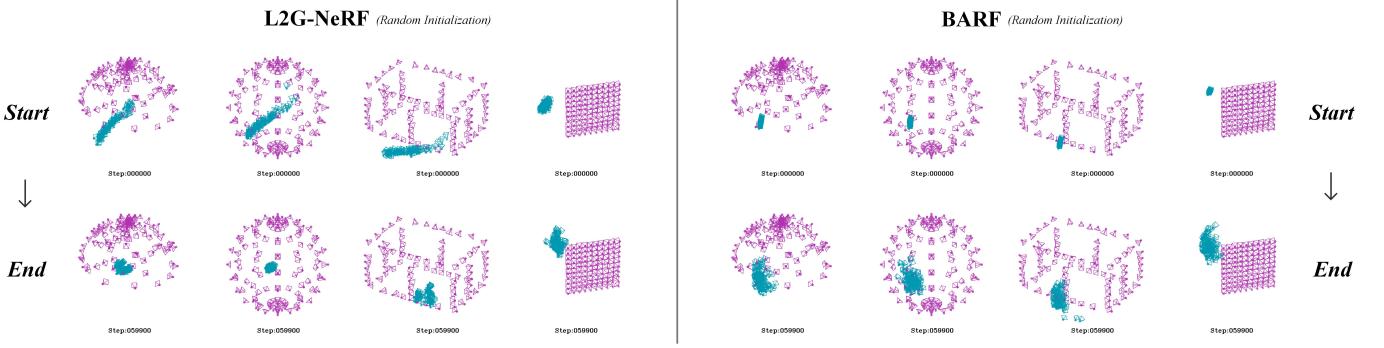


Fig. 10: Comparison of poses regression performance between BARF and L2G-NeRF with the random initialization poses.

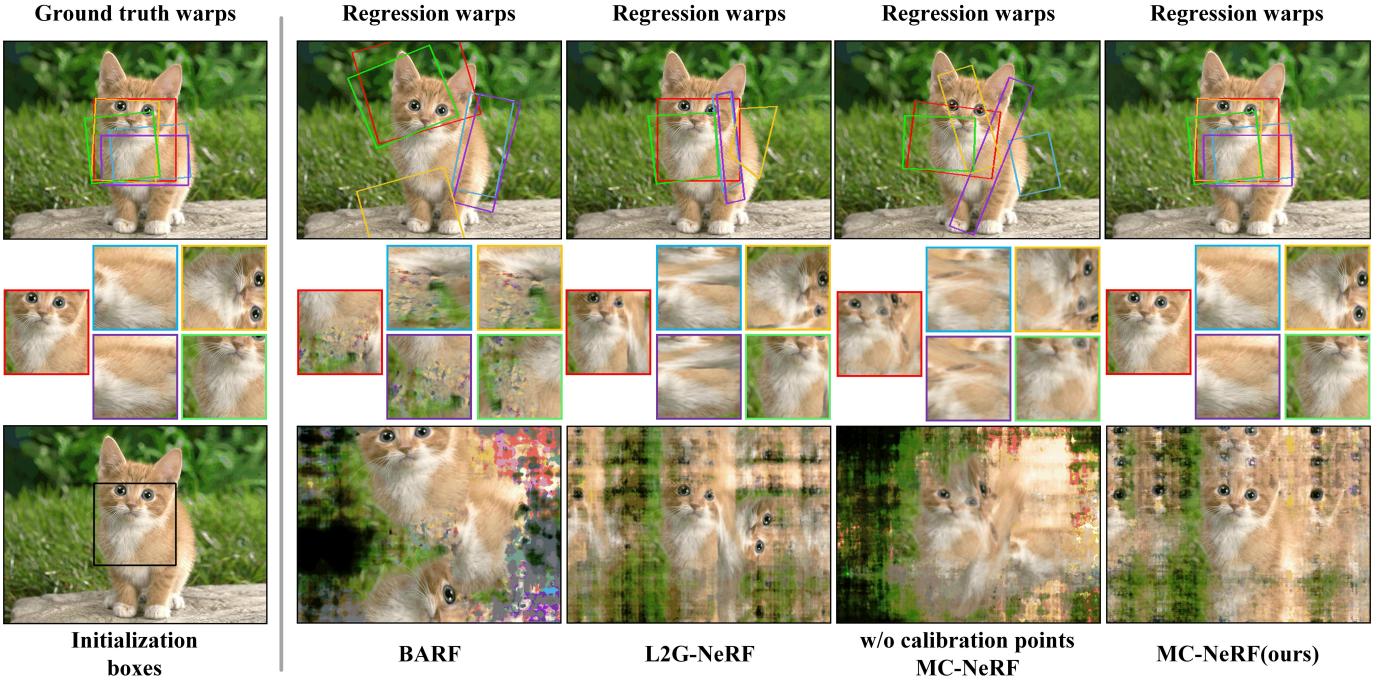


Fig. 11: Qualitative results of the planar image alignment experiment. We visualize the optimized warps (top row), the patch reconstructions in corresponding colors (middle row), and recovered image representation from f_{mlp} (bottom row). MC-NeRF with calibration points is able to recover accurate alignment and high-fidelity image reconstruction

In summary, extrinsic parameter regression is necessary. Randomly initializing external parameters cannot guarantee obtaining accurate results. By using calibration cube data to initialize extrinsic parameters, we can subsequently achieve precise camera poses using the previously mentioned methods.

TABLE 5: Quantitative results of neural image alignment experiment. MC-NeRF optimizes for high-quality alignment and patch reconstruction.

	Corner Error \downarrow (pixels)	Patch PSNR \uparrow	Patch SSIM \uparrow	Patch LPIPS \downarrow
BARF	85.6157	14.2270	0.5149	0.6641
L2G-NeRF	55.3545	20.5440	0.7884	0.3364
MC-NeRF(w/o)	55.3096	15.9662	0.5923	0.6268
MC-NeRF(ours)	0.4084	40.8309	0.9828	0.0318

4.5 Exp.4 Planar Image Alignment (2D)

In this experiment, our goal is to compare the performance of the proposed method with previous works in the task of 2D image alignment. Furthermore, we also explore the performance of the method when calibration points are not provided. We choose the same representative image "cat" as BARF used from ImageNet [51]. We select five patches sampled from the original image, apply linear transformations and rigid perturbations, and optimize Eq.(17) to find the transformation L and T for each patch. Simultaneously, we train the neural network f_{mlp} to represent the entire image.

The initial poses and ground truth transformations for each patch are depicted on the left side of Fig. 11. It's important to note that in BARF, MC-NeRF (W/O), and MC-NeRF, L represents linear transformation, and T represents rigid rotation and translation. However, in L2G-NeRF, we combine L and T



Fig. 12: Rendering results of Fix-Step NeRF and MC-NeRF. We visualize the ground truth images (top row), the rendered RGB images (middle row), and the predicted depth images (bottom row). Thanks to the global optimization, MC-NeRF exhibits improved rendering details and sharper object boundaries.

and directly replace them with a homography transformation. The reason for this lies in the design of L2G-NeRF, where finding differentiable parameter estimation solvers for custom computational flows is challenging. In theory, the homography transformation solver provided by L2G-NeRF itself can simultaneously handle scale, rotation, and translation, and we use the homography transformation to replace L and T .

As mentioned in Section 3, joint optimization of intrinsic parameters can be equivalent to adding an additional linear transformation L . To decouple this linear transformation from rotation and translation (as discussed in Section 3.4.2), we selected six pairs of randomly chosen calibration points as constraints for each patch. This constraint is analogous to the calibration points obtained by calibration cube in 3D dimensional space. In 2D space, the loss function for MC-NeRF is *Eq.(20)*. In all test models, a 6-layer fully connected network with 256 neurons each is respectively defined. The training epochs are set to 50,000 to ensure convergence for all methods. The specific regression process is demonstrated on our project webpage.

In summary, the inclusion of linear transformations leads to both BARF and L2G-NeRF being unable to produce accurate results. Even in cases where calibration points are not provided, proposed method also faces challenges in achieving precision results. Notably, in the case of MC-NeRF without calibration points, the loss function comprises only the photometric loss and does not include the reprojection term.

4.6 Exp.5 Fix Step VS Global optimization

In this experiment, our goal is to validate the effectiveness of the global optimization framework and compare its performance against the Fixed Step NeRF rendering (Intrinsic and extrinsic parameters regression + NeRF, without other

optimization). Fixed Step rendering simulates the typical steps involved in NeRF-based reconstruction using a multi-camera acquisition systems: Initially, camera intrinsic and extrinsic parameters are determined using calibration information. Subsequently, NeRF is trained using these fixed parameters to represent the 3D scene. Throughout the NeRF training phase, the camera parameters remain constant.

In theory, global optimization is expected to provide more accurate intrinsic parameters, extrinsic parameters, and rendering results compared to the Fix Step method. In this experiment, both approaches utilized identical camera initialization parameters and employed the Adam optimizer with a learning rate of 0.01 for 20 epochs. This ensured that, upon entering the Global Optimization Stage in Fig.5, the camera parameters of the global optimization method closely resembled those of the Fix Step method.

The experimental results are presented in Fig.12, Table 6, and Table 7. We believe the global optimization has better performance compared to the Fix Step method. Regarding image rendering quality, in Fig.12, we randomly selected three scenes rendered in four different styles. The global optimization method stands out by producing sharper object boundaries and capturing finer object details. Table 6 shows the quantitative performance for 32 scenes. Firstly, MC-NeRF shows a significant performance improvement in Array Style, where it achieves the best scores across all three evaluation metrics. Secondly, we observed that MC-NeRF consistently outperforms in all LPIPS metrics. However, when it comes to PSNR and SSIM, except for the Array scene, MC-NeRF often has lower scores. We explored the reasons, as depicted in Fig.13.

Although MC-NeRF achieves better rendering details, it demonstrates larger errors in object boundaries alignment.

TABLE 6: Quantitative rendering results of Fix-Step NeRF and MC-NeRF on all scenes.

	Scene	PSNR↑				SSIM↑				LPIPS↓			
		Ball	HalfBall	Room	Array	Ball	HalfBall	Room	Array	Ball	HalfBall	Room	Array
Fix-Step	Computer	25.1685	25.3800	20.6838	16.6674	0.9230	0.9088	0.8869	0.8838	0.0882	0.1020	0.1419	0.1571
	Ficus	26.9503	25.9101	25.3156	17.2587	0.9539	0.9427	0.9313	0.8893	0.0431	0.0542	0.0762	0.1590
	Gate	28.9521	28.9134	29.8277	18.1242	0.9416	0.9348	0.9341	0.8853	0.0542	0.0633	0.0782	0.1515
	Lego	25.5330	25.1424	24.9465	15.6066	0.9219	0.8822	0.8742	0.8672	0.0998	0.1106	0.1214	0.1897
	Materials	26.7998	25.8398	25.8738	16.3879	0.9443	0.9504	0.9462	0.8953	0.0666	0.0543	0.0634	0.1618
	Snowtruck	25.1714	25.1153	23.7291	14.7770	0.9166	0.9114	0.8989	0.8770	0.0959	0.1093	0.1287	0.1728
	Statue	30.5160	31.5379	24.4189	16.0766	0.9710	0.9694	0.9380	0.8938	0.0383	0.0420	0.0954	0.1571
	Train	24.3452	24.1038	22.1991	15.0438	0.9240	0.8779	0.8578	0.8768	0.0876	0.1301	0.1466	0.1807
	Mean	26.6795	26.4928	24.6243	16.2428	0.9370	0.9222	0.9084	0.8836	0.0717	0.0832	0.1065	0.1662
MC-NeRF	Computer	24.2311	23.0962	23.6603	24.6946	0.9217	0.8876	0.9008	0.9395	0.0794	0.0933	0.1004	0.0444
	Ficus	24.9479	24.7662	26.5778	24.2929	0.9488	0.9070	0.9386	0.9504	0.0336	0.0519	0.0449	0.0387
	Gate	27.9102	26.0876	29.3081	26.7734	0.9333	0.9131	0.9250	0.9356	0.0519	0.0525	0.0602	0.0439
	Lego	24.1226	23.4818	23.3858	24.3502	0.9227	0.8682	0.8837	0.9629	0.0620	0.0901	0.0874	0.0343
	Materials	26.8742	26.6170	27.1323	25.0106	0.9484	0.9488	0.9529	0.9601	0.0321	0.0423	0.0294	0.0397
	Snowtruck	24.8742	24.0678	23.4415	23.3363	0.9211	0.8882	0.8937	0.9375	0.0433	0.0882	0.0874	0.0451
	Statue	28.6882	29.2767	28.7953	24.6619	0.9731	0.9670	0.9612	0.9651	0.0303	0.0301	0.0398	0.0281
	Train	23.3830	23.5271	22.2576	23.4073	0.9311	0.8773	0.8589	0.9395	0.0454	0.0991	0.0835	0.0507
	Mean	25.6289	25.1151	25.5698	24.5659	0.9375	0.9072	0.9144	0.9488	0.0473	0.0684	0.0666	0.0406

TABLE 7: Quantitative camera parameters regression results of Fix-Step NeRF and MC-NeRF on all scenes.

	Scene	Loss_K↓				Loss_R↓				Loss_T↓			
		Ball	HalfBall	Room	Array	Ball	HalfBall	Room	Array	Ball	HalfBall	Room	Array
Fix-Step	Computer	12.0723	11.2606	17.6997	28.6428	0.0131	0.0124	0.0309	0.0157	0.0806	0.0830	0.1530	0.2616
	Ficus	10.2547	11.0173	16.9588	28.9794	0.0147	0.0128	0.0276	0.0148	0.0802	0.0925	0.1507	0.2714
	Gate	10.0650	11.0287	17.4918	26.3233	0.0134	0.0138	0.0344	0.0157	0.0781	0.0797	0.1604	0.2433
	Lego	10.3323	11.0146	17.0964	24.9460	0.0135	0.0163	0.0315	0.0156	0.0787	0.0830	0.1577	0.2505
	Materials	11.3069	10.8165	15.3205	27.4074	0.0157	0.0148	0.0300	0.0193	0.1005	0.0860	0.1458	0.2704
	Snowtruck	11.7263	10.9522	13.9390	29.3079	0.0131	0.0136	0.0270	0.0141	0.0811	0.0727	0.1106	0.2619
	Statue	10.3607	10.0933	17.1346	27.5365	0.0123	0.0104	0.0352	0.0149	0.0775	0.0774	0.1480	0.2569
	Train	11.5821	10.2598	17.1833	27.8020	0.0136	0.0139	0.0334	0.0164	0.0862	0.0762	0.1630	0.2595
	Mean	10.9625	10.8054	16.6030	27.6182	0.0137	0.0135	0.0313	0.0158	0.0829	0.0813	0.1487	0.2594
MC-NeRF	Computer	1.2758	5.4864	6.3094	5.0589	0.0033	0.0101	0.0090	0.0057	0.0231	0.0398	0.0696	0.0549
	Ficus	1.2524	1.3501	3.0402	5.6844	0.0031	0.0042	0.0036	0.0081	0.0209	0.0234	0.0306	0.0734
	Gate	11.9388	1.2863	5.4361	4.6863	0.0089	0.0043	0.0066	0.0064	0.0426	0.0607	0.0511	0.0524
	Lego	1.1857	1.5403	5.6895	5.1950	0.0024	0.0032	0.0073	0.0073	0.0243	0.0243	0.0546	0.0661
	Materials	4.0039	1.5266	9.3145	4.1782	0.0043	0.0035	0.0151	0.0088	0.0398	0.0177	0.1096	0.0709
	Snowtruck	1.5929	1.2136	8.6714	4.2423	0.0036	0.0032	0.0209	0.0058	0.0267	0.0156	0.0659	0.0619
	Statue	1.2508	1.2926	5.1592	4.2392	0.0040	0.0032	0.0073	0.0082	0.0184	0.0148	0.0539	0.0654
	Train	7.4066	1.3244	10.9638	4.4496	0.0151	0.0030	0.0101	0.0069	0.0940	0.0139	0.0549	0.0588
	Mean	3.7384	1.8775	6.8230	4.7167	0.0056	0.0043	0.0100	0.0072	0.0362	0.0263	0.0613	0.0630

We believe that the reason behind this phenomenon is the absence of enforced ray alignment constraints for rendering in MC-NeRF. Specifically, the camera parameters of Fix-Step NeRF are constant and remain stable during the rendering stage, which maintains a fixed ray distribution, and all fitting computations contained within NeRF’s MLPs. Based on the rendering results, it can be observed that to compensate for the ray distribution error resulting from inaccuracies in camera parameters, the rendered objects exhibit a multi-layer shadow structure. In Fig.13, the boundaries of the computer show

noticeable shadows, while those of MC-NeRF appear much clearer and sharper. This multi-layer shadow structure has more advantages for pixel-to-pixel calculation methods like PSNR. If the clear-edged object generated by MC-NeRF is not aligned with the ground truth, the PSNR scores can be easily affected by the background, resulting in lower scores. Although MC-NeRF has some drawbacks in terms of boundary alignment, we prioritize high-quality rendering. Additionally, based on the rendering images, we believe that this alignment error is tolerable and unlikely to be readily noticeable.

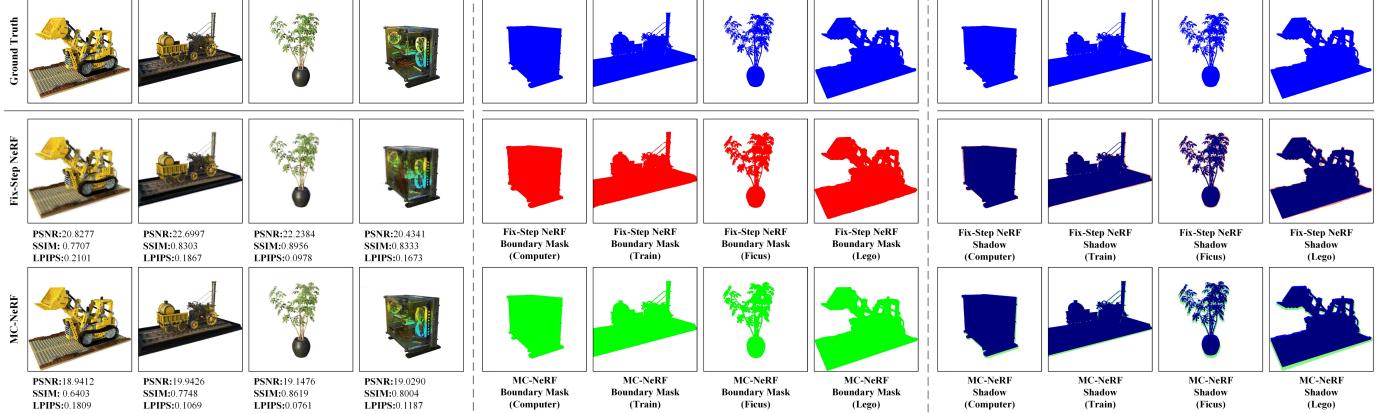


Fig. 13: Boundary alignment results for rendering objects. On the left, the rendering results and evaluation metrics for Fix-Step NeRF and MC-NeRF are displayed. In the middle section, the object boundary masks are presented. On the right, the comparison results between the object boundaries and the ground truth are shown. It can be observed that Fix-Step NeRF exhibits better object alignment performance compared to MC-NeRF. The more pixels contained within the mask, the more favorable it is for evaluation methods like PSNR, which involve pixel-to-pixel calculations.

Concerning the regression of camera parameters, as presented in Table 7, global optimization yields more accurate results with a significant improvement. We believe that during the global optimization, NeRF takes the lead as a supervisor for the regression. To achieve higher-quality rendering results, NeRF enforces the alignment of global rays, and this alignment constraint outperforms traditional camera calibration using calibration points. Additionally, the results also demonstrate that proposed decoupling constraint effectively resolves the coupling issue between intrinsic and extrinsic parameters in joint regression, allowing for the separation of parameters.

In summary, proposed method is proven to be effective. With global optimization, both the rendering images and the accuracy of regression parameters surpass those achieved using initial intrinsic and extrinsic parameters. Moreover, in this experiment, we noted that achieving higher-quality rendering results does not necessarily correlate with higher PSNR scores. We believe that this phenomenon is unique to the joint regression of intrinsic and extrinsic parameters. Because even when the regression results for both intrinsic and extrinsic parameters are highly accurate, their values are often challenging to align perfectly with ground truth. This leads to slight shifts of objects in the rendered image, which have a notable impact when calculating pixel-wise differences for PSNR. However, the LPIPS can still objectively reflect rendering performance even in such cases, as demonstrated by the conclusions in Table 6, where the results of global optimization consistently outperformed the Fix Step method.

5 CONCLUSIONS

In this paper, we present Multi-Camera Neural Radiance Fields (MC-NeRF) for Multi-Camera Image Acquisition Systems, a method that enables the joint optimization of both multi-camera intrinsic and extrinsic parameters during the training of neural radiance fields. We analyzed the challenges

of joint optimization, including intrinsic parameters degenerated case and strong coupling between these parameters. Based on this analysis, we design the network architecture with training sequence, calibration objects, and a calibration data acquisition strategy. The proposed method primarily tackles the substantial workload associated with acquiring multi-camera intrinsic and extrinsic parameters when employing multi-camera acquisition systems for 3D scene representation. Code and models will be made available to the research community to facilitate reproducible research.

Although MC-NeRF allows for the regression of camera intrinsic and extrinsic parameters with NeRF, it still necessitates the acquisition of additional calibration data. We believe that the data collection scheme we propose can significantly reduce time requirements and enhance accuracy, but there is potential for optimization in the human-involved actions.

ACKNOWLEDGMENTS

This work is partly supported by National Natural Science Foundation of China (Grant No. 62233002, 61973034, U1913203 and CJSP Q2018229). The authors would like to thank Tianji Jiang, Xi Xu, Jiadong Tang, Zhaoxiang Liang, Xihan Wang, Dianyi Yang, and all other members of the ININ Lab of the Beijing Institute of Technology for their contribution to this work.

References

- [1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” *Commun. ACM*, vol. 65, no. 1, p. 99–106, dec 2021.
- [2] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines,” *ACM Trans. Graph.*, vol. 38, no. 4, jul 2019.
- [3] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, “Neural sparse voxel fields,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 15651–15663.
- [4] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, “Mip-nerf 360: Unbounded anti-aliased neural radiance fields,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 5460–5469.
- [5] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.
- [6] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, “Large-scale data for multiple-view stereopsis,” *International Journal of Computer Vision*, vol. 120, pp. 153–168, 2016.
- [7] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, “Mvsnet: Depth inference for unstructured multi-view stereo,” in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 785–801.
- [8] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA, USA: MIT Press, 1993.
- [9] R. Tsai, “A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.
- [10] Z. Zhang, “Flexible camera calibration by viewing a plane from unknown orientations,” in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, 1999, pp. 666–673 vol.1.
- [11] H. Lee, E. Shechtman, J. Wang, and S. Lee, “Automatic upright adjustment of photographs with robust camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 833–844, 2014.
- [12] M. Zhai, S. Workman, and N. Jacobs, “Detecting vanishing points using global image context in a non-manhattan world,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5657–5665.
- [13] Q. Chen, H. Wu, and T. Wada, “Camera calibration with two arbitrary coplanar circles,” in *Computer Vision - ECCV 2004*, T. Pajdla and J. Matas, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 521–532.
- [14] O. Bogdan, V. Eckstein, F. Rameau, and J.-C. Bazin, “Deepcalib: A deep learning approach for automatic intrinsic calibration of wide field-of-view cameras,” in *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production*, ser. CVMP ’18. New York, NY, USA: Association for Computing Machinery, 2018.
- [15] Y. Hold-Geoffroy, D. Piché-Meunier, K. Sunkavalli, J.-C. Bazin, F. Rameau, and J.-F. Lalonde, “A perceptual measure for deep single image camera and lens calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10603–10614, 2023.
- [16] M. Landy and J. A. Movshon, *Shape from X: Psychophysics and Computation*. MIT Press, 1991, pp. 305–330.
- [17] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, “Multi-view stereo for community photo collections,” in *2007 IEEE 11th International Conference on Computer Vision*, Oct 2007, pp. 1–8.
- [18] C. M. Parameshwara, G. Hari, C. Fermüller, N. J. Sanket, and Y. Aloimonos, “Diffposenet: Direct differentiable camera pose estimation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 6845–6854.
- [19] S. Lee and Y.-K. Moon, “Camera pose estimation using voxel-based features for autonomous vehicle localization tracking,” in *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, 2022, pp. 185–188.
- [20] W. Ye, X. Lan, S. Chen, Y. Ming, X. Yu, H. Bao, Z. Cui, and G. Zhang, “Pvo: Panoptic visual odometry,” 2023.
- [21] C.-C. Chiu, H.-K. Yang, H.-W. Chen, Y.-W. Chen, and C.-Y. Lee, “Vitvo: Vision transformer based visual odometry with attention supervision,” in *2023 18th International Conference on Machine Vision and Applications (MVA)*, 2023, pp. 1–5.
- [22] A. Kendall, M. Grimes, and R. Cipolla, “Posenet: A convolutional network for real-time 6-dof camera relocalization,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [23] Y.-Y. Jau, R. Zhu, H. Su, and M. Chandraker, “Deep keypoint-based camera pose estimation with geometric constraints,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4950–4957.
- [24] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, “Inerf: Inverting neural radiance fields for pose estimation,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1323–1330.
- [25] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, “Nerf++: Analyzing and improving neural radiance fields,” 2020.
- [26] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, “Nerf in the wild: Neural radiance fields for unconstrained photo collections,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 7206–7215.
- [27] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, “Graf: Generative radiance fields for 3d-aware image synthesis,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 20154–20166.
- [28] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, “Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5835–5844.
- [29] H. Jin, I. Liu, P. Xu, X. Zhang, S. Han, S. Bi, X. Zhou, Z. Xu, and H. Su, “Tensoir: Tensorial inverse rendering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 165–174.
- [30] T. Hu, S. Liu, Y. Chen, T. Shen, and J. Jia, “Efficientnerf efficient neural radiance fields,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 12902–12911.
- [31] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, “Zip-nerf: Anti-aliased grid-based neural radiance fields,” 2023.
- [32] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.
- [33] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, “Tensorf: Tensorial radiance fields,” in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 333–350.
- [34] C. Wu, “Towards linear-time incremental structure from motion,” in *2013 International Conference on 3D Vision - 3DV 2013*, 2013, pp. 127–134.
- [35] F. Radenovic, J. L. Schönberger, D. Ji, J.-M. Frahm, O. Chum, and J. Matas, “From dusk till dawn: Modeling in the dark,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5488–5496.
- [36] Z. Li, S. Niklaus, N. Snavely, and O. Wang, “Neural scene flow fields for space-time view synthesis of dynamic scenes,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6494–6504.
- [37] S.-F. Chng, S. Ramasinghe, J. Sherrah, and S. Lucey, “Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation,” in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 264–280.
- [38] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, “Barf: Bundle-adjusting neural radiance fields,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5721–5731.
- [39] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, “Nerf-: Neural radiance fields without known camera parameters,” 2022.
- [40] Y. Xia, H. Tang, R. Timofte, and L. V. Gool, “Sinerf: Sinusoidal neural radiance fields for joint pose estimation and scene reconstruction,” 2022.
- [41] Y. Jeong, S. Ahn, C. Choy, A. Anandkumar, M. Cho, and J. Park, “Self-calibrating neural radiance fields,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5826–5834.
- [42] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, “Nerfies: Deformable neural radiance fields,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5845–5854.
- [43] A. Hertz, O. Perel, R. Giryes, O. Sorkine-hornung, and D. Cohen-or, “Sape: Spatially-adaptive progressive encoding for neural optimization,”

- in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 8820–8832.
- [44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.
- [45] W. Bian, Z. Wang, K. Li, and J.-W. Bian, “Nope-nerf: Optimising neural radiance field with no pose prior,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 4160–4169.
- [46] P. Truong, M.-J. Rakotosaona, F. Manhardt, and F. Tombari, “Sparf: Neural radiance fields from sparse and noisy poses,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 4190–4200.
- [47] Y. Chen and G. H. Lee, “Dbarf: Deep bundle-adjusting generalizable neural radiance fields,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 24–34.
- [48] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, “Ibrnet: Learning multi-view image-based rendering,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 4688–4697.
- [49] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, “pixelnerf: Neural radiance fields from one or few images,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 4576–4585.
- [50] Q. Meng, A. Chen, H. Luo, M. Wu, H. Su, L. Xu, X. He, and J. Yu, “Gnerf: Gan-based neural radiance field without posed camera,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 6331–6341.
- [51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.