FBINeRF: Feature-Based Integrated Recurrent Network for Pinhole and Fisheye Neural Radiance Fields

Yifan Wu[1], Tianyi Cheng[1], Peixu Xin[1], and Janusz Konrad[1]

Department of Electrical and Computer Engineering, Boston University
{wuyifan1, chengty, brucexin, jkonrad}@bu.edu

Abstract. Previous studies aiming to optimize and bundle-adjust camera poses using Neural Radiance Fields (NeRFs), such as BARF and DBARF, have demonstrated impressive capabilities in 3D scene reconstruction. However, these approaches have been designed for pinhole-camera pose optimization and do not perform well under radial image distortions such as those in fisheye cameras. Furthermore, inaccurate depth initialization in DBARF results in erroneous geometric information affecting the overall convergence and quality of results. In this paper, we propose adaptive GRUs with a flexible bundle-adjustment method adapted to radial distortions and incorporate feature-based recurrent neural networks to generate continuous novel views from fisheye datasets. Other NeRF methods for fisheye images, such as SCNeRF and OMNI-NeRF, use projected ray distance loss for distorted pose refinement, causing severe artifacts, long rendering time, and are difficult to use in downstream tasks, where the dense voxel representation generated by a NeRF method needs to be converted into a mesh representation. We also address depth initialization issues by adding MiDaS-based depth priors for pinhole images. Through extensive experiments, we demonstrate the generalization capacity of FBINeRF and show high-fidelity results for both pinhole-camera and fisheye-camera NeRFs.

1   Introduction

Neural Radiance Field (NeRF) [28] is a state-of-the-art method that reconstructs a 3D scene from 2D images and can create highly-realistic and detailed renderings from new perspectives. NeRF employs a neural network encoder to minimize the difference between rendered views and actual dataset images. Additionally, it uses a multilayer perceptron (MLP) to create a scene representation by evaluating a 5D implicit function that estimates scene density and radiance.

NeRF relies on relatively precise initial camera pose information. However, in practice the acquisition of camera poses is expensive. In previous studies, camera poses have been usually acquired by means of Structure-from-Motion (SfM) methods [23,33,52]. In SfM and its variants, camera poses are optimized under a keypoint-metric reprojection error in a process referred to as bundle adjustment [38]. SfM may encounter difficulties, *e.g.,* in scenes lacking texture

Fig. 1: FBINeRF and DBARF reconstructions for natural and synthetic fisheye data. Two examples from FisheyeNeRF dataset [19] demonstrate that even a very small radial distortions degrade performance of DBARF but not of FBINeRF. **Top:** Chairs image - DBARF reconstruction (no lens-distortion model) shows severe distortions on some chairs and window blinds. **Middle:** Rock image - obvious distortions especially at periphery. **Bottom**: Synthetic image [12] - cannot be reconstructed by DBARF.

or containing self-similar elements, and can often be time-consuming, requiring days to complete large-scale scenes. As a result, a significant challenge for NeRF is its strong dependence on precise camera poses in order to produce high-quality renderings. Recent studies, such as BARF [22], have proposed to jointly train camera-pose recovery with NeRF. Inspired by non-smoothness optimization in high-dimensional functions, BARF employs a coarse-to-fine approach initially eliminating high-frequency components and then gradually reactivating them once low-frequency components have reached stability. The resulting smoother objective function eases training although the optimizer may still get trapped in a local optimum. During training, camera poses are fine-tuned using a photo-metric loss rather than relying on keypoint-metric cost as in SfM. Despite these innovations, BARF still relies on pre-computed camera poses.

More recently, generalizable NeRF methods [7, 8, 41] have drawn much attention. Most of them, unlike BARF, directly extract features for optimization instead of positional encoding. DBARF [8] jointly bundle-adjusts camera poses by leveraging a smooth feature cost map (from Feature Pyramid Network) as an

implicit cost function in a self-supervised manner. The contribution of DBARF is that it combines camera poses with generalizable NeRF methods and thus can be applied to NeRF not optimized per-scene, without accurate initial camera poses. However, DBARF has limited ability to tackle geometrically-distorted images (Fig. 1). NeRF methods to-date involve camera-pose optimization **using global optimization with geometric constraints leading to slow convergence of distortion parameters and severe artifacts**. Moreover, they **initialize depth maps** *via* **a mapping from features to depth values** [8, 16], thus potentially resulting in reduced accuracy and difficulties with convergence when dealing with distorted camera poses, eventually reducing their generalization ability.

We propose *FBINeRF* to overcome the issues mentioned above. We use IBR-Net as a baseline for NeRF rendering and a deep recurrent network for relative pose refinement. Pinhole and fisheye NeRF rendering are separated. In the fisheye camera pipeline (our main contribution), we utilize adaptive GRUs to process image features extracted from DenseNet [18] and update the pose with incremental hidden states. Then, a flexible bundle adjustment block is adapted to radial image distortions when jointly training relative camera poses with feature cost map in the recurrent neural network. This enables faster convergence during training compared to traditional fisheye NeRF methods and generates dense and continuous synthetic views from fisheye images (see videos in supplementary materials). For the pinhole camera pipeline, inspired by ZoeDepth [5], we add MiDaS-like depth priors (self-supervised) and metric-bins module (supervised) to initialize depth for better convergence and adaptation to unseen data. *FBINeRF* switches to different models according to the input image data via designated types.

In summary, we present a novel integrated generalizable network that generates dense voxel fisheye novel synthetic views that can be converted into mesh representations for downstream tasks in Blender, Unity, or Unreal Engine 5 where simulations, such as for autonomous vehicles, may require meshes or point clouds. We also introduce depth priors to fix DBARF's convergence issues caused by random depth initialization in the pinhole camera model and enhance the pinhole model's generalization ability to produce depth priors from unseen datasets.

2   Related Work

Depth Estimation Conventional approaches [11,17,24] use stereo vision, where depth is computed from feature correspondence between multiple images taken from different viewpoints. These methods are limited in their ability to handle occlusions and textureless regions. In recent years, deep learning has revolutionized depth estimation by employing large-scale datasets. For example, in [15] a self-supervised learning framework was introduced that exploits left-right consistency in stereo pairs to train a depth-estimation model without requiring ground-truth depth annotations. In [51], a deep-learning approach was proposed that combines RGB and sparse depth data to produce dense high-resolution

depth maps. The most recent depth-estimation research can be grouped into two categories: relative depth estimation [20, 26, 32] and metric depth estimation [4, 13, 21]. Inspired by results reported in [5], we leverage both types of methods in FBINeRF.

Neural Radiance Fields NeRF methods developed to-date have used diverse approaches to improve rendering quality or speed. Some methods focus on mitigating aliasing effects [1–3]. Other methods aim to reduce both training and rendering time [6, 29, 35, 40, 47]. Pixel-NeRF [48] is a benchmark method that first generalized NeRFs to unseen views. It utilizes projection and interpolation to construct a feature volume containing features extracted from images. A subsequent MLP network renders RGB and density values. IBRNet [41] combines image features on a per-point basis using a weighting scheme inspired by PointNet [31]. It extends that approach by introducing a ray transformer [39] to estimate density and an additional MLP to predict pixel colors. MVSNeRF [7] builds a 3D feature-cost volume based on N depth hypotheses, and employs a 3D CNN to reconstruct a neural-voxel volume followed by an MLP to predict pixel colors and volume density. Another category of NeRF methods applies *pose refinement* [9, 10, 25, 44, 46, 49]. NeRF- - [44] jointly optimizes a NeRF network and camera-pose embeddings, and achieves comparable accuracy to NeRF methods that require posed images. SiNeRF [45] avois the sub-optimality issue in NeRF- - by using a mixed-region sampling strategy. VMRF [49] allows to learn NeRF without known camera poses. An unbalanced optimal transport is incorporated to model the relative transformation between the actual and rendered image; the predicted relative poses are utilized to update camera poses, thus enabling a more refined NeRF training.

NeRF with Distortion Rectification OMNI-NERF [34] extends the capabilities of traditional NeRF models to omnidirectional imagery, allowing it to handle scenes captured with omnidirectional cameras. SCNeRF [19] addresses the challenge of fisheye-lens distortions up to mild-distortion levels. However, the method attempts to capture geometric information from basic NeRF model to assure photometric consistency, which limits the potential for better optimization and rectification of distortions. In contrast, our method, FBINeRF, takes advantage of generalized NeRF models for pose refinement to help tackle distortions.

3 Methods

3.1 Selection of Neighbor Poses

Existing generalized NeRF methods aggregate features from nearby views by means of the nearest top-k views. For planar keypoint matching in pinhole images, we use FeatureBooster [42], a lightweight network that enhances feature
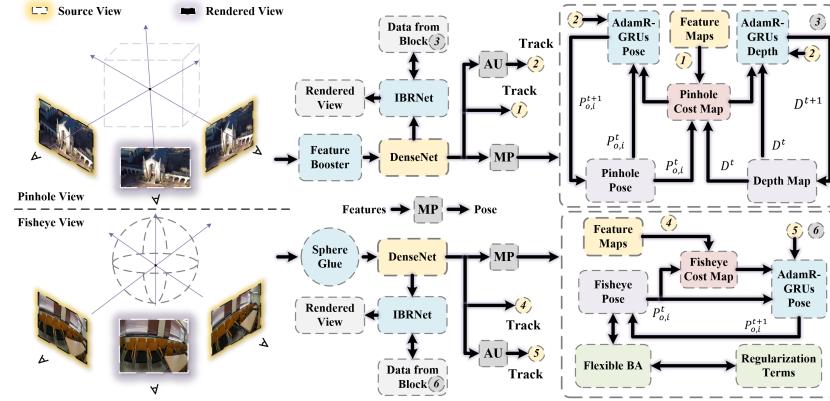
Fig. 2: Network architecture of FBINeRF. 1) The input can be either pinhole or fisheye views. Neighbor views are selected by keypoint matching: FeatureBooster [42] in pinhole views and SphereGlue [14] in fisheye views. 2) Image features are extracted by DenseNet-like [18] backbone and sent to subsequent blocks. 3) Depth priors are utilized for depth map initialization in pinhole model and kept updating; Metric-bins module enhances predicted depth priors through fine-tuning with supervised learning, please refer to [5]. 4) Pose/depth, image features, contextual features, cost map, and context vectors are sent to AdamR-GRU [30] optimizer to update poses and depth values. For fisheye views, depth update is replaced by flexible bundle adjustment with a lens-distorsion model to update camera poses. 5) IBRNet is used as the baseline to predict color and density values, jointly training with both pose-optimization blocks.

descriptors, followed by nearest-neighbor (NN) search. Traditional spherical keypoint matching fails to leverage global keypoint neighbor information leading to erroneous or insufficient matches. We use SphereGlue [14] to handle geometric distortions in fisheye images. We use the epipolar constraint [36] as a "sanity check" to validate our network, only for debugging purpose. Ground truth poses are not used during training.

3.2   Pinhole-Pose Refinement

Given a pinhole image $\mathcal{I}_o$ and its N nearby views $\{\mathcal{I}_i\}_{i=1}^{N}$, image features are extracted using DenseNet and are fed into a mapping block (MP in Fig. 2) to obtain the initial relative camera pose $\mathcal{P}_{o,i}^{init}$ from image o to image i, while MiDaS-like depth priors are predicted for depth initialization. Then, image pose/depth, cost map, contextual features and context vectors (*via* attention unit, AU in Fig. 2) are sent to AdamR-GRUs-like [30] optimizer for either pose or depth refinement, in order to minimize photometric loss projected into feature space [37]. As the feature cost map that measures the distance between aligned feature maps, we use the average of multiple-cost values inspired from DBARF [8]:

$$C(x) = \frac{1}{N} \sum_{i=1}^{N} ||\mathcal{F}_i(\Pi(\mathcal{T}_i \circ \Pi^{-1}(x, D(x)))) - \mathcal{F}_o(x)||_2 \qquad (1)$$

where $\Pi(\cdot)$ is a mapping of 3D point to the image plane, and its inverse $\Pi^{-1}(\cdot)$ maps an image pixel x, given its depth $D(x)$, to a point in 3D space. $\mathcal{T}_i$ transforms 3D points between coordinate spaces of two images, e.g., $\mathcal{I}_o$ and $\mathcal{I}_i$. The iterative steps to minimize the feature cost map C are [16]:

$$
\begin{aligned}
D^{t+1} &= D^t + \mathcal{M}(D^t, C^t, \mathcal{F}_o^C, c_o^t) \\
\mathcal{P}_{o,i}^{t+1} &= \mathcal{P}_{o,i}^t + \mathcal{M}(\mathcal{P}_{o,i}^t, C^t, \mathcal{F}_i^C, c_i^t)
\end{aligned}
\tag{2}
$$

where $\mathcal{M}(\cdot)$ is a mapping in AdamR-GRUs [30] optimizer involving either the depth D or relative camera pose $\mathcal{P}_{o,i}$, cost map C, contextual feature $\mathcal{F}^C$, and context vector c. The context vector c is derived from the attention unit (see below) and the contextual feature $\mathcal{F}^C$ is obtained *via* similar feature-extraction backbone with a different mapping.

DenseNet and Feature Matrix DenseNet-121 [18] is used as a backbone to extract features with shared feature weight from different images, denoted $\mathcal{F}_o$ for image $\mathcal{I}_o$ and $\{\mathcal{F}_i\}_{i=1}^N$ for nearby views. The extracted features form a high-dimensional vector, where each element originates from all prior layers in DenseNet. More specifically, the output of the lth layer is: $Q_l = \mathcal{T}_q(q_0, q_1, \ldots, q_{l-1})$, where $\{q_0, q_1, \ldots, q_{l-1}\}$ denotes the concatenation of features extracted from previous layers and $\mathcal{T}_q$ denotes a non-linear transformation. To reduce the computational complexity, transition layers are introduced as a bottleneck [30]. We map the features into a variable-length feature matrix $\mathcal{A}$ that is later reshaped through attention units to match the dimensions of the input to AdamR-GRU. For images $\mathcal{I}_o$, $\mathcal{I}_i$, respective feature matrices are $\mathcal{A}_o \in \mathbb{R}^{H_o \times W_o}$ and $\mathcal{A}_i \in \mathbb{R}^{H_i \times W_i}$. Each extracted feature $\mathcal{F}_i$ contains contextual feature $\mathcal{F}_i^C$ and feature volume $\mathcal{F}_i^V \in \mathbb{R}^{H_i \times W_i \times d}$, containing d feature matrices $\mathcal{A}_i$ for image $\mathcal{I}_i$.

Attention Units and AdamR-GRUs The inputs to attention units consist of each image's feature matrix $\mathcal{A}$, the current attention sum from previous time step $t-1$, and the previous hidden state h^{t-1}. The output is a context vector c^t. We wrap this vector, either the depth value D or pose $\mathcal{P}$, contextual feature $\mathcal{F}^C$ and the feature cost map $\mathcal{C}$ into a tensor denoted m^t at time step t. The current pose $\mathcal{P}_{o,i}^t$ and depth D^t are then predicted from hidden state h^t. Also, in AdamR-GRUs we replace the linear operation with convolution for processing temporal sequences of images and add an adaptive parameter in basic GRUs [16] as follows:

$$
\begin{aligned}
\hat{v}^{t+1}, \hat{h}^{t+1}, \hat{m}^{t+1} &= \text{AR-G}(h^t, v^t, m^t) \\
v^{t+1}, h^{t+1}, m^{t+1} &= \text{AR-G}(c^t, \hat{h}^{t+1}, \hat{v}^{t+1}, \hat{m}^{t+1})
\end{aligned}
\tag{3}
$$

where v^t denotes momentum auxiliary state at time t, and c^t denotes the current context vector from the attention unit. Finally, either the depth maps or camera poses are iteratively updated using the hidden state h^t.

3.3   Fisheye Pose Refinement

We introduced FBINeRF for pinhole-camera pose optimization using a deep recurrent neural network. Now, we extend our method to fisheye-pose refinement.

Distortion Modeling The key to recover the true ray direction of pixel co-ordinates on the image plane regardless of distortion is to model the projection of the fisheye lens onto the image plane. The radial distance r_d from the image center to a distorted image point can be calculated as follows [12]:

$$r_d = 2f \cdot \sin(\frac{\theta}{2}) \tag{4}$$

where θ is the incoming angle measured from the lens axis and f is the focal length. To account for different types of camera lenses, we consider an approximation of the incoming ray direction θ by an odd-order polynomial [36]:

$$\theta = \theta_d + \sum_{i=1}^{3} k_i \cdot \theta_d^{2i+1}, \quad \theta_d = \arctan(\frac{r_d}{f}) \tag{5}$$

where k_1, k_2, k_3 are coefficients modeling fisheye-lens distortions. Given pixel $\mathbf{P}$ with coordinates (u, v) on the image plane, we can express the radial distance r_d as follows:

$$r_d = \sqrt{(u - c_x)^2 + (v - c_y)^2} \tag{6}$$

where (c_x, c_y) are the principle-point coordinates. The actual direction of the ray for pixel $\mathbf{P}$, as a vector, is: $\mathbf{d}_c = [\sin(\theta) \cdot (u - c_x), \sin(\theta) \cdot (v - c_y), \cos(\theta)]^T$. Applying rotation $\mathbf{R}$ and translation $\mathbf{t}$, ray direction $\mathbf{d}_c$ in world coordinates is expressed as follows:

$$\mathbf{d} = [\mathbf{R}^{-1}|\mathbf{t}] \, \mathbf{d}_c \tag{7}$$

where the rotation matrix $\mathbf{R}$ can be parameterized by an axis-angle representation expressed through parameters $\mathbf{\Phi}$, as explained in [34]. The unknown camera parameters to estimate are:

$$\pi_i = (f_x, f_y, c_x, c_y, \mathbf{\Phi}_i, \mathbf{t}_i, k_1, k_2, k_3) \tag{8}$$

where f_x and f_y are focal-length to pixel-size ratios along the x and y axes, respectively, and c_x and c_y are coordinates of the principal point with respect to the top-left corner of the image. Intrinsic parameters are known and the poses (in π_i) are randomly initialized. We jointly optimize camera parameters π_i, and parameters $\mathbf{\Theta}$ of the IBRNet to handle geometric distortions by minimizing the photometric cost in feature space:

$$C(x) = \frac{1}{N} \sum_{i=1}^{N} ||\mathcal{F}_i(\Pi(\mathcal{T}_i \circ \Pi^{-1}(x, \pi_i)) - \mathcal{F}_o(x)||_2 \tag{9}$$

where $\Pi^{-1}(x, \pi_i)$ represents the inverse mapping of pixel coordinates x, given π_i, to 3D space. Subsequently, each 3D point undergoes transformation $\mathcal{T}_i$ based on camera parameters π_i. Combining the warped features generated by the IBRNet baseline with the refined fisheye pose from FBINeRF, we can produce continuous and densely synthesized fisheye views.

Discussion The primary distinction between the pinhole camera pipeline and the fisheye camera pipeline lies in the distortion model (exclusive to fisheye cameras) and depth priors (exclusive to pinhole cameras). Both pipelines utilize the same feature extraction method and similar adaptive GRUs for pose refinement. Additionally, they leverage warped features from IBRNet to render novel views.

Flexible Bundle Adjustment We utilized flexible bundle adjustment in our fisheye pipeline within feature-based deep recurrent neural network to update fisheye poses. However, due to the finer pose updates required by fisheye cameras, failure to control the step size of updates can lead to the model getting trapped in local optima and failing to converge. Thus, we propose a regularization term across time steps to enforce similarity of suitably-scaled rotation and translation parameters (defined in Section 3.3) as follows:

$$\mathcal{R}_S(\pi_i) = \frac{1}{N} \sum_{i=1}^{N} (\sum_{j=1}^{T} ||\mathbf{\Phi}_i^{j+1} - \mathbf{\Phi}_i^j||^2 + \sum_{j=1}^{T} ||\mathbf{t}_i^{j+1} - \mathbf{t}_i^j||^2) \tag{10}$$

where T is the number of time steps for each training batch. Incorporating a pose-update similarity regularization term fosters smooth transitions between adjacent frames, mitigating abrupt variations and ensuring consistency. This regularization enhances optimization stability by constraining pose changes, minimizing oscillations and divergence. Additionally, it aids in mitigating error propagation by safeguarding against significant estimation errors in one frame affecting subsequent frames.

Fisheye Model Final Loss We combine the photometric cost mapped into feature space (Eq. 9) and the camera regularization terms for flexible bundle adjustment training with distorted camera poses:

$$\mathcal{L}_{FBA} = C(x) + \lambda \mathcal{R}_S(\pi_i) \tag{11}$$

where $\lambda = 0.5$. This FBA loss function is only used for fisheye views.

Fisheye Depth We experimented with fisheye depth estimation, but encountered challenges with convergence during training, leading to disappointing results. Fisheye images often have significant distortions and wide-angle perspectives, making accurate depth estimation more difficult compared to pinhole cameras. However, despite the absence of depth information, the final rendering results for fisheye novel views show promise.

3.4   Training Objectives

In supervised training with ground-truth depth, we use the metric-bin module [5] and SI log-loss to improve depth quality. In self-supervised depth optimization, we use an edge-adaptive smoothness loss [8] *via* pretrained depth priors (for pinhole images only):

$$\mathcal{L}_{depth} = |\partial x D| e^{-|\partial x \mathcal{I}|} + |\partial x D| e^{-|\partial x \mathcal{I}|} \tag{12}$$

In self-supervised training for camera pose refinement, we use the following photometric loss [16]:

$$\mathcal{L}_{photo} = \frac{1}{N} \sum_{j=1}^{N} \alpha \frac{1 - SSIM(\mathcal{I}'_j, \mathcal{I}_o)}{2} + (1 - \alpha) ||\mathcal{I}'_j - \mathcal{I}_o||_2^2 \tag{13}$$

where SSIM is the structural similarity index [43] and $\mathcal{I}'_j$ is the j-th neighboring view warped to the target view $\mathcal{I}_o$. As the photometric error between the target image rendered by IBRNet from nearby views, namely $\widehat{\mathcal{I}}_i$, and the ground-truth target image $\mathcal{I}_i$, we use [41]:

$$\mathcal{L}_{rgb} = ||\widehat{\mathcal{I}}_i - \mathcal{I}_i||_2^2 \tag{14}$$

The final pinhole-image loss function to optimize is:

$$\mathcal{L}_{pinhole} = e^{\beta \cdot t} (\mathcal{L}_{depth} + \mathcal{L}_{photo}) + (1 - e^{\beta \cdot t}) \mathcal{L}_{rgb} \tag{15}$$

where $\beta = $ -1e4 and t is the current training iteration number. For fisheye images, we use flexible bundle adjustment loss $\mathcal{L}_{FBA}$ (Eq. 11) to achieve faster convergence and ensure consistency. We optimize the following fisheye-image loss function with $\beta = $ -1e3:

$$\mathcal{L}_{fisheye} = e^{\beta \cdot t} (\mathcal{L}_{FBA} + \mathcal{L}_{photo}) + (1 - e^{\beta \cdot t}) \mathcal{L}_{rgb} \tag{16}$$

4   Experiments

Training We train Zoe-NK [5] on NYU-depth-v2 dataset to enforce depth priors during planar-pose optimization. We pretrain IBRNet [41], DBARF [8] and our method on 70% of LLFF [27] and 80% of NeRF_360_v2 [2]. Only IBRNet uses pseudo-absolute ground-truth camera poses from COLMAP. We also train SCNeRF and our method on a natural fisheye NeRF dataset [19] and SCNeRF, OMNI-NeRF, and our method on 70% of a synthetic fisheye NeRF dataset [12].

Evaluation We evaluate IBRNet, DBARF, and our method on the LLFF (forward facing) [27] and NeRF_360_v2 [2] pinhole-image datasets. On the other hand, we evaluate SCNeRF [19], OMNI-NeRF [34] and our method on fisheye NeRF datasets [12, 19]. In the supplementary material, we also provide results

on a self-collected dataset to demonstrate FBINeRF's generalization ability. We perform testing of the pinhole model on 10% and 5% of images from LLFF and NeRF_360_v2, respectively (not used during pre-training). Other images are reserved for per-scene fine-tuning. We evaluate SCNeRF [19] and our method using 5-fold cross-validation on the natural fisheye NeRF dataset, and we evaluate SCNeRF, OMNI-NeRF and our method on 20% of the synthetic fisheye NeRF dataset. Additional results are available in the supplementary material.

4.1   Novel-View Synthesis − Pinhole

Depth Priors Fig. 3 shows depth maps predicted by DBARF's depth optimizer and by FBINeRF with ZoeDepth-like pretrained model. The results demonstrate our method's ability to recover more accurate depth compared to DBARF (a quantitative comparison is not possible since no ground-truth depth maps are available in LLFF).
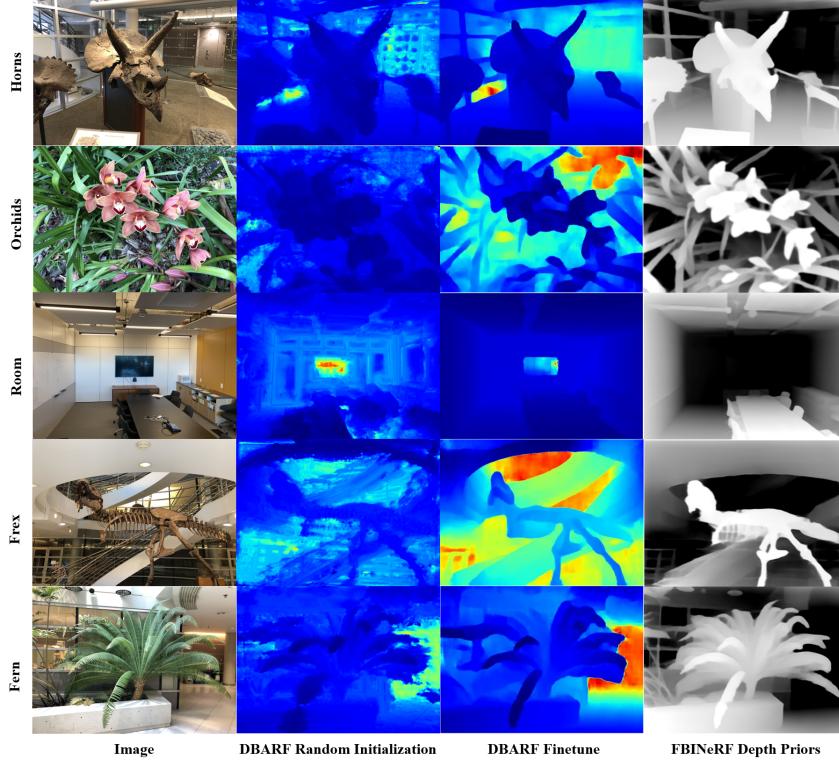

Fig. 3: **Qualitative comparison of depth maps predicted from LLFF dataset** [27]. FBINeRF depth maps (produced by depth priors and adaptive GRU optimizer) are more accurate than from DBARF (generated by depth header) even after finetune.

Moreover, adding a ZoeDepth-like model with metric-bins module extends the model allowing it to learn (including fine-tuning) from ground-truth depth, if available. This applies to both supervised and self-supervised scenarios in FBINeRF, while DBARF is designed for self-supervised depth scenarios only. Next we analyze both qualitative and quantitative results of the pinhole camera model.

Ablation Tables We use PSNR, SSIM [43], and LPIPS [50] as the metrics for novel-view quality assessment. Table 1 shows these metrics for DBARF, IBRNet and our method on the LLFF dataset.

Table 1: Quantitative comparison of novel-view synthesis for LLFF dataset [27]. Each column shows results either with (**w**) or without (**w/o**) per-scene fine-tuning. IBRNet [41] and DBARF [8] results are from the corresponding original papers.

Scenes	PSNR↑ IBRNet w/o	w	DBARF w/o	w	Ours w/o	w	SSIM↑ IBRNet w/o	w	DBARF w/o	w	Ours w/o	w	LPIPS↓ IBRNet w/o	w	DBARF w/o	w	Ours w/o	w
Fern	**23.61**	25.56	23.12	25.97	23.08	**27.91**	**0.743**	0.825	0.724	0.840	0.733	**0.842**	**0.240**	0.139	0.277	0.120	0.279	**0.117**
Flower	22.92	23.94	21.89	**23.95**	**22.97**	23.90	**0.849**	0.895	0.793	**0.895**	0.812	0.889	**0.123**	0.074	0.176	0.074	0.148	**0.071**
Fortress	**29.05**	31.18	28.13	31.43	28.88	**32.58**	0.850	0.918	0.820	**0.918**	**0.863**	0.917	**0.087**	0.046	0.126	0.046	0.098	**0.042**
Horns	24.96	**28.46**	24.17	27.51	**25.01**	26.33	**0.831**	**0.913**	0.799	0.903	0.801	0.903	0.144	**0.070**	0.194	0.076	**0.121**	0.073
Leaves	**19.03**	**21.28**	18.85	20.32	18.70	19.98	0.737	**0.807**	0.649	0.758	**0.752**	0.744	0.289	**0.137**	0.313	0.156	**0.271**	0.147
Orchids	**18.52**	20.83	17.78	20.26	18.33	**21.92**	**0.573**	**0.722**	0.506	0.693	0.529	0.691	**0.259**	0.142	0.352	0.151	0.357	**0.138**
Room	28.81	31.05	27.50	31.09	**28.90**	**32.21**	0.926	**0.950**	0.901	0.947	**0.934**	0.949	0.099	**0.060**	0.142	0.063	**0.089**	0.061
Trex	**23.51**	**26.52**	22.70	22.82	22.03	24.77	**0.818**	0.905	0.783	0.848	0.794	**0.912**	**0.160**	**0.074**	0.207	0.120	0.168	0.084
Average	**23.80**	26.10	23.02	25.42	23.49	**26.20**	**0.791**	**0.867**	0.747	0.850	0.777	0.856	**0.175**	0.093	0.223	0.101	0.191	**0.092**

Analysis Table 1 shows these metrics for DBARF, IBRNet and our method on the LLFF dataset. Our method outperforms DBARF in the *average* value of each metric both with and without per-scene fine-tuning. While without per-scene fine-tuning our method is outperformed by IBRNet, this is not surprising since IBRNet is tuned to the LLFF dataset and uses absolute camera poses whereas our method *estimates* relative camera poses.

Table 2: Training time comparison of novel-view synthesis for LLFF dataset [27]. We compare time consumption in view selection and pose refinement between DBARF and our method. IBRNet uses absolute pose and no depth is involved, so we exclude it.

Scenes	View Select DBARF w/o	w	Ours w/o	w	Feature Extract DBARF w/o	w	Ours w/o	w	Pose Refine DBARF w/o	w	Ours w/o	w
LLFF	4.62	4.78	**1.33**	1.65	**1.94**	1.98	3.56	3.75	5.12	5.73	**2.68**	2.89

Analysis Table 2 reveals that DBARF requires more time for view selection and pose refinement compared to our method. However, it excels in feature extraction since it uses a ResNet-like backbone. The values in Table 2 represent a relative time cost of training. Both methods processed the entire LLFF dataset

and generated rendered views for all scenes. Overall, the training time required by our method is notably smaller than that of DBARF.

4.2   Novel-View Synthesis − Fisheye

Synthetic Dataset Fig. 4 demonstrates improvements offered by our method over SCNeRF and OMNI-NeRF on scenes from a synthetic fisheye NeRF dataset [12]. While both SCNeRF and OMNI-NeRF produce very visible distorions in the clouds, tree and car wheel (top images) and on the mirror and drawer (bottom images), FBINeRF generates very accurate renderings.
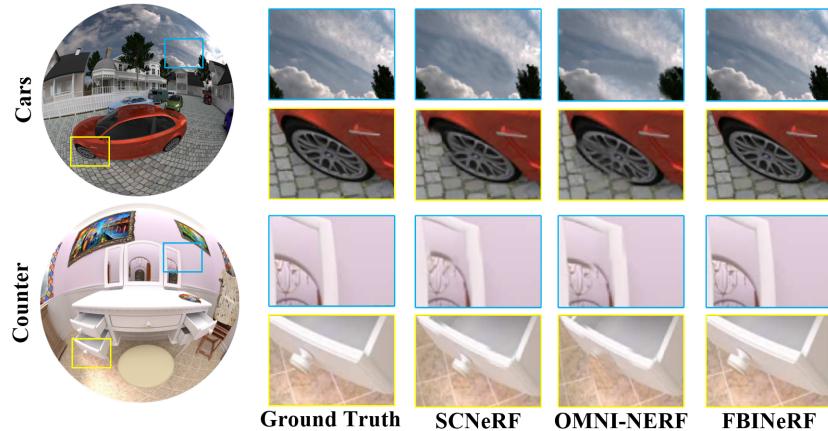

Ground Truth   SCNeRF   OMNI-NERF   FBINeRF

Fig. 4: Qualitative comparison of novel views generated from a synthetic fisheye NeRF dataset [12]. Both SCNeRF and OMNI-NeRF produce clear distorions in the clouds, tree and car wheel (top) and on the mirror and drawer (bottom) whereas FBINeRF does not.

Table 3: Quantitative comparison of novel-view synthesis for a natural fisheye NeRF dataset [19]. The FBINeRF results here are for spherical ray sampling while the third column of images in Fig. 5 are rendered via rectified poses.

Scenes	PSNR↑		SSIM↑		LPIPS↓	
	SCNeRF	FBINeRF	SCNeRF	FBINeRF	SCNeRF	FBINeRF
Chair	17.62	**24.29**	0.312	**0.428**	**0.129**	0.134
Rock	18.91	**25.18**	0.201	**0.225**	0.223	**0.112**
Heart	25.33	**27.94**	**0.520**	0.402	0.451	**0.089**
Flowers	**28.19**	26.73	0.792	**0.822**	**0.197**	0.378
Globe	23.70	**27.67**	0.838	**0.910**	0.210	**0.096**
Cube	27.44	**29.22**	0.692	**0.793**	0.514	**0.081**
Average	23.53	**26.83**	0.559	**0.597**	0.287	**0.148**

Fisheye NeRF Dataset Fig. 5 shows novel-view generation results on a natural fisheye NeRF dataset [19] for SCNeRF and our method. Very obvious blur-

ring can be seen in the views rendered by SCNeRF while those generated by FBINeRF are sharp and very close to the ground truth. However, we could not synthesize novel views from this dataset using OMNI-NeRF due to a mismatch between parameters available in the dataset and those required by OMNI-NeRF. We would like to point out that SCNeRF necessitates long training. One possible reason is its use of the projected ray distance loss function which requires volume rendering at each training iteration (each novel fisheye-view scene requires almost 6 hours to be generated without continuous rendering of views).

Meshes Figure 6 shows examples of meshes produced FBINeRF that can be used as a dense representation for downstream tasks.

5   Limitation and Discussion

Our method can produce high-fidelity synthesic views for both pinhole and fisheye cameras. However, the presence of abnormal geometric-distortion parameters can lead to failures in the inference stage when using pre-trained models. This is because fisheye images require more precise pose estimation, leading to reduced generalization of our method. In principle, depth maps in a fisheye-image scenario could be jointly optimized with camera poses. However, a simple third-order polynomial approximation of geometric distortions cannot accurately model images from a fisheye camera. In our future work, we are planing to research self-supervised depth methods for fisheye cameras that can be incorporated into our method to speed up the training process. We are also planning to incorporate more accurate lens-distortion modeling.

6   Conclusion

In this work, we have identified persistent challenges that SOTA NeRF methods face. While generalized NeRF methods are sensitive to inaccurate depth initialization and are inefficient in updating feature cost maps, fisheye NeRF methods suffer from severe visual artifacts due to radial distortions in the input views. To address these issues, we developed FBINeRF that incorporates DenseNet and AdamR-GRU-like optimizers to enhance pose optimization. FBINeRF also improves depth initialization by introducing depth priors through a ZoeDepth-like pretrained model thus facilitating both supervised and self-supervised scenarios. Preliminary results of our attempt to integrate a feature-based deep recurrent neural network with flexible-bundle adjustment for fisheye-image modeling in NeRF are also promising, suggesting the potential for further enhancements.
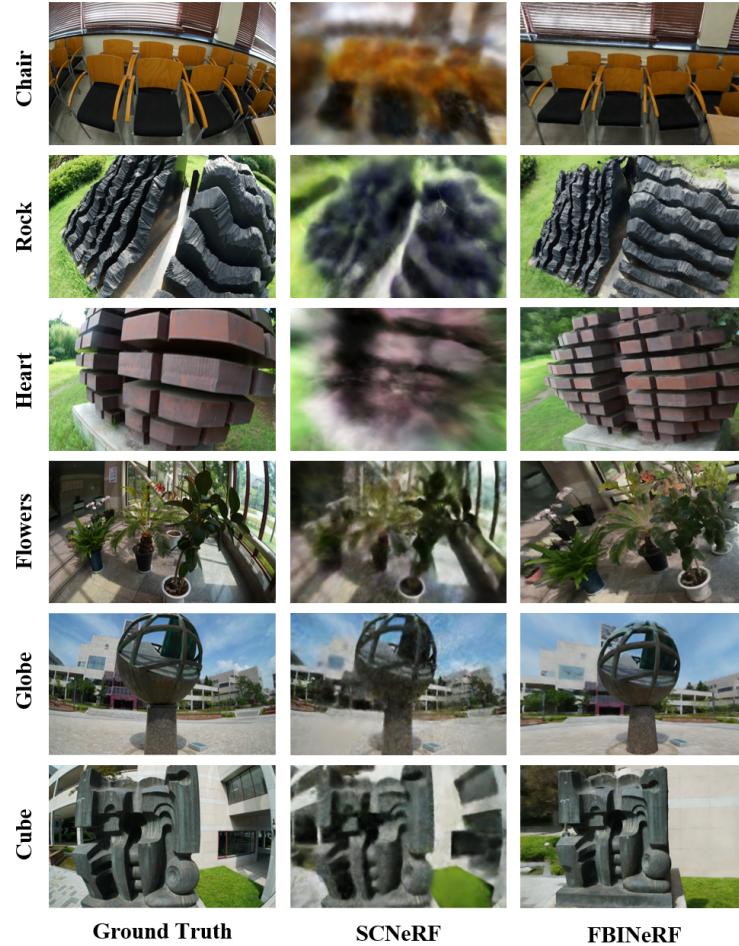
Fig. 5: Qualitative results on fisheye NeRF dataset [19] for SCNeRF and FBIN-eRF. We show qualitative results for our method and SCNeRF on the fisheye NeRF dataset [19]. As we can see, our method produces sharper, less noisy, and more accurate rendered views than SCNeRF. On average, SCNeRF requires over 6 hours to train for each scene in this dataset to obtain results shown in Fig. 5 while ours generates continuous dense voxel fisheye novel views within half an hour.
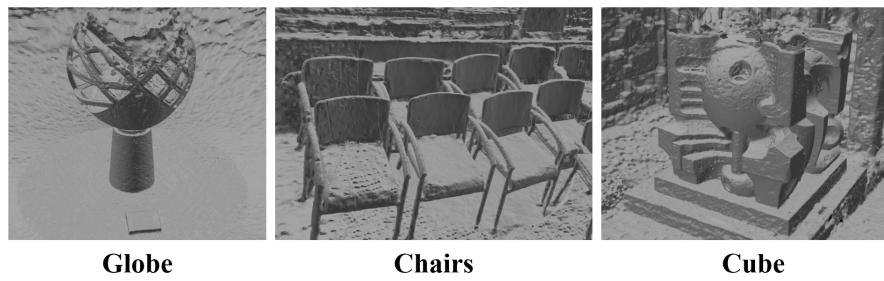
Globe   **Chairs**   **Cube**

Fig. 6: Qualitative mesh results from fisheye NeRF dataset [19] produced by
FBINeRF. Meshes from FBINeRF can be imported into Unity, Unreal Engine, Blender
software for rendering or other downstream tasks.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields (2021) 4
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields (2022) 4, 9
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields (2023) 4
4. Bhat, S.F., Alhashim, I., Wonka, P.: Localbins: Improving depth estimation by learning local distributions (2022) 4
5. Bhat, S.F., Birkl, R., Wofk, D., Wonka, P., Müller, M.: Zoedepth: Zero-shot transfer by combining relative and metric depth (2023) 3, 4, 5, 9
6. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields (2022) 4
7. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo (2021) 2, 4
8. Chen, Y., Lee, G.H.: Dbarf: Deep bundle-adjusting generalizable neural radiance fields (2023) 2, 3, 5, 9, 11
9. Chen, Y., Lee, G.H.: Dreg-nerf: Deep registration for neural radiance fields (2023) 4
10. Chng, S.F., Ramasinghe, S., Sherrah, J., Lucey, S.: Garf: Gaussian activated radiance fields for high fidelity reconstruction and pose estimation (2022) 4
11. Choi, S., Zhou, Q.Y., Koltun, V.: Robust reconstruction of indoor scenes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5556–5565 (2015). https://doi.org/10.1109/CVPR.2015.7299195 3
12. Eichenseer, A., Kaup, A.: A data set providing synthetic and real-world fisheye video sequences (2022) 2, 7, 9, 12
13. Farooq Bhat, S., Alhashim, I., Wonka, P.: Adabins: Depth estimation using adaptive bins. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (Jun 2021). https://doi.org/10.1109/cvpr46437.2021.00400, http://dx.doi.org/10.1109/CVPR46437.2021.00400 4
14. Gava, C., Mukunda, V., Habtegebrial, T., Raue, F., Palacio, S., Dengel, A.: Sphereglue: Learning keypoint matching on high resolution spherical images (06 2023). https://doi.org/10.1109/CVPRW59228.2023.00653 5
15. Godard, C., Aodha, O.M., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency (2017) 3
16. Gu, X., Yuan, W., Dai, Z., Zhu, S., Tang, C., Dong, Z., Tan, P.: Dro: Deep recurrent optimizer for video to depth (2023) 3, 6, 9
17. Henry, P., Krainin, M., Herbst, E.V., Ren, X., Fox, D.: Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In: International Symposium on Experimental Robotics (2010) 3
18. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2018) 3, 5, 6
19. Jeong, Y., Ahn, S., Choy, C., Anandkumar, A., Cho, M., Park, J.: Self-calibrating neural radiance fields (2021) 2, 4, 9, 10, 12, 14, 15
20. Lee, J.H., Kim, C.S.: Monocular depth estimation using relative depth maps. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 4

21. Li, Z., Wang, X., Liu, X., Jiang, J.: Binsformer: Revisiting adaptive bins for monocular depth estimation (2022) 4
22. Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields (2021) 2
23. Lindenberger, P., Sarlin, P.E., Larsson, V., Pollefeys, M.: Pixel-perfect structure-from-motion with featuremetric refinement (2021) 1
24. Longuet-Higgins, H.: A computer algorithm for reconstructing a scene from two projections. In: Fischler, M.A., Firschein, O. (eds.) Readings in Computer Vision, pp. 61–62. Morgan Kaufmann, San Francisco (CA) (1987). https://doi.org/https://doi.org/10.1016/B978-0-08-051581-6.50012-X, https://www.sciencedirect.com/science/article/pii/B978008051581650012X 3
25. Meng, Q., Chen, A., Luo, H., Wu, M., Su, H., Xu, L., He, X., Yu, J.: Gnerf: Gan-based neural radiance field without posed camera (2021) 4
26. Mertan, A., Duff, D.J., Unal, G.: Single image depth estimation: An overview. Digital Signal Processing **123**, 103441 (Apr 2022). https://doi.org/10.1016/j.dsp.2022.103441, http://dx.doi.org/10.1016/j.dsp.2022.103441 4
27. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines (2019) 9, 10, 11
28. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis (2020) 1
29. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics **41**(4), 1–15 (Jul 2022). https://doi.org/10.1145/3528223.3530127, http://dx.doi.org/10.1145/3528223.3530127 4
30. Pal, A., Singh, K.P.: Adamr-grus: Adaptive momentum-based regularized gru for hmer problems. Applied Soft Computing **143**, 110457 (2023). https://doi.org/https://doi.org/10.1016/j.asoc.2023.110457, https://www.sciencedirect.com/science/article/pii/S1568494623004751 5, 6
31. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation (2017) 4
32. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer (2020) 4
33. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4104–4113 (2016). https://doi.org/10.1109/CVPR.2016.445 1
34. Shen, J., Song, B., Wu, Z., Xu, Y.: Omninerf: Hybriding omnidirectional distance and radiance fields for neural surface reconstruction (2022) 4, 7, 9
35. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction (2022) 4
36. Szeliski, R.: Computer vision: algorithms and applications. Springer Nature (2022) 5, 7
37. Tang, C., Tan, P.: BA-net: Dense bundle adjustment networks. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=B1gabhRcYX 5
38. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment - a modern synthesis. In: Workshop on Vision Algorithms (1999), https://api.semanticscholar.org/CorpusID:1354186 1
39. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023) 4

40. Wang, P., Liu, Y., Chen, Z., Liu, L., Liu, Z., Komura, T., Theobalt, C., Wang, W.: F^2-nerf: Fast neural radiance field training with free camera trajectories (2023) 4

41. Wang, Q., Wang, Z., Genova, K., Srinivasan, P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering (2021) 2, 4, 9, 11

42. Wang, X., Liu, Z., Hu, Y., Xi, W., Yu, W., Zou, D.: Featurebooster: Boosting feature descriptors with a lightweight neural network (2023) 4, 5

43. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004). https://doi.org/10.1109/TIP.2003.819861 9, 11

44. Wang, Z., Wu, S., Xie, W., Chen, M., Prisacariu, V.A.: Nerf–: Neural radiance fields without known camera parameters (2022) 4

45. Xia, Y., Tang, H., Timofte, R., Gool, L.V.: Sinerf: Sinusoidal neural radiance fields for joint pose estimation and scene reconstruction (2022) 4

46. Yen-Chen, L., Florence, P., Barron, J.T., Rodriguez, A., Isola, P., Lin, T.Y.: Inerf: Inverting neural radiance fields for pose estimation (2021) 4

47. Yu, A., Fridovich-Keil, S., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks (2021) 4

48. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images (2021) 4

49. Zhang, J., Zhan, F., Wu, R., Yu, Y., Zhang, W., Song, B., Zhang, X., Lu, S.: Vmrf: View matching neural radiance fields (2023) 4

50. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric (2018) 11

51. Zhang, Y., Funkhouser, T.: Deep depth completion of a single rgb-d image (2018) 3

52. Zhu, S., Shen, T., Zhou, L., Zhang, R., Wang, J., Fang, T., Quan, L.: Parallel structure from motion from local increment to global averaging (2017) 1