Multi-layer Scene Representation from Composed Focal Stacks

Reina Ishikawa, Hideo Saito, Denis Kalkofen, and Shohei Mori

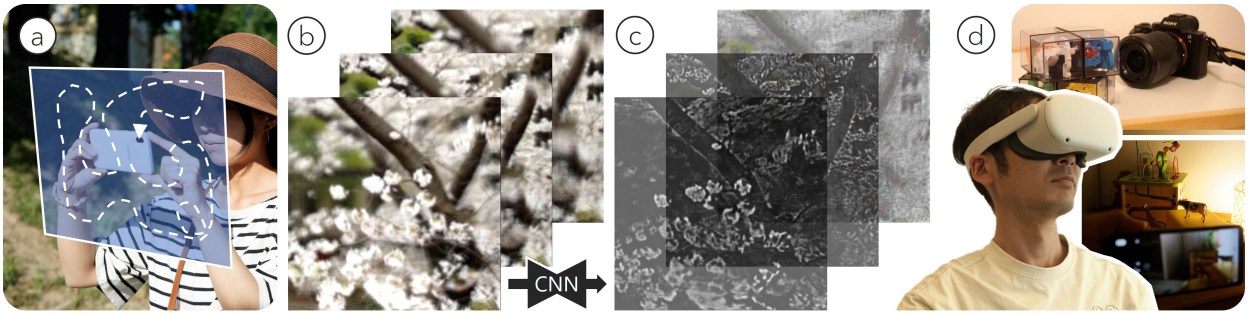

Fig. 1: We propose representing scenes as composed focal stacks, which we compute from registered images, for view synthesis. Our approach enables diminishing local artifacts such as motion blur and ISO noise, to improve multi-layer scene representations. (a) Our approach allows the user to randomly capture photos in a continuous motion. (b) From the captured images, we compose a synthetic focal stack, from which we derive a multi-layer scene representation using a CNN. (c) The outcome of the network is a multi-layer scene representation, which enables continuous viewpoint changes. (d) Our approach supports several applications, including 6 degrees of freedom wide field of view scene representation, which enables photo-realistic VR applications (left), scene representations from underexposed images (right), and multi-layer image generation from captured focal stacks (top).

Abstract—Multi-layer images are a powerful scene representation for high-performance rendering in virtual/augmented reality (VR/AR). The major approach to generate such images is to use a deep neural network trained to encode colors and alpha values of depth certainty on each layer using registered multi-view images. A typical network is aimed at using a limited number of nearest views. Therefore, local noises in input images from a user-navigated camera deteriorate the final rendering quality and interfere with coherency over view transitions. We propose to use a focal stack composed of multi-view inputs to diminish such noises. We also provide theoretical analysis for ideal focal stacks to generate multi-layer images. Our results demonstrate the advantages of using focal stacks in coherent rendering, memory footprint, and AR-supported data capturing. We also show three applications of imaging for VR.

Index Terms—Multi-layered scene representation, focal stack, view synthesis, AR-supported imaging

1 Introduction

View synthesis from multi-view input has been in great demand for decades, starting from image-to-image view warping [49] to plenoptic sampling via explicit proxies [6, 26], and recently neural networks [52]. The selection of proxies for efficient pixel resampling and interpolation has been one of the major design decisions that users need to make. Virtual reality (VR) and augmented reality (AR) especially require high-quality and -performance rendering with limited hardware resources so that the system does not impose any uncomfortable experiences [53]. Apart from the implicit representation of neural radiance fields (NeRF), which requires per-scene training, recent research uses a multi-layer scene representation [13, 33], which consists of explicit RGB+α textured layer proxies such as front-facing planes (i.e., multi-plane image (MPI)) [51, 60], and co-axial spheres (i.e., multi-sphere image (MSI)) [1, 5] for efficiency. Every layer is composed of the furthest to closest layers, and thus, the data structure allows access to the well-studied raster-scan rendering pipeline with parallel computing on GPU. Multi-layer scene representation, therefore, performs as an efficient light field representation at discrete depth samples [33].

Convolutional neural network (CNN) enables to analyze pixel-wise scene depth certainties from multi-view resources and to encode them as alpha values (i.e., opaqueness) more robustly than handcrafted feature description [51, 60]. User captured images from a tracked camera (e.g., AR-supporting smartphones), however, can be a major error source even with AR visual guidance [33], and thus, careful and time-consuming calibration [47] has been required. With multiple registered images ($N \geq 2$), occluded areas are partly observed, and the scale is known as baseline lengths [33], and more input images may potentially lead to improved robustness [37] in inference. Alas, the current network is designed for the nearest $N = 5$ views following the strict capture guidance [33], and thus, those networks may suffer local errors in captured images when inferring MPIs. The design bottleneck comes from a significantly increased memory footprint as CNN receives N plane sweep volumes (PSVs), which increases an original image by D of the number of layers. Another solution takes more views into account with sacrificed rendering speed [56].

To advantageously take more images into MPI generation, we propose using a focal stack that can be composed of the multi-view inputs and thus has a constant size tensor regardless of the number of registered images (Fig. 1). The potential impact of this rather straightforward approach is not only the reduced memory footprint but also the robustness against locally erratic poses and images over the captured scene. Our framework allows accepting an arbitrary number of images. Therefore, the resultant focal stack input can lower local impacts by composing them with the other more accurate registered images, which results in a coherent rendering over the recorded area.

Multi-layer scene generation via focal stack imaging can also bring us better usability in capturing and opens up new applications. Multi-

- *Reina Ishikawa and Hideo Saito are with Keio University. E-mail: hs@keio.jp.*
- *Denis Kalkofen is with Flinders University and Graz University of Technology. E-mail: kalkofen@icg.tugraz.at*
- *Shohei Mori is with Graz University of technology and Keio University. E-mail: s.mori.jp@ieee.org.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

view inputs that satisfy sampling theory in the light field need to be captured in a 2D grid, where AR 3D annotations guide the user [33]. Contrary to strict camera positioning, focal stack synthesis allows us to randomly and casually capture images on a 2D plane because the system arranges focal stacks on a precise 2D grid upon the focal stack synthesis. Inside-out multi-view MPIs for immersive VR is clearly a promising application. In the context of denoising, focal stack imaging allows us to create MPIs with under-exposed images (e.g., in a dark room) or a lens focal stack with a large aperture.

The contributions of this paper can be summarized as follows:

- We propose a system that composes a focal stack from multi-view images to subsequently infer a multi-planar image (MPI) representation of the scene from it. The intermediate focal stack representation allows for diminishing the impact of motion blur, iso-noise, and tracking errors,
- discuss theoretical bounds of transforming focal stacks to MPIs,
- demonstrate the advantages of our approach for MPI generation from focal stacks. In particular, we discuss its robustness against pose and image noises, its benefits to the memory footprint, and the data capturing process, and
- demonstrate and discuss the benefits of our approach for photo-realistic VR applications.

Limitation. In this paper, we conduct experiments using both synthetic and realistic images to demonstrate the effectiveness of the proposed method. Note, however, that the focal stack used in the proposed framework is only a part of an original light field [43]. With the trade-off, we show the robustness, usefulness, and wide application range of the focal stack imaging brought to MPI generation.

2 Background and Related Work

Lightweight dense scene representation may drive many VR/AR applications. 3D mesh representation is one of the most efficient data structures to be rendered with commodity hardware (i.e., GPU). However, its reconstruction of natural scenes has been a challenging topic in the vision and graphics communities [4, 47–49]. Multi-layer scene reconstruction [13, 60] is an emerging technology that may overcome the tradeoff between the efficiency and fidelity of imaging and rendering.

2.1 Multi-layer Scene Representation

Multi-layer scene representations use a layered mesh, either as MPI or MSI, with RGB+α textures. The textured proxies are warped and blended from back to front using alpha compositing with the "over" operator so that the stacked proxies represent full parallax under 6 degrees of freedom camera motion. Given differently shaped proxies that sweep the volumetric space, variants can infer MPI or MSI from a perspective [31,54], stereo [1,50,60], and multi-view images [13,33,51]. Combined with more explicit depth geometry prior, we could expect improved rendering quality with additional costs [10, 21, 28, 61]. Since multi-layer scene representation is a general 3D scene representation empowered by conventional and thus lightweight rendering pipeline, its applications in VR/AR range from 3D scene photos [13] and videos [5] for remote user assistance [45] to a dense scene reconstruction for near-eye displays [12]. Compared to the NeRF implicit representation [34], the multi-layer scene representation is explicit (cf. [19, 36]) and thus editable and plenoptic sampling theory is known valid [33].

The multi-layer scene representation was invented by Szeliski and Golland [51]. A more recent approach by Penner *et al*. explains the alpha values as depth certainties from multi-view depth maps (i.e., soft reconstruction [42]). Zhou *et al*. let a CNN analyze and encode depth certainties as pixel alpha values [60], and since then, CNNs have been the best-practical solution to this problem rather than the previous hand-crafted features. CNNs use per-view PSVs so that the network can "see" the disparities from the multi-view data of a light field. However, the available memory practically limits the number of input views for the network inference since the input ends up with a concatenated N view PSVs. Thus far, a baseline approach, local light field fusion (LLFF), allows $N \leq 5$ views. DeepView [13] can accept more than three viewpoint images, although the required number of CNNs linearly increases.

Instead of having PSV inputs, we propose a framework that takes a digitally composed focal stack to allows using significantly more images ($5 \ll N$) to diminish the local noises in input registered images.

2.2 Light Field Acquisition

The light field represents a scene from 4D ray samples on a proxy, usually two planes. Instead of a direct 4D array [14, 26], we can use data structures that we are spatially more familiar with, such as multi-view images [18, 40]. The quality improves with more input views and appropriate spatial sampling [8, 38]. Multi-layer scene representation is more efficient by means of reduced spatial sampling views by the factor of the number of layers, D, and cheap for rendering [33]. Since multi-layer scene representation is an approximated light field with discrete depth samples, any input data modality for light field reconstruction can potentially be the input for multi-layer scene data reconstruction. The data structures include multi-view images [6, 11, 15, 26], lenslet images [39], coded aperture images [17], and focal stacks [12, 43].

Among those possible options, we use a set of registered multi-view images as an initial input for MPI generation since it allows covering a wider range of the scene without special optics such as coded aperture and mechanical or liquid varifocal lenses followed by pixel resampling for scale corrections. Nonetheless, our network allows using an optically photographed focal stack. We demonstrate such a case as one of our applications (See Section 6).

2.3 Focal Stack Imaging

A focal stack consists of a set of images exposed at different focus distances. When a scene point is in focus (i.e., within the depth of field), the point appears clear within a pixel, while when it is out of focus, the point appears larger than a pixel. Multiple rays in a captured scene passing through a lens with an aperture are eventually encoded into an image. Focal stack is, therefore, when structurally captured, an approximation of a light field [43]. We can capture focal stack optically with varifocal lens [12,22,61], shifting imaging sensor [25], or synthetically with a synthetic aperture photography [18, 39, 55]. Given calibrated images including a reference view, all images are warped to and merged on the reference view image plane back-projected to the space at a distance. Synthetically differently focused images are obtained by varying the plane distance uniformly in diopter distance for efficient capturing.

Therefore, the task for our deep neural network (DNN) is similar to the auto-focus function of a camera, which finds an optimal distance that falls within a depth of field with a modulated lens. Instead of a specific point in space, with a synthetic focal stack, our DNN must describe the certainty of the depth estimation that may distribute over a set of layers so that points are rendered when "over" alpha composited.

Our goal is to implement such a DNN and explore its practical configurations, which include different spatial camera arrangements, the number of input registered images, and tolerance against image noise and registrations.

2.4 Expectations to Synthetic Focal Stack

Recall that the current baseline approach accepts per-view PSVs [33], which ends up with $W \times H \times C \times D \times N$ pixel data for a CNN input, where each denotes width, height, channel, the number of MPI layers, and the number of images, respectively. One of the clear advantages of the use of focal stacks is that we can reduce the input by N, regardless of the number of input images, for MPI inference. With the increased multi-view inputs, we expect more stable multi-layer scene reconstruction, which may follow the same analogy to dense simultaneous localization and mapping (SLAM) [37] compared with keyframe-based SLAM [23]. With more images with shorter baselines, our network should be able to disambiguate points in the sense of triangulation or defocus analysis.

We expect more robust and consistent multi-layer scene reconstruction over the input views, from the synthetic composition of multiple

Fig. 2: Composing focal stack from multi-view registered images under different local noises (Motion blur, ISO noise, and positional noise). The first row shows close-up example noisy images near a simulated noise (See Section 5.5 for the details). The second row shows composed focal stacks from images, including the noisy images. The focal stack composition diminishes the apparent noises.

scene images. Prolonged exposure or multiple exposures is a well-known photography trick to take a noise-free image. In the focal stack composition, misaligned photos are naturally down-weighted by the other more accurately registered photos if the latter is the majority (Fig. 2).

3 View Synthesis Pipeline

Our view synthesis pipeline consists of three steps, including video capture with poses and focal stack composition from the registered video frames (Section 3.1), CNN-based MPI generation (Section 3.2), and scene rendering with the MPIs (Section 3.3).

3.1 Posed Video Capture and Focal Stack Composition

Our network accepts focal stacks composited from registered images and thus allows the user to casually capture consecutive frames (i.e., a video) according to a predetermined 3D proxy. Instead of sparsely and strictly locating images one by one in 3D space (cf. prescriptive sampling guidelines [33]), our system automatically selects necessary image samples for the user among the recorded frames. As we describe in Section 4, the aperture size of a synthetic aperture camera [55] and baseline lengths between generated MPI cameras must satisfy the underlying plenoptic sampling theory [8, 33].

We create a synthetic focal stack from N registered images, $\mathscr{I}_k = \{I_k,\ [\mathbf{R}_k|\mathbf{t}_k], \mathbf{K}_k\} \in \mathbf{I}^{\mathrm{MV}}$, that fall within a valid aperture radius, $A/2$. Here, I_k is an image, $\mathbf{R}_k$ and $\mathbf{t}_k$ represent a relative $\mathbb{SO}(3)$ rotation matrix and 3D translation vector to a synthetic aperture camera, respectively, and $\mathbf{K}_k$ is a 3×3 camera matrix. A resultant focal stack consists of D images placed at z_i $(i \in \{0,1,...,D-1\},\ z_i > z_{i+1}) \in \mathbf{z}$ distances. To this end, all registered images are projected to every image plane at z_i of a target synthetic aperture camera, $\mathscr{I}_{\mathrm{tgt}}$, summed up, and normalized by the number of projected pixels on each pixel. Given a target synthetic aperture camera with $\mathbf{K}_{\mathrm{tgt}}$ and the forward-facing direction $\mathbf{n}_{\mathrm{tgt}}$, such an image-to-image projection is calculated by a Homography matrix of

$$\mathbf{H}_{k,\mathrm{tgt}} = \mathbf{K}_{\mathrm{tgt}} \frac{\mathbf{R}_{k,\mathrm{tgt}} - \mathbf{t}_{k,\mathrm{tgt}}\mathbf{n}_{\mathrm{tgt}}^{\mathrm{T}}}{z_i} \mathbf{K}_k^{-1}. \tag{1}$$

Thus, the normalized pixel color $c_i(\tilde{\mathbf{u}})$ at $\tilde{\mathbf{u}} = [u, v, 1]^{\mathrm{T}}$ can be represented as

$$c_{\mathrm{tgt}}(\tilde{\mathbf{u}}) = \frac{\sum_{k=0}^{N-1} I_k(\mathbf{H}_{k,\mathrm{tgt}}^{-1}\tilde{\mathbf{u}})}{\sum_{k=0}^{N-1} \mathbb{1}_k(\mathbf{H}_{k,\mathrm{tgt}}^{-1}\tilde{\mathbf{u}})}, \tag{2}$$

where $\mathbb{1}$ is an indicator function that returns 0 or 1:

$$\mathbb{1}_k(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{u} \text{ lays within } k_{\mathrm{th}} \text{ camera FoV} \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

We denote the above focal stack rendering process as

$$\mathscr{R}^{\mathrm{FS}}(\mathbf{I}^{\mathrm{MV}},\ \mathbf{z}) \rightarrow \mathbf{I}^{\mathrm{FS}}, \tag{4}$$

where $\mathbf{I}^{\mathrm{FS}}$ is the resultant focal stack with $W \times H \times 3 \times D$ dimensions.

3.2 Multi-Layered Scene Generation

After generating a synthetic D-layer focal stack, we feed it into a CNN similar to U-Net [46], as a widely used baseline, to infer D-layer MPI. Further discussions on other network design choices to be explored are presented in Section 7.

Network structure. Since the up-convolutions in the original U-Net are known to cause checker pattern artifacts in the output, we replaced the up-convolutions with simple 2×2 bilinear up-sampling. We assume the output MPI layers are also equally spaced in inverse depth from back to front. We trained our network to output RGBα values at each pixel on each depth layer of such an MPI. Therefore, the inference is represented as

$$\mathscr{N}(\mathbf{I}^{\mathrm{FS}}) \rightarrow \mathbf{I}^{\mathrm{MPI}}, \tag{5}$$

where $\mathscr{N}$ denotes our network and $\mathbf{I}^{\mathrm{MPI}}$ is a $W \times H \times 4 \times D$ MPI.

Training objectives. We implement a differentiable renderer to optimize our network with respect to the rendered view. To render a novel viewpoint image, we use the "over" alpha composition [44]:

$$\mathscr{R}^{\mathrm{Over}}(\mathbf{I}_{\mathrm{MPI}}) := \sum_{i=0}^{D-1} \left(C_i^{\mathrm{MPI}} * \alpha_i^{\mathrm{MPI}} * \prod_{j=i+1}^{D-1} \left(\mathbf{1} - \alpha_j^{\mathrm{MPI}}\right) \right), \tag{6}$$

where C_i^{MPI} and α_i^{MPI} are color and alpha values of i_{th} MPI layer, respectively, i.e., $(C_i,\ \alpha_i^{\mathrm{MPI}}) \in \mathbf{I}^{\mathrm{MPI}}$. Generalizing the renderer for N-view images as $\mathscr{R}_N^{\mathrm{Over}}$, we train our network to minimize a loss function, $\mathscr{L}$:

$$\arg\min_{\mathbb{W}} \mathscr{L}(\mathscr{R}^{\mathrm{FS}}(\mathscr{R}_N^{\mathrm{Over}}(\mathscr{N}(\mathbf{I}^{\mathrm{FS}})), \mathbf{z}),\ \mathbf{I}^{\mathrm{FS}}), \tag{7}$$

where $\mathbb{W}$ denotes $\mathscr{N}$'s network weights.

The objective $\mathscr{L}$ consists of a L1 loss for the learnable histogram loss [57], $\mathscr{L}^{\mathrm{Hist}}$, with 256 bins, the perceptual loss, $\mathscr{L}^{\mathrm{Percept}}$, with the backbone of *VGG16* [29], and standard L1 loss, $\mathscr{L}^{\mathrm{L1}}$ as

$$\mathscr{L} = \mathscr{L}^{\mathrm{Percept}} + \lambda\mathscr{L}^{\mathrm{L1}} + \gamma\mathscr{L}^{\mathrm{Hist}}. \tag{8}$$

We found such training is practically difficult on a GPU with limited memory space. Therefore, we separate our training into two steps. First,

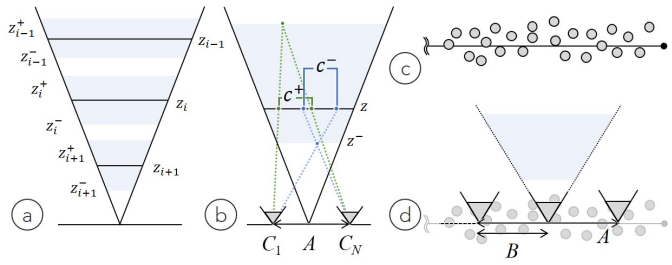

Fig. 3: Optimal Depth of Field in Synthetic Aperture for MPI Generation and View Resampling: (a) Symbols representing the depth of field of each focal stack slice. (b) The depth of field and disparities for a synthetic focal stack slice. (c) View samples scattered from the ideal capture plane in practice. (d) Resampled focal stack views.

we train the network with a loss between a single ground-truth image, I_{tgt}, and its corresponding rendered image as

$$\arg\min_{\mathbb{W}} \mathscr{L}(\mathscr{R}^{\text{Over}}(\mathbf{I}^{\text{MPI}}),\, I_{\text{tgt}}) + \mathscr{L}(\mathscr{R}^{\text{Over}}(\mathbf{I}'^{\text{MPI}}),\, I_{\text{tgt}}), \tag{9}$$

where $\mathbf{I}'^{\text{MPI}}$ represents an MPI whose colors are replaced with those of the original focal stack input, $\mathbf{I}^{\text{FS}}$ (equation 4), with $\lambda = 0$ and $\gamma = 1$. The latter term in equation 9 gives reasoning to distribute colors over MPI layers. Without it, we observed that all colors appear only in the farthest layer, as the objective may explain the perspective. After the loss for a single view optimization in equation 9 converges, we optimize our network for focal stacks as in eq. 7. For the focal stack loss, we sum the loss of every layer with $\lambda = 1$ and $\gamma = 0$.

3.3 Rendering

We apply the local light field fusion approach [33] that renders the multiple MPIs for a linear approximation of non-Lambertian scenes. Namely, we take n-closest MPIs from a rendered view ($n = 5$ by default), render them in individual render buffer, and blend them into one to compensate occluded views in each MPI. Refer to equation 8 in the literature [33] for the details. As we discuss in the next section, our approach allows synthetic focal stack views, thus MPIs as well, to be placed uniformly on a grid in the 2D proxy case because the views are reassembled when forming the focal stacks. Therefore, by the viewpoint resampling, smoother view transitions are expected [20].

4 Theoretical Bounds

Since our network has to encode color and soft depth values at individual MPI layers from a focal stack, the focal stack should guarantee that a point in space is observed without blurs, at least in one layer. With this configuration, our network does not have to guess new colors out of blurred images. To this end, we describe theoretical bounds for the synthetic aperture size and intervals between MPI cameras from given camera parameters and depth range. Once those parameters are calculated and the user captures a set of registered images, synthetic aperture cameras are located, and focal stacks are composited with a set of registered images fully automatically. Consequently, such data capturing requires filling in a specified 2D surface with registered images using a tracked camera. Refer to Table 1 for symbols.

4.1 Largest Synthetic Aperture Size

Given D layers at z_i ($i \in \{0, 1, ..., D-1\}$, $z_i > z_{i+1}$) focus distances, DoF at z_i of a synthetic aperture camera with aperture size, A, is bounded by $[z_i^+, z_i^-]$. In a flat world without losing generality (Fig. 3), all scene points fall within the DoF appear sharp, and therefore, DoFs must overlap to cover the defined depth volume $[z_i, z_{D-1}]$ (Fig. 3a) as

$$z_{i+1}^+ \geq z_i^-. \tag{10}$$

For z_i, the furthest and closest points within the DoF appear c_i^+ and c_i^- on the image plane, respectively. Therefore, from the similar triangles in Fig. 3b,

$$z_i^- = \frac{A z_i}{A + c_i^-} \text{ and } z_i^+ = \frac{A z_i}{A - c_i^+}. \tag{11}$$

Substituting equation 11 to equation 10 and reformulating the equation, we obtain

$$A \leq \frac{z_i c_{i+1}^+ + z_{i+1} c_i^-}{z_i - z_{i+1}}. \tag{12}$$

This equation describes the upper bound of the aperture size. To clarify the relationship between the equation and camera parameters for practical usage, we express c^- and c^+ (no indexing for brevity) with camera FoV, θ_{fov}, and camera image width in pixels, W_{px}, as

$$c^- = c^+ = \frac{2 C_{\text{px}} \tan(\theta_{\text{fov}}/2)}{W_{\text{px}}}, \tag{13}$$

where C_{px} is the maximum disparity in pixels, which is 1 for the highest quality. Substituting equation 13 to equation 12, we obtain

$$A \leq \frac{4 C_{\text{px}} z_i z_{i+1} \tan(\theta_{\text{fov}}/2)}{W_{\text{px}}(z_i - z_{i+1})}. \tag{14}$$

Since layers are linearly placed in inverse depth space, intervals between two layers, Δz^{-1}, are calculated as

$$\Delta z^{-1} = \frac{1}{D-1}\left(\frac{1}{z_{D-1}} - \frac{1}{z_0}\right) = \frac{z_0 - z_{D-1}}{(D-1) z_0 z_{D-1}} \Leftrightarrow \frac{z_i - z_{i+1}}{z_i z_{i+1}}. \tag{15}$$

Therefore, substituting equation 15 to equation 14 results in

$$A \leq \frac{4 C_{\text{px}} \tan(\theta_{\text{fov}}/2)}{W_{\text{px}} \Delta z^{-1}}. \tag{16}$$

Further, the camera view frustums must overlap at a point at z_{D-1}^-, which gives the following additional constraint,

$$A \leq 2 z_{D-1} tan(\theta_{\text{fov}}/2). \tag{17}$$

Overall, our synthetic aperture camera's aperture size is bounded by equations 16 and 17,

$$A \leq \min\left(\frac{4 C_{\text{px}} \tan(\theta_{\text{fov}}/2)}{W_{\text{px}} \Delta z^{-1}},\, 2 z_{D-1} tan(\theta_{\text{fov}}/2)\right). \tag{18}$$

For a practical example, with $D = 32$, $\theta_{fov} = \pi/3$, $W_{px} = 256$, $z_0 = 9.0$ m, and $z_{D-1} = 1.0$ m, the aperture upper bound A is 0.315 m.

Reassembling sets of multi-view inputs (Fig. 3c) bounded by the aperture into focal stacks (Fig. 3d) is beneficial both for capturing and rendering. The user only needs to take photos by roughly moving the camera while recording a video in the tracked space [11], rather than aligning the camera precisely to visual annotations [33]. For the rendering, the system can position the focal stacks in a uniform way, which results in smooth rendering results [20].

Table 1: Reference for symbols used in Section 4. By camera, we refer to either synthetic aperture camera, MPI camera, or physical camera for those sharing the same intrinsic parameters.

Symbol	Unit	Definition
W_{px}	pixels	Camera image width
θ_{fov}	radian	Camera FoV
A	meter	Synthetic aperture size
D	none	Number of layers
C_{px}	pixels	Maximum circle of confusion
B	meter	Baseline between two MPIs
N	none	Number of MPIs
z_i	meter	i_{th} layer or focal distance
$z^{-/+}$	meter	Close/far depth of field (DoF) range
$c^{-/+}$	meter	Circle of confusion for $z^{-/+}$

4.2 Largest Baseline Between MPIs

While equation 18 guarantees the quality of an MPI, one MPI can cover only a limited range. Therefore, multiple MPIs are necessary to render a scene fully. In addition, baselines between MPIs should satisfy the sampling theory to avoid aliasing on rendered images with the minimum number of MPIs. As addressed in the literature [33], the maximum baseline length, B, is bounded as

$$B \leq \min\left(\frac{2C_{\mathrm{px}}\tan(\theta_{\mathrm{fov}}/2)}{w_{px}\Delta z^{-1}},\ z_{D-1}\tan(\theta/2)\right). \tag{19}$$

As a note, the maximum B is equal to half of the maximum A in equation 18 since focal stack imaging does not take occlusions into account in the calculation [8].

5 Experimental Evaluation

To validate the impact of local noises in input registered images, we compare the MPI results of our approach and baselines, which do not require per-scene training (e.g., NeRF [34]). As discussed in Sections 1 and 2, a focal stack is a subset of a light field [43]. Therefore, we are interested more in consistency in rendering quality over a scene than per-image absolute scores, especially when local noises exist in captured registered images. We further discuss the required memory footprint, number of input images, and data capture scheme in the context of MPI generation from registered images from a tracked phone.

5.1 Approaches under Comparison

We prepared two baseline approaches, LLFF [33] and IBRNet [56], with pre-trained weights provided by the authors and compared them with our approach. Upon discussing the network structures, we also compare their memory consumption. However, since all three methods use different frameworks that rely on specific hardware, a fair comparison is difficult on a machine. Therefore, we compare the amount of input data. Table 2 shows the dependent variables and Fig. 4 shows memory consumption on inference with $W = 640$, $H = 480$, and $D = 32$.

LLFF. LLFF generates MPIs from the five nearest registered multi-view images and renders the sparsely sampled multi-MPI with alpha blending. Each MPI is, therefore, generated at each camera location. The number of input images is fixed to five of a reference, top, bottom, left, and right views positioned with the support of the capture guideline. In other words, the user must take photos on a navigation grid that indicates ideal sampling intervals. If one may want to use the LLFF network with more than five views to lower the impact of local errors in MPI inference, required memory significantly increases with larger N. The training, therefore, becomes infeasible. The main cause is PSV, which expands one image to D images to keep the raw light field information.

IBRNet. The network learns the view interpolation function to estimate radiance and volume density and then renders a novel view using volume rendering. The network structure can accept multiple views as much as GPU memory limits permit. However, the rendering performance is limited due to the rendering scheme without a lightweight medium such as MPI. Note that IBRNet, therefore, relies only on N. However, no discussion on the appropriate N or spacing in capturing is provided in the literature [56].

Ours. Our pipeline allows to the composition of all input images within the $A \times A$ aperture area into a focal stack. In other words, our network infers MPI from a focal stack. As such, our network does not rely on N. In addition, since the focal stacks are generated at arbitrary virtual viewpoints, the spacing between each focal stack or MPI can be kept exactly at the interval B guaranteed by the sampling theorems.

5.2 Dataset

For training our network and quantitative comparisons, we generated a synthetic dataset in a similar manner to that of the literature [58]. We rely on the synthetic dataset to insert controlled local noises. Namely, we trained our network without any noises and evaluated the influences by inserting noises to test data instead of adding noises in the training step (*cf.*, data augmentation). Due to the defocus nature, we expect a smaller impact on virtual–real domain shift [32].

We created 120 synthetic scenes in total using an open source software, *Blender* [9]. The dataset is separated into 80, 20, and 20 scenes for training, validation, and testing, respectively. For training and validation scenes, 121 virtual cameras in an 11×11 grid are placed. The side length of the grid is set to the same as the maximum aperture size, A, calculated in equation 18. For the test dataset, we set up 441 cameras of 21×21 images with the same camera baselines as the other datasets. Individual methods were evaluated by comparing the center 11×11 ground-truth images with images rendered at the same locations to avoid missing border cameras (See Fig. 5a).

3D objects from *Thingi10K* [59] were randomly generated from 1.0 to 10.0 meters away from the camera array plane. A textured quad was placed at 10.0 meters to cover the FoV completely. The objects had either a uniform color or a textured pattern whose color distribution follows that of *CIFER-10* [24]. The camera resolution is fixed to 256×256 pixels, and the horizontal and vertical FoV was 56.2475°.

5.3 Training Details

We implemented our network with the PyTorch framework [41] v1.11.0. We used the computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by the National Institute of Advanced Industrial Science and Technology (AIST) to train our network. We used the RMSprop optimizer to train the network (learning rate: $1^{-2} \rightarrow 1^{-3}$ (eq. 9 $\rightarrow$ eq. 7), weight decay: 1^{-8}, momentum: 0.9, and batch size:

Table 2: Tensor size per inference.

Alg.	Input image size
LLFF	$W \times H \times C \times N \times D$
IBRNet	$W \times H \times C \times N$
Ours	$W \times H \times C \times D$

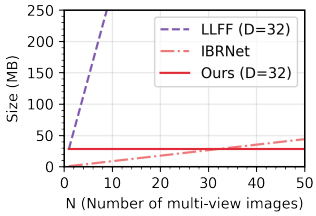

Fig. 4: Data amount per inference.

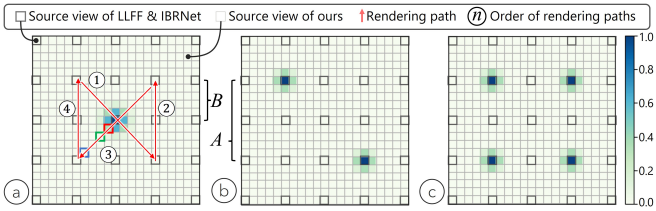

Fig. 5: Local noises added to our test dataset. The bluish gradations show the noise level. We added positional noise, ISO- noise, and motion blur to (a) one, (b) two, and (c) four locations. The red, green, and blue rectangles in (a) correspond to the rendering locations in Fig. 7.

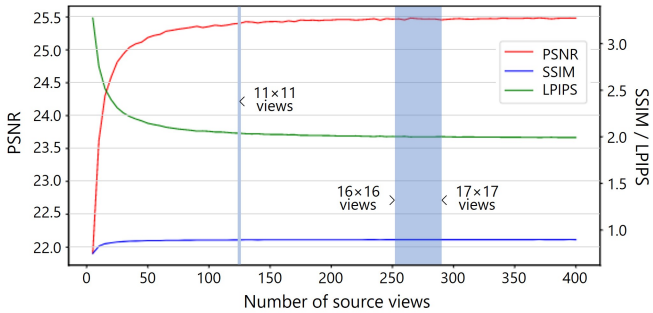

Fig. 6: PSNR, SSIM, and LPIPS quality measures over different numbers of source views for our network trained with 11×11 views. The improvements saturate between 16×16 and 17×17 views (the blue band), which approximates the plenoptic sampling theory well.

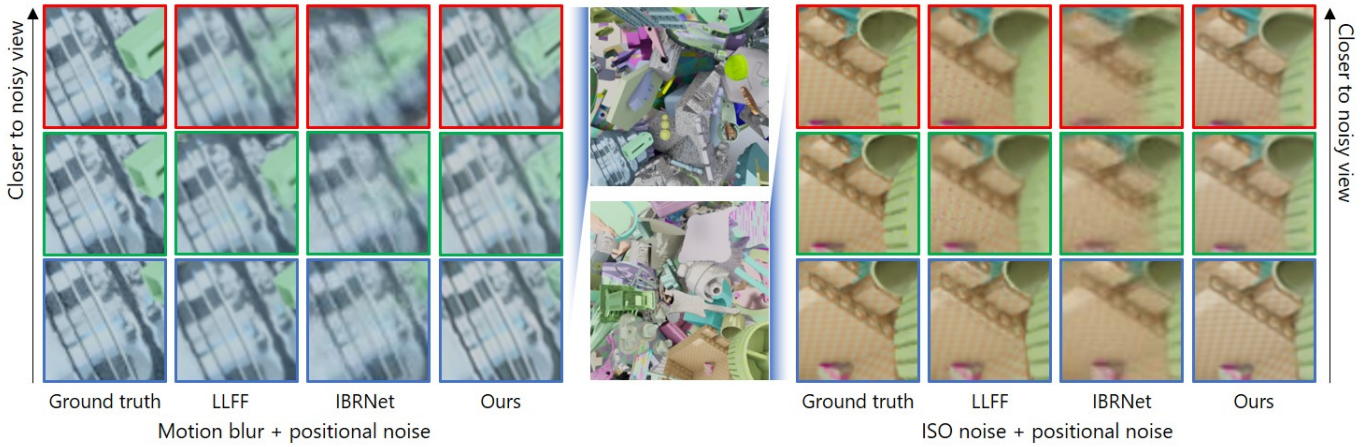

Fig. 7: Qualitative comparison of rendered images under (left) motion blur and positional noise and (right) ISO noise and positional noise. The red, green, and blur frames of the enlarged figures indicate the rendering positions depicted in Fig. 5a. While our approach keeps the quality over different locations, the contenders show degraded results where the noise appears close (red and green frames).

$4 \rightarrow 1$). All tests were executed on a computer with an Intel(R) Xeon(R) W-3235 CPU and an NVIDIA Titan RTX GPU (24GB). Due to the GPU memory limit, we fixed the number of depth layers to $D = 32$.

5.4 Number of Input Views

Although our framework can accept an arbitrary number of source views, there should be quality upper and lower bounds depending on the number of views.

Fig. 6 shows the relationship between the rendering quality and the number of source views given to our network trained with 11×11 views. For tests with denser input views, we created another dataset, which contains $1681 (= 41 \times 41)$ images within the $2A \times 2A$ range, and we evaluated by rendering images to the same locations of the center 21×21 ground-truth images, as in the same manner as explained in Section 5.2. For each sample of input view numbers, we randomly selected the required number of views from the available 441 images, generated focal stacks, fed them to our network, and evaluated the rendering quality in three different quality metrics: Peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS) of $\mathscr{L}^{\text{Percept}}$. We calculated average values over 21×21 test images of 20 test scenes.

The rendering quality reaches its peak between at 16×16 $(= 256)$ and 17×17 $(= 289)$ views. Given the optimal aperture size in our dataset, the baseline between two cameras is 9.59 mm and 8.99 mm for 16×16 and 17×17 views, respectively, while, according to the plenoptic sampling theory, the minimum baseline is 9.28 mm that lies between the above-mentioned values. We, therefore, conclude that our network is well-generalized with the training data. Nonetheless, in the following results, we fix the number of input views to 121 for a good compromise between storage and quality from sparser inputs in practical use cases. Based on the results in Fig. 6, when the $[0,\ 100]\%$ range corresponds to the quality range of $[5,\ 289]$ views, 11×11 views would lead to 95% quality or higher.

5.5 Robustness Against Local Noise in MPI Generation

Purpose. As outlined in Sections 1 and 2, a focal stack can be considered an approximation of a light field. Consequently, given an ideal setup that satisfies sampling theory, our network cannot achieve better MPI reconstruction than LLFF, which employs raw light fields. In this context, we aim to highlight the benefits of using focal stacks generated from registered multi-view images, with a particular focus on enhancing robustness against local noise present in the source views. Such noise can be mitigated through the formation of a focal stack. To this end, we compare our approach with two alternative methods, assessing the gradients of rendering quality during camera movement along a specified path depicted in Fig. 5a. Specifically, we calculate the absolute differences between quality values (PSNR/SSIM/LPIPS) of two consecutive rendered frames, which should remain continuously small to ensure robustness against local noise. The quantitative results are presented in Table 3 under the Grad. column.

Local noise modeling. We add three types of noises to the test dataset that may appear during photography: ISO noise, motion blur, and positional noises. We used the Albumentations library [7], which receives $[0, 1]$ values as intensities to add ISO noises and motion blurs to images. The noise intensities and positional errors follow absolute values of a Gaussian distribution ($\mu = 0.0$, $\sigma = 1.0$) truncated between $[0, 1]$. We scaled the $[0, 1]$ value to $[0, 0.3]$ and $[0, 25]$ for ISO noises and motion blurs, respectively, to keep the image plausible. For positional errors, we scaled the $[0, 1]$ value to $[0, 0.2]$ and used the value as a radial distance in a spherical coordinate system centered at the ideal camera location. We then applied uniform random values for the polar angles and added the resultant values to the ideal camera location. We added such noises at different locations as depicted in Fig. 5.

Source view selection. Our network inference takes all views within the calculated aperture, A, while LLFF network inference takes the five closest views at B intervals to generate an MPI (Fig. 5). We rendered the closest MPI at a time to evaluate per-MPI quality and coherency when switching MPIs for our approach and LLFF. IBRNet rendering used the closest five views.

Results. Fig. 7 presents qualitative comparisons under motion blur and positional noise (left) and ISO noise and positional noise (right). Our approach successfully maintains the rendering quality over different locations regardless of how far the noise is located. On the other hand, the contenders show deteriorated results, such as blurry appearance and destructed structures, when the noise appears close.

Fig. 8 shows quality transitions along the path of Fig. 5a under (a) motion blur and positional noises and (b) ISO noises and positional noises. Table 3 summarizes the quantitative evaluation in gradients and also, for a reference, mean and standard deviation values in each quality metric under different numbers of noises in Fig. 5. Since we observe that typical MPI rendering results are presented after cropping the borders to remove artifacts due to missing warped MPI pixels. To investigate the effects of such cropping, we removed 16 pixels on each side of the rendered image and evaluated the rendering quality (Table 3, with cropping). The results show that cropping particularly performs well for approaches that use MPIs but have less impact on IBRNet.

We find that the gradients of our approach stay low while our contenders suffer from noise, resulting in varying metric scores depending on the rendered locations. IBRNet seems sensitive to the noise and its gradients fluctuate. In other words, only very accurate images can lead to promising rendering results in IBRNet. We note that the best scores

Table 3: Quantitative evaluation in mean gradients, Grad. (↓), under a predetermined path in Fig. 5a in PSNR, SSIM, and LPIPS. Mean, and standard deviation values are for reference. The table on the left shows the evaluation of the rendered images without cropping their edge, while the table on the right shows the results by cropping the perimeter by 16 pixels as in typical MPI rendering. The highest scores in gradients under each noise condition are highlighted in bold fonts.

		without cropping						with cropping					
	# of	PSNR (↑)		SSIM (↑)		LPIPS (↓)		PSNR (↑)		SSIM (↑)		LPIPS (↓)	
Method	noises	Mean (Std.)	Grad.	Mean (Std.)	Grad.	Mean (Std.)	Grad.	Mean (Std.)	Grad.	Mean (Std.)	Grad.	Mean (Std.)	Grad.
LLFF	0	25.87 (2.27)	1.53	0.93 (0.023)	0.014	1.56 (0.29)	0.19	28.24 (1.99)	1.11	0.94 (0.020)	**0.011**	1.53 (0.26)	0.16
	1	24.56 (2.60)	1.46	0.90 (0.054)	0.024	1.87 (0.51)	0.25	26.69 (2.50)	1.20	0.91 (0.049)	0.020	1.81 (0.45)	0.22
	2	24.71 (2.34)	1.55	0.90 (0.049)	0.024	1.84 (0.46)	0.26	26.87 (2.26)	1.24	0.91 (0.044)	0.021	1.78 (0.41)	0.22
	4	23.07 (2.17)	1.17	0.86 (0.064)	0.029	2.23 (0.52)	0.26	24.80 (2.28)	1.00	0.87 (0.059)	0.026	2.16 (0.47)	0.21
IBRNet	0	29.99 (4.77)	2.58	0.95 (0.022)	**0.011**	1.29 (0.38)	0.21	30.56 (5.02)	2.71	0.95 (0.021)	**0.011**	1.40 (0.42)	0.23
	1	26.23 (5.48)	2.68	0.87 (0.088)	0.035	1.95 (0.80)	0.37	26.67 (5.74)	2.79	0.87 (0.088)	0.035	2.02 (0.79)	0.37
	2	26.79 (5.09)	1.55	0.89 (0.090)	0.024	1.82 (0.78)	0.26	27.23 (5.36)	2.90	0.89 (0.091)	0.037	1.91 (0.77)	0.39
	4	23.76 (4.08)	1.91	0.82 (0.105)	0.047	2.38 (0.80)	0.38	24.09 (4.27)	1.99	0.83 (0.11)	0.047	2.43 (0.79)	0.38
Ours	0	23.48 (1.56)	**1.11**	0.90 (0.025)	0.014	1.93 (0.24)	**0.17**	26.02 (1.37)	**0.68**	0.91 (0.025)	0.013	1.93 (0.21)	**0.13**
	1	23.14 (1.42)	**0.89**	0.90 (0.026)	**0.009**	1.99 (0.24)	**0.13**	25.76 (1.35)	**0.41**	0.90 (0.026)	**0.008**	1.98 (0.21)	**0.08**
	2	23.16 (1.40)	**0.90**	0.89 (0.025)	**0.010**	1.98 (0.23)	**0.13**	25.81 (1.33)	**0.42**	0.90 (0.026)	**0.008**	1.97 (0.21)	**0.09**
	4	23.07 (1.35)	**0.86**	0.89 (0.025)	**0.009**	2.00 (0.23)	**0.12**	25.64 (1.30)	**0.38**	0.90 (0.025)	**0.007**	1.99 (0.21)	**0.08**

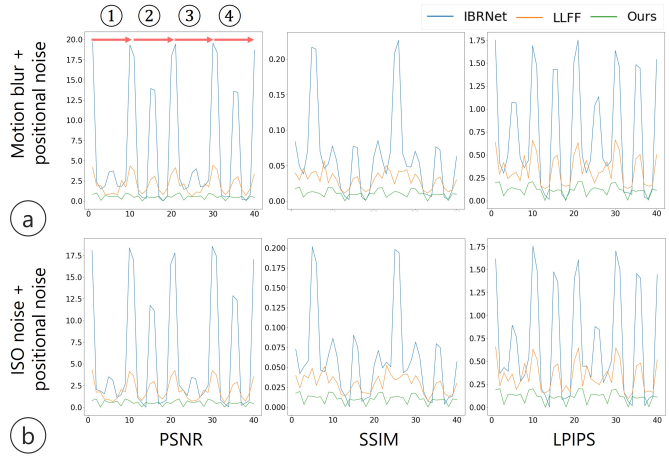

Fig. 8: View consistency evaluation in two different types of local noises without image border cropping. (a) Motion blur and positional noise and (b) ISO noise and positional noise. The numbers and red arrows correspond to those in Fig. 5a. See Grad. in Table 3 for better illustrations of quality transitions over time.

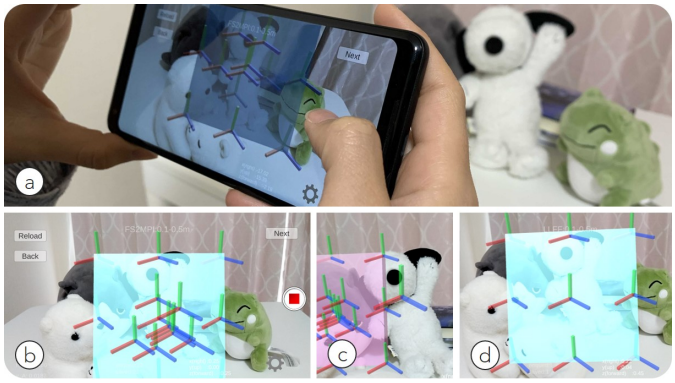

Fig. 9: AR capture app implementation. (a) A 3D flat cube visualizes the valid range for photographing. The placement and size are determined based on the camera configuration and scene distances. (b) Upon imaging, a 3D axis is placed to show the recorded pose and the current coverage. (c) In case the camera is out of the valid range, the cube turns red from blue. (d) For LLFF visualization, we placed 3D axes to be aligned with that registered to the camera frame of reference.

of IBRNet in Table 3 are when no noise is given. The LLFF network, by architecture, finds agreements of five view inputs. Therefore, the impact of an erratic image seems lower. Given four noise spots, our approach presents close-to-equivalent quality even in absolute measure. The artifacts in motion are better seen in the supplemental video.

5.6 Demonstrating View Synthesis Pipeline

Purpose. We demonstrate our entire view synthesis pipeline (Section 3). We take real-scene photos with a tracked phone to infer and render resultant MPIs. Apart from per-MPI evaluations in the synthetic dataset (Section 5.5), we render the nearest four multiple MPIs at a frame (Section 3.3) in a 3×3 grid. We present rendered results of our approach and the contenders for qualitative comparisons.

AR capture app. We implemented our capture application running on an Android Phone (Google Pixel 2XL, Android 11) to capture registered images using Unity[1] 2020.3.23f1 with ARFoundation[2] 4.1.12 for tracking (Fig. 9a). 121 registered images were automatically recorded locally on the phone at five fps after determining the scene- and camera-dependent synthetic aperture size (equation 18). Although our network is trained for a 1 to 10 m range, the range is adjustable with different aperture sizes depending on the scenes. The application visualizes 3D axes to represent recorded registered images and the current pose at the center of the camera FoV. When the first image is recorded, the application registers a semitransparent AR cube registered at the photo pose (Fig. 9a). The cube size indicates the valid capture range, and therefore, width and height correspond to A, and thickness is 0.05 m by default. To avoid diverse changes in depth, the application recorded images only when the pose was within the range. To navigate the photographer to the valid range, the AR cube is colored in blue or red when the phone is within or out of the range, respectively (Fig. 9b, c). The rendered occlusion between the center 3D axis and the cube visualizes if the current pose is behind or in front of the valid range.

MPI generation and rendering. We took photos at the native resolution, 1440×2880 pixels, resized them to 256×512 pixels, applied our network to the left- and right-half of an image individually, and then merged two MPIs into one MPI at the original aspect ratio. MPIs are generated in the exact 3×3 grid (recall Fig. 3c, d). We selected and rendered the four nearest MPIs at a view and blended the rendered frames to fill in disoccluded areas at each view [33].

Rendering results. We recorded four real scenes. For comparison, we present the results of LLFF and IBRNet in the same scene. To use the same real scene dataset, we generated MPIs using the nearest five views for LLFF, which was oversampled. For IBRNet, we used the nearest 10 views to render a frame. Fig. 10 summarizes the rendering

[1] https://unity.com/
[2] https://unity.com/unity/features/arfoundation

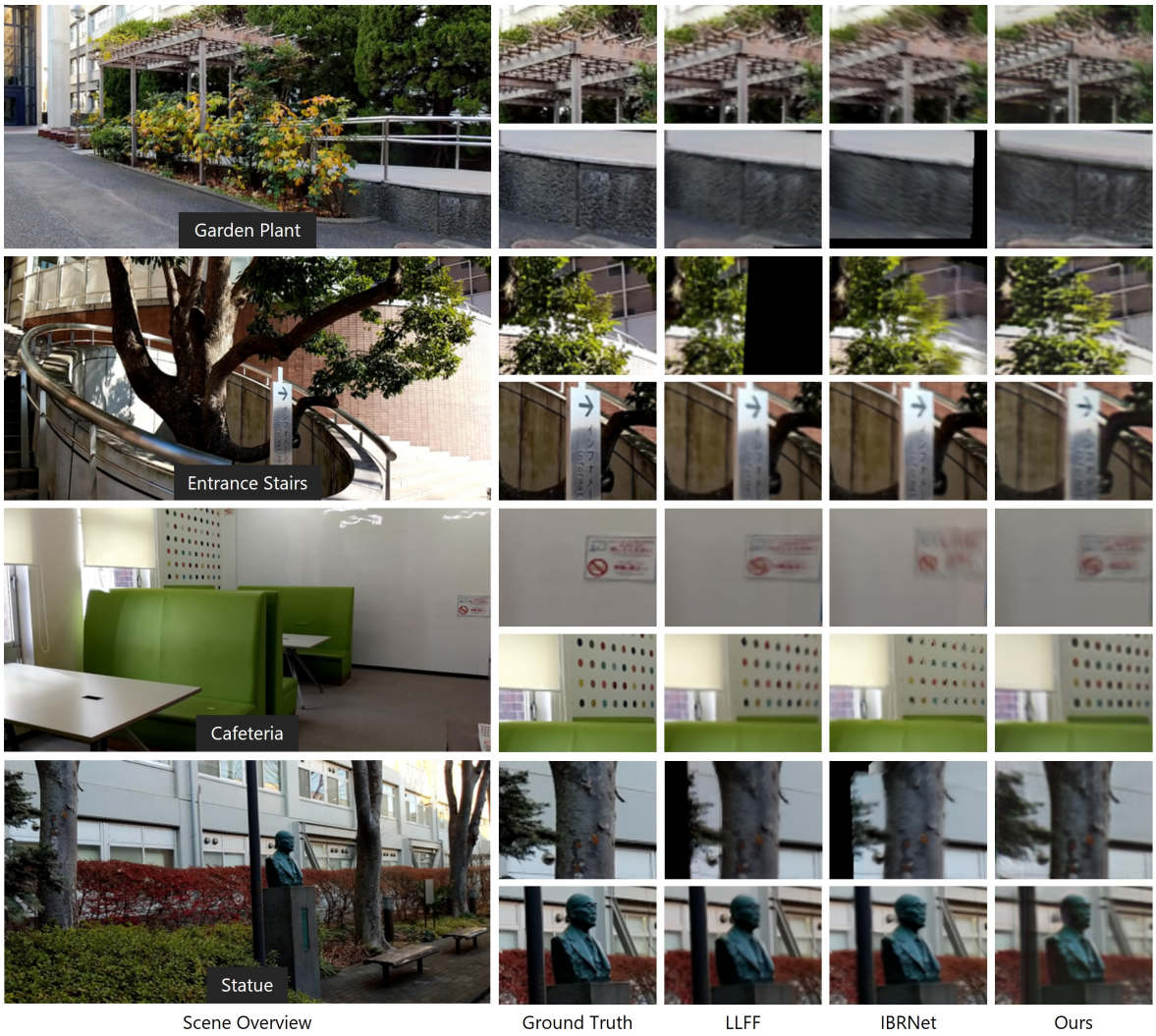

Fig. 10: Example real-scene rendering in four indoor and outdoor real scenes.

results. Although our results show slightly blurred images compared to LLFF, they miss fewer pixels at the image borders owing to the MPIs at exact 3×3 locations. LLFF requires carefully positioned posed images, which could lead to recording efforts for the user (see the discussions in the next section). IBRNet suffers the real cases seemingly due to the limited camera pose accuracy.

5.7 Interactive LF Capturing / User Study

We designed a repeated measures within-subjects study to compare two different AR capture guidance: LLFF [33] and ours. The task for the LLFF guidance was to align 3D axis in a 3×3 grid (Fig. 9d). A registered photograph was saved when the screen-referenced 3D axis collided with the one in the grid. For our visual guidance, we used the same implementation described in Section 5.6. Since the visualization scales depending on the scene scale, we prepared two differently scaled scenes, desktop and room scale, each ranging approximately from 0.1 to 0.5 m and 1.0 to 10.0 m in depth, respectively.

We collected 20 participants (eight females and 12 males, age $\bar{X} = 26.6$, SD $= 11.8$ years old, self-rated experience in AR was $\bar{X} = 2.35$, SD $= 1.3$ in $[1,5]$). After receiving textual instructions and signing an informed consent form, participants performed tasks with randomly selected methods and in randomly selected scene-scales until they covered all combinations. Every after a trial, the participants were asked to score the method in raw NASA Task Load Index (NASA-TLX) [16]. Eventually, we collected 80 $(= 2 \times 2 \times 20)$ ratings.

After performing Shapiro-Wilk and outlier tests, we performed two-way repeated measures analysis of variance (ANOVA) for the independent variable, "raw-TLX score," within two factors, "method" and "scene-scale." Namely, we assumed that (**H1**) our visual guidance is superior in task load due to less restrictive 3D alignments and (**H2**) scene-scale has an impact in task load due to the scale changes in visualization. The analysis revealed significant differences between methods $(F(1,19) = 15.65,\ p < 0.001,\ \eta^2 = 0.076)$ and also between scene-scales $(F(1,19) = 4.99,\ p < 0.05,\ \eta^2 = 0.019)$. The mean difference between methods is 8.73 (ours: $44.10 <$ LLFF: 52.83) and that of scene-scales is 4.19 (desktop: $50.56 >$ room: 46.38). Therefore, both **H1** and **H2** are supported. We found no interaction between the two factors $(F(1,19) = 0.40,\ p = 0.54,,\ \eta^2 = 0.002)$. Overall, our visual guidance requires a lower task load than that of LLFF, and room-scale data capture requires a lower task load than on the desktop scale.

6 Applications

We discuss three applications enabled by the proposed framework.

Denoised MPI. By drawing an analogy to prolonged exposure in photography, we generate a clean MPI using registered multi-view images of a dark room scene captured by a tracked smartphone, Google Pixel 2XL. Fig. 11a illustrates one of the input images with visible ISO noise, the composed focal stack that acts as a denoiser, and the resulting MPI rendering.

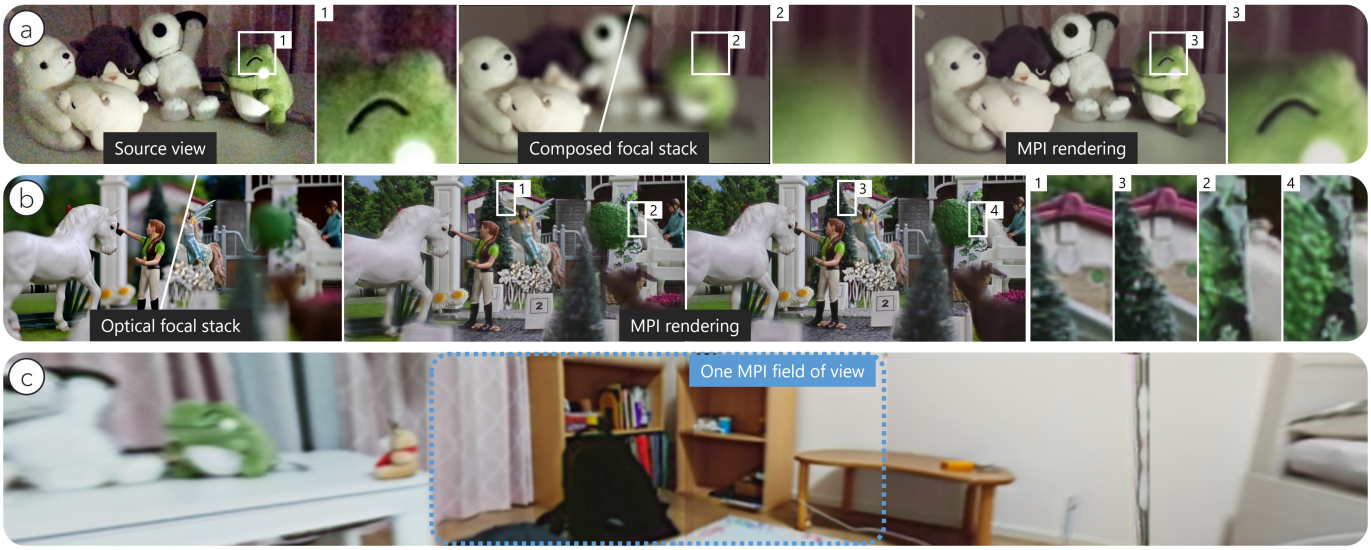

Fig. 11: Three applications of multi-layer scene representation from composed and optical focal stacks. (a) Generating denoised MPI from photographs of a dark room scene. (b) MPI from an optically photographed focal stack with a DSLR camera (Nikon Z6, f/1.8, 35mm). (c) Wide FoV rendering using four MPIs facing different angles inside-out.

Lens focal stack to MPI. Our user study results suggest that AR-supported light field photography appears mentally demanding in small-scale scenes. To solve this issue, we can photograph a focal stack using a stationary lens camera without moving the camera (e.g., a motorized lens in a typical DSLR, a focus tunable lens [12], or actuated image sensor [25]). Our network should be able to accept focal stack images from any of those imaging systems. To this end, we took a focal stack with a DSLR (Nikon Z6, f/1.8, 35mm) in a miniature scene and generated an MPI. Fig. 11b shows the results. Although the viewing range is limited to the lens aperture, we observe scene disparities.

Wide field of view imaging and rendering. Due to the limited camera resolution and physical limitations in optics, photographing a focal stack using a wide FoV camera is difficult. However, a combination of multiple MPIs can form a wide FoV multi-layer scene, which is suitable for immersive VR viewing. We show the simplest implementation by projecting four MPIs facing towards different directions from a center (Fig. 11c). While each focal stack capture proxy is restricted to a plane, integrating multiple MPIs into MSI [5] or layered depth data [28] further improves the quality.

7 Limitations and Future Directions

This paper introduced a framework to generate MPIs from synthetic focal stacks successfully. Here, we discuss several known limitations that the current implementation struggles with.

Approximations in MPI. A focal stack is an approximation of a light field [43]. In our case, all input views are merged into defocus images at different focus distances, and therefore, the original view-dependent nature is lost in averaged values. To maintain the original information better, one may want to preserve differences or distributions additionally as statistically meaningful values, which have been used in the multi-view stereo approaches [47]. Nonetheless, different network training strategies like generative adversarial network (GAN) or an intermediate NeRF representation [27] would improve the generated MPI quality. NeRF from a focal stack is also an interesting direction.

We guarantee occlusions within an MPI only through the loss design in training. Aiming at focal stack imaging as a good approximation of more images than the conventional "five input views," we trained our network so that it reproduces a focal stack from an MPI. In other words, we do not evaluate generated MPIs with the original multi-view inputs. As another aspect, limited GPU memory prevented us from evaluating differentiable renderings against many ground truth images.

A single MPI assumes that specularity appears at a single depth and thus the rendering is narrow-ranged. Apart from single MPI rendering, multi-view MPIs rendering and blending is a practical linear approximation of original light fields [33] as we implemented (Section 3.3).

Network and dataset design. We used a basic U-Net, while, naturally, testing our framework with different network architectures is an interesting future venue. Investigating minor changes in skip connections, network depths, and convolutions [30] would be beneficial for higher quality and speed. 3D CNN gives control of the number of resultant layers, D [33]. Some attempts with residual networks suggest that a few focal stack slices can generate a light field [17], which may apply to MPI generation. Generative networks are potentially advantageous in MPI generation and denoising.

We observed blurry MPI layers, especially at frontal objects (See the blurry tree and goat on the right in MPI rendering results in Fig. 11b). We also found similar cases in our test dataset results. When generating our training dataset, we guaranteed that the randomly generated objects were located within the valid depth of field range by clipping off objects to avoid including extreme cases. However, which may have led to a lack of examples of frontal objects in our training data. Overall, effective object distribution in a test dataset remains future work.

Efficient data recording. Since our approach requires recording hundreds of registered images in practice, efficient data recording clearly contributes to usability and efficiency. Efficient light field capturing is a unique set of challenges but an attractive research area [2, 3, 11, 35]. A tracked all-in-focus camera (e.g., a smartphone with a tiny camera and support of AR functionality) enables sophisticated AR visualization [11, 35] and AR navigation [3] for improved efficiency.

8 Conclusion

This paper proposes a novel imaging-to-rendering framework for view synthesis. Compositing a focal stack from multi-view calibrated images is the key to multi-layer scene (aka. MPI) generation. We use a focal stack composed of input views to incorporate more images from diverse viewpoints and feed it to our U-Net-based network to generate an MPI with suppressed noise. Unlike conventional MPI generation methods that use PSV at each input view, focal stack imaging requires smaller data volume and accepts different imaging devices (i.e., tracked smartphones for all-in-focus multi-viewpoint images and lens cameras for lens focal sweeping). Our results demonstrate coherent rendering over the captured scene and fewer mental loads in our capture mode.

Acknowledgments

This work was supported by the Austrian Science Fund FWF (grant no. P33634). We used the computational resource of AI Bridging Cloud

Infrastructure (ABCI) provided by the National Institute of Advanced Industrial Science and Technology (AIST). The authors thank Shiori Ueda for her support on synthetic dataset generation. The authors also thank Thomas Layer for providing light field data in Fig. 2.

REFERENCES

[1] B. Attal, S. Ling, A. Gokaslan, C. Richardt, and J. Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 441–459. Springer, 2020. 1, 2

[2] C. Birklbauer and O. Bimber. Panorama light-field imaging. In *Computer Graphics Forum*, vol. 33, pp. 43–52. Wiley Online Library, 2014. 9

[3] C. Birklbauer and O. Bimber. Active guidance for light-field photography on smartphones. *Computers and Graphics (C&G)*, 53(PB):127–135, dec 2015. 9

[4] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, vol. 11, pp. 1–11, 2011. 2

[5] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39(4):86–1, 2020. 1, 2, 9

[6] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 425–432, 2001. 1, 2

[7] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. 6

[8] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum. Plenoptic sampling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 307–318, 2000. 2, 3, 5

[9] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 5

[10] P. Dai, Y. Zhang, Z. Li, S. Liu, and B. Zeng. Neural point cloud rendering via multi-plane projection. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7830–7839, 2020. 2

[11] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. In *Computer Graphics Forum*, vol. 31, pp. 305–314. Wiley Online Library, 2012. 2, 4, 9

[12] C. Ebner, S. Mori, P. Mohr, Y. Peng, D. Schmalstieg, G. Wetzstein, and D. Kalkofen. Video see-through mixed reality with focus cues. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2256–2266, 2022. 2, 9

[13] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[14] T. Fujii. Ray space coding for 3d visual communication. In *Picture Coding Symposium'96*, vol. 2, pp. 447–451, 1996. 2

[15] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 43–54, 1996. 2

[16] S. G. Hart. NASA-task load index (NASA-TLX); 20 years later. In *Proc. Human Factors and Ergonomics Society Annual Meeting*, vol. 50, pp. 904–908, 2006. 8

[17] Y. Inagaki, Y. Kobayashi, K. Takahashi, T. Fujii, and H. Nagahara. Learning to capture light fields through a coded aperture camera. In *Proc. European Conference on Computer Vision (ECCV)*, September 2018. 2, 9

[18] A. Isaksen, L. McMillan, and S. J. Gortler. Dynamically reparameterized light fields. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, p. 297–306. ACM Press/Addison-Wesley Publishing Co., USA, 2000. 2

[19] C. Jambon, B. Kerbl, G. Kopanas, S. Diolatzis, G. Drettakis, and T. Leimkühler. Nerfshop: Interactive editing of neural radiance fields. *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 6(1), 2023. 2

[20] P. Kellnhofer, L. Jebe, A. Jones, R. Spicer, K. Pulli, and G. Wetzstein. Neural lumigraph rendering. In *CVPR*, 2021. 4

[21] T. Khakhulin, D. Korzhenkov, P. Solovev, G. Sterkin, A.-T. Ardelean, and V. Lempitsky. Stereo magnification with multi-layer images. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8687–8696, 2022. 2

[22] H. Kim, C. Richardt, and C. Theobalt. Video depth-from-defocus. In *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 370–379. IEEE Computer Society, Los Alamitos, CA, USA, oct 2016. 2

[23] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 225–234. IEEE, 2007. 2

[24] A. Krizhevsky. Learning multiple layers of features from tiny images. pp. 32–33, 2009. 5

[25] S. Kuthirummal, H. Nagahara, C. Zhou, and S. K. Nayar. Flexible depth of field photography. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(1):58–71, 2011. 2, 9

[26] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, p. 31–42. Association for Computing Machinery, New York, NY, USA, 1996. 1, 2

[27] J. Li, Z. Feng, Q. She, H. Ding, C. Wang, and G. H. Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *Proc. International Conference on Computer Vision (ICCV)*, pp. 12578–12588, 2021. 9

[28] K.-E. Lin, Z. Xu, B. Mildenhall, P. P. Srinivasan, Y. Hold-Geoffroy, S. DiVerdi, Q. Sun, K. Sunkavalli, and R. Ramamoorthi. Deep multi depth panoramas for view synthesis. In *European Conference on Computer Vision*, pp. 328–344. Springer, 2020. 2, 9

[29] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 730–734, 2015. 3

[30] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11976–11986, 2022. 9

[31] D. C. Luvizon, G. S. P. Carvalho, A. A. dos Santos, J. S. Conceicao, J. L. Flores-Campana, L. G. L. Decker, M. R. Souza, H. Pedrini, A. Joia, and O. A. B. Penatti. Adaptive multiplane image generation from a single internet picture. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2556–2565, January 2021. 2

[32] M. Maximov, K. Galim, and L. Leal-Taixe. Focus on defocus: Bridging the synthetic to real domain gap for depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 5

[33] 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 Transactions on Graphics (TOG)*, 2019. 1, 2, 3, 4, 5, 7, 8, 9

[34] 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. In *ECCV*, 2020. 2, 5

[35] P. Mohr, S. Mori, T. Langlotz, B. H. Thomas, D. Schmalstieg, and D. Kalkofen. Mixed reality light fields for interactive remote assistance. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pp. 1–12, 2020. 9

[36] S. Mori, D. Schmalstieg, and D. Kalkofen. Exemplar-based inpainting for 6dof virtual reality photos. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2023. 2

[37] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proc. International Conference on Computer Vision (ICCV)*, pp. 2320–2327. IEEE, 2011. 1, 2

[38] R. Ng. Fourier slice photography. In *ACM transactions on graphics (TOG)*, vol. 24, pp. 735–744. ACM, 2005. 2

[39] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light Field Photography with a Hand-Held Plenoptic Camera. Technical report, Stanford Tech Report CTSR 2005-02 Light, 2005. 2

[40] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec. A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018. 2

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. 5

[42] E. Penner and L. Zhang. Soft 3d reconstruction for view synthesis. 36(6), 2017. 2

[43] F. Pérez, A. Pérez, M. Rodríguez, and E. Magdaleno. Lightfield recovery from its focal stack. *Journal of Mathematical Imaging and Vision*, 56(3):573–590, 2016. 2, 5, 9

[44] T. Porter and T. Duff. Compositing digital images. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, p. 253–259. Association for Computing Machinery, New York, NY, USA, 1984. 3

[45] L. Ribeiro Skreinig, A. Stanescu, S. Mori, F. Heyen, P. Mohr, M. Sedlmair, D. Schmalstieg, and D. Kalkofen. Ar hero: Generating interactive augmented reality guitar tutorials. In *IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 395–401, 2022. 2

[46] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241. Springer, 2015. 3

[47] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 9

[48] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. European Conference on Computer Vision (ECCV)*, 2016. 2

[49] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics (TOG)*, 25(3):835–846, jul 2006. 1, 2

[50] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 175–184, 2019. 2

[51] R. Szeliski and P. Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision (IJCV)*, 32(1):45–61, 1999. 1, 2

[52] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, vol. 41, pp. 703–735. Wiley Online Library, 2022. 1

[53] J. Thatte and B. Girod. Towards perceptual evaluation of six degrees of freedom virtual reality rendering from stacked omnistereo representation. *Electronic Imaging*, 2018(5):352–1, 2018. 1

[54] R. Tucker and N. Snavely. Single-view view synthesis with multiplane images. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[55] V. Vaish, B. Wilburn, N. Joshi, and M. Levoy. Using plane + parallax for calibrating dense camera arrays. In *In Proc. CVPR*, pp. 2–9, 2004. 2, 3

[56] 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 *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 5

[57] Z. Wang, H. Li, W. Ouyang, and X. Wang. Learnable histogram: Statistical context features for deep neural networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., *Computer Vision – ECCV 2016*, pp. 246–262. Springer International Publishing, Cham, 2016. 3

[58] L. Xiao, A. Kaplanyan, A. Fix, M. Chapman, and D. Lanman. Deepfocus: Learned image synthesis for computational displays. 37(6), 2018. 5

[59] Q. Zhou and A. Jacobson. Thingi10k: A dataset of 10, 000 3d-printing models. *CoRR*, abs/1605.04797, 2016. 5

[60] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2018. 1, 2

[61] W. Zhou, G. Liu, J. Shi, H. Zhang, and G. Dai. Depth-guided view synthesis for light field reconstruction from a single image. *Image Vision Comput.*, 95(C), mar 2020. 2