DeepFocus: Learned Image Synthesis for Computational Displays

LEI XIAO, Facebook Reality Labs

ANTON KAPLANYAN, Facebook Reality Labs

ALEXANDER FIX, Facebook Reality Labs

MATTHEW CHAPMAN, Facebook Reality Labs

DOUGLAS LANMAN, Facebook Reality Labs

Fig. 1. *DeepFocus* is a unified rendering and optimization framework, based on convolutional neural networks, that enables real-time operation of accommodation-supporting head-mounted displays. (Top row) For varifocal displays, *DeepFocus* produces physically plausible defocus blur from a single RGB-D input, as shown in the simulated retinal images on the right. (Middle row) For multifocal displays, the network outputs a multilayer decomposition from a single RGB-D input. (Bottom row) For light field displays, *DeepFocus* generates dense multiview imagery from a sparse set of RGB-D images.

Addressing vergence-accommodation conflict in head-mounted displays (HMDs) requires resolving two interrelated problems. First, the hardware must support viewing sharp imagery over the full accommodation range of the user. Second, HMDs should accurately reproduce retinal defocus blur to correctly drive accommodation. A multitude of *accommodation-supporting* HMDs have been proposed, with three architectures receiving particular attention: varifocal, multifocal, and light field displays. These designs all extend depth of focus, but rely on computationally expensive rendering and optimization algorithms to reproduce accurate defocus blur (often limiting content complexity and interactive applications). To date, no unified framework has been proposed to support driving these emerging HMDs using commodity content. In this paper, we introduce *DeepFocus*, a generic, end-to-end convolutional neural network designed to efficiently solve the

full range of computational tasks for accommodation-supporting HMDs. This network is demonstrated to accurately synthesize defocus blur, focal stacks, multilayer decompositions, and multiview imagery using only commonly available RGB-D images, enabling real-time, near-correct depictions of retinal blur with a broad set of accommodation-supporting HMDs.

CCS Concepts: • **Computing methodologies** → *Virtual reality; Mixed / augmented reality; Neural networks*;

Additional Key Words and Phrases: computational displays, deep learning, depth of field, varifocal, multifocal, light fields, accommodation

ACM Reference Format:

Lei Xiao, Anton Kaplanyan, Alexander Fix, Matthew Chapman, and Douglas Lanman. 2018. DeepFocus: Learned Image Synthesis for Computational Displays. *ACM Trans. Graph.* 37, 6, Article 200 (November 2018), 13 pages. <https://doi.org/10.1145/3272127.3275032>

1 INTRODUCTION

Computational displays are aimed at bridging the gap between synthesized images and physical reality through the joint design of optics and rendering algorithms, as informed by our ever-expanding knowledge of the human visual system [Masia et al. 2013]. Today's head-mounted displays (HMDs) present a means to more closely approach this goal than prior direct-view displays, depicting accurate perspective, shading, binocular, and motion parallax depth cues.

Authors' addresses: Lei Xiao, Facebook Reality Labs, lei.xiao@fb.com; Anton Kaplanyan, Facebook Reality Labs, anton.kaplanyan@oculus.com; Alexander Fix, Facebook Reality Labs, alexander.fix@oculus.com; Matthew Chapman, Facebook Reality Labs, matt.chapman@oculus.com; Douglas Lanman, Facebook Reality Labs, douglas.lanman@oculus.com.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2018 Copyright held by the owner/author(s).

0730-0301/2018/11-ART200

<https://doi.org/10.1145/3272127.3275032>

However, existing HMDs rely on a fixed optical focus and do not accurately reproduce retinal blur throughout an extended scene, resulting in vergence-accommodation conflict (VAC). Sustained VAC has been associated with biased depth perception [Watt et al. 2012] and visual fatigue [Hoffman et al. 2008; Shibata et al. 2011].

Accommodation-supporting HMDs [Kramida 2016] have been proposed to mitigate VAC, not only with novel optical elements, but also with the joint design of rendering and optimization algorithms. These displays must solve two interrelated problems. First, they must create an extended focus range, i.e., an extended depth of focus (EDoF), such that the viewer can sharply perceive virtual objects from within arm's reach out to the horizon. Second, they must depict perceptually accurate retinal defocus blur. The *varifocal* and *multifocal* displays, shown in Figure 1, are two examples of such EDoF HMDs. Varifocal HMDs continuously adjust the virtual image distance [Dunn et al. 2017], whereas multifocal HMDs create a volumetric depiction using multiple focal planes, further requiring a decomposition algorithm to partition the image across these layers [Narain et al. 2015]. While differing in construction, both designs partially rely on *synthetically rendered blur*, rather than the blur created optically due to the natural accommodative response of the viewer. Without rendered blur, these displays create incorrect cues, which have been linked to diminished depth perception [Held et al. 2010; Zannoli et al. 2016b]. Moreover, recent work by Cholewiak et al. [2017] provides additional evidence that rendered blur may more effectively drive accommodation. While promising, synthesizing perceptually accurate retinal defocus blur is computationally taxing and requires modifications to existing rendering engines.

HMDs that rely on rendered blur often require accurate eye tracking. Near-eye *light field* displays (Figure 1) circumvent this requirement [Huang et al. 2015; Lanman and Luebke 2013]. Such HMDs approximate retinal blur by presenting the optical superposition of many viewpoints. However, these displays introduce another daunting computational challenge: requiring the scene to be rendered from tens (or even hundreds) of viewpoints.

In this paper, we present a unified framework for efficient rendering into these accommodation-supporting HMDs that employs recent advances in machine learning. Specifically, we introduce *DeepFocus*, a generic, end-to-end convolutional neural network (CNN) designed to efficiently solve the full range of computational tasks for emerging near-eye displays. This network synthesizes defocus blur, focal stacks, multilayer decompositions, and multiview imagery—the critical inputs required for all major variants of varifocal, multifocal, and light field displays. Moreover, DeepFocus uses modest inputs we can realistically expect from today's real-time rendering systems: in-focus color image(s) and depth map(s).

1.1 Contributions

- We introduce a novel deep network architecture, tailored to support real-time image synthesis. This network includes volume-preserving interleaving layers, related to the pixel shuffling layer introduced by Shi et al. [2016], to reduce the spatial dimensions of the input, while fully preserving image details, allowing for significantly improved run times.

- We synthesize physically plausible defocus blur in real-time from a single RGB-D image, supporting both conventional and varifocal displays.
- We generalize this network to output a focal stack, with defocus blur inferred for a discrete set of focal distances, supporting optimal rendering for multifocal displays.
- We demonstrate real-time multilayer decompositions for multifocal displays, taking either complete focal stacks or a single RGB-D image as input to directly solve this computationally expensive inverse optimization problem. By accepting direct RGB-D inputs, we further avoid the computational overhead introduced by focal stack generation.
- Finally, we extend our network to the task of generating a dense light field from a sparse set of RGB-D images, supporting near-eye light field displays.

For all of these rendering problems, we achieve high accuracy and efficiency while using a single network architecture (differing only in the number of layers and interleaving/de-interleaving rates), suggesting that these results generalize across applications.

2 RELATED WORK

2.1 Retinal Defocus Blur and Accommodation

DeepFocus is inspired by increasing evidence of the important role retinal defocus blur plays in driving accommodative responses, as well as the perception of depth and physical realism. Smithline [1974] identified retinal defocus blur, chromatic aberration, and looming as potentially involved in accommodative control. Burge and Geisler [2011] reported reliable depth estimates solely from defocus blur. Synthesizing accurate defocus blur has also been shown to result in the correct perception of depth and scale [Held et al. 2012, 2010]. Using a multifocal display, Zannoli et al. [2016a] found retinal defocus blur is sufficient to recover depth ordering. Moreover, Mauderer et al. [2014] and Cholewiak et al. [2017] demonstrated that accurate retinal defocus blur increases perceived realism. We emphasize that DeepFocus is designed to be generalizable: as our understanding of the necessary qualities of retinal blur advances, so can the depictions learned to be synthesized by the network.

2.2 Accommodation-Supporting Displays

2.2.1 Varifocal and EDoF Displays. Most HMDs have a simple construction: a display is positioned within the focal length of an eyepiece to create a virtual image at a fixed distance. Shiwa et al. [1996] first proposed a varifocal configuration, translating the lens to alter the image distance. When coupled with eye tracking, varifocal HMDs extend the depth of focus, as demonstrated by Dunn et al. [2017], Padmanaban et al. [2017] and Kouliris et al. [2017]. Alternatives to lens movement include electronically tunable lenses [Konrad et al. 2016; Liu and Hua 2010], deformable mirrors [Dunn et al. 2017], and translated mirrors [Akşit et al. 2017].

Varifocal HMDs are not the only means to extend depth of focus. Von Waldkirch [2005] and Konrad et al. [2017] demonstrate *accommodation-invariant* HMDs, using pinhole apertures and electronically tunable lenses to minimize variation of the point spread function as the user accommodates.

Varifocal and other EDoF HMDs must rely on rendered defocus blur, as points in the virtual scene will not project with *perceptually correct* retinal blur. As a result, delivering correct accommodation cues with such HMDs requires not only hardware innovation, but also the development of real-time rendering of defocus blur.

2.2.2 Rendering Defocus Blur. Synthetically generated defocus blur can be produced by simulating a virtual camera with a finite aperture. Conventionally, the aperture is sampled via stochastic ray tracing [Cook et al. 1984], with the accumulation buffer [Haerberli and Akeley 1990] being a well-known variant. While such methods produce physically accurate blur, they require many samples and, thus, are not suitable for real-time applications. Two classes of real-time algorithms have emerged. Methods from the first class apply defocus effects before compositing the final image [Křivánek et al. 2003; Lee et al. 2009, 2008; Selgrad et al. 2015]. Such approaches achieve high visual quality, but come at the cost of resolving visibility. The second class comprises methods that filter in image space. Shinya [1994] was the first to propose such post-processing for approximating defocus blur from a single image with color and depth (RGB-D), and this approach remains widely used in real-time applications [Demers 2004; Yang et al. 2016]. We refer the reader to Barsky [2008] for extended discussion. DeepFocus falls into the latter class: effectively applying post-processing so that existing rendering engines can apply it to create physically plausible blur.

2.2.3 Multifocal Displays. Neil et al. [1997] introduced multifocal HMDs to eliminate the need for eye tracking by creating multiple virtual images. Such displays often rely on temporal multiplexing: refreshing the display as the eyepiece focal length is rapidly modulated [Hu and Hua 2014; Love et al. 2009; Wu et al. 2016]. As the viewer accommodates, differently defocused images superimpose on the retina. Thus, a *multilayer decomposition* must partition the scene across these images. Akeley et al. [2004] introduced “linear blending” for this task, depicting each point in the RGB-D input on the two nearest layers. More recently, Narain et al. [2015] introduced “optimized blending” to address visual artifacts produced by this method, which occur at depth discontinuities and with translucent or reflective materials. These benefits come at a high computational cost, with a reported runtime on the order of minutes. Mercier et al. [2017] recently accelerated this algorithm to near-interactive frame rates. To our knowledge, we are the first to produce high-resolution optimized multilayer decompositions, in real-time, using learned image synthesis. Furthermore, we are the first to do so by optimizing directly from RGB-D inputs, further reducing computational overhead by eliminating focal stack rendering.

2.2.4 Light Field Displays. Light field HMDs contain an array of miniaturized projectors, often constructed by placing a microlens array over a single display. Lanman and Luebke [2013] and Hua and Javidi [2014] describe virtual and augmented reality configurations, respectively. As a user accommodates, the image created by each projector shifts on the retina, with the resulting superposition approximating natural defocus blur without eye tracking.

As shown in Figure 1, light field HMDs require synthesizing dense multiview imagery. Prior works have evaluated these views directly by ray tracing and rasterization, limiting content to simplified scenes

for real-time interaction. More recently, Sun et al. [2017] introduced foveated light field rendering. While reducing the computational overhead, this approach continues to rely on ray tracing, limiting performance, and reintroduces the requirement for eye tracking.

Multiview image synthesis is also required for direct-view auto-multiscopic displays. Didyk et al. [2013] and Kellnhofer et al. [2017] synthesize views from stereoscopic content. Widmer et al. [2015] apply screen-space ray tracing to synthesize light fields from stereoscopic RGB-D inputs. Other methods taking only RGB inputs have been proposed [Chaurasia et al. 2013; Shi et al. 2014; Wanner and Goldluecke 2014; Zhang et al. 2015]. To our knowledge, we are the first to tailor image synthesis networks for view generation from sparse RGB-D inputs for real-time HMD applications.

2.3 Deep Learning for Image Synthesis

2.3.1 Common Network Architectures. Encoder-decoder neural networks have a “U-Net” [Ronneberger et al. 2015] shape, with successive downsampling (pooling) followed by upsampling (unpooling) to produce the final image. Applications span denoising [Chaitanya et al. 2017], inpainting [Pathak et al. 2016], and screen-space rendering [Nalbach et al. 2017], the latter of which includes defocus blur rendering. The encoder-decoder architecture gives the output neurons a large receptive field, although to preserve detail in the final output, skip connections from the encoder layers to corresponding decoder layers are necessary. Alternatively, the network can maintain full resolution at all layers, without any pooling. The receptive field of the output neurons can be increased with dilated-convolution network (“Dilated-Net”) [Yu and Koltun 2016].

Residual networks [He et al. 2016], developed to ease training of very deep networks, have proven effective for image processing. A residual block adds its input layer to the output of its second layer through a residual connection. With all weights set to zero, the residual block is the identity function — small variations near zero allow the residual block to learn to *refine* its input. This refinement is relevant to our applications, where the desired output is close to the input but with added blur or moderately shifted viewpoints.

2.3.2 CNNs for Defocus and Multiview Synthesis. The use of deep networks for depth of field rendering was considered by Nalbach et al. [2017], who address multiple screen-space shading tasks. Their network is an encoder-decoder architecture. We compare directly with this method in Section 5. DeepFocus achieves better performance, both in PSNR and perceptual metrics like SSIM and HDR-VDP-2, despite their network being trained to maximize SSIM.

For light field synthesis, Flynn et al. [2016] use deep networks to synthesize novel views from images with wide baselines. Wu et al. [2017] take advantage of the texture structure of the epipolar plane image for light field view synthesis. Both methods take several minutes to run. Kalantari et al. [2016] demonstrate view reconstruction in a light field camera using only the four corner views, with follow up work [Srinivasan et al. 2017] using only the central image. This method uses two sequential networks in a pipelined approach: the first network predicts disparity, per pixel, and a second network predicts color images from this disparity and pre-warped versions of the input images. Reported runtimes are seconds per frame. For

Fig. 2. The DeepFocus network architecture. The network includes a pair of novel interleaving and de-interleaving layers that reduce spatial resolution of input features while preserving their volume, as well as convolutional layers composed of residual blocks and skip connections that operate on the interleaved layer. The reshaping that interleaving performs is depicted here by the movement of color-coded pixels between the RGB-D input and the interleaved layer. In this example, the interleaving rate $r = 2$, but this hyperparameter can be tailored for specific applications. See Section 3 for a full explanation of the network.

display applications, we do not need a separate network to predict disparity, as it is provided as part of the RGB-D input.

DeepFocus leverages learnings from the evolving literature in image synthesis networks, introducing a new architecture designed to enable efficient HMD rendering and optimization.

3 METHOD

3.1 Network Architecture

In this section, we describe our network architecture and the required inputs and training procedures to support rendering for varifocal (Section 3.2), multifocal (Section 3.3), and light field displays (Section 3.4). Our framework provides a unified design to efficiently solve these problems with high quality and real-time performance.

We illustrate our network architecture in Figure 2. At a high level, the network contains a volume-preserving interleaving layer, a number of fully convolutional layers with residual blocks [He et al. 2016] and with a long-distance skip connection, and finally a volume-preserving de-interleaving layer. We explain each component in the following subsections.

3.1.1 Volume-Preserving Interleaving and De-interleaving Layers.

Time complexity of convolutional networks is linearly proportional to the spatial resolution of input features. To improve runtime performance, previous work [Chaitanya et al. 2017; Nalbach et al. 2017; Ronneberger et al. 2015] apply (max or mean) pooling to reduce the spatial resolution of features. These pooling techniques were initially used for high-level vision tasks, to aggregate information across large receptive fields, but for image synthesis applications the information loss from pooling may degrade image details.

To overcome this problem, we introduce a pair of new “pooling” and “unpooling” operators, which we call *volume-preserving interleaving* and *de-interleaving layers*. The interleaving layer fully preserves input pixel information while reducing the spatial resolution, by correspondingly increasing the number of channels of the reduced-resolution output. Specifically, given input features with a resolution of (H, W, C) pixels, we divide each channel of the features to non-overlapping blocks of $r \times r$ pixels, and then flatten and stack the $r \times r$ pixels of each block into the channel dimension, resulting in output features with resolution $(\frac{H}{r}, \frac{W}{r}, C \times r^2)$.

The paired volume-preserving de-interleaving layer is the transpose of the interleaving layer, which reshapes the input features with resolution of $(\frac{H}{r}, \frac{W}{r}, C' \times r^2)$ to be (H, W, C') for the final full-resolution output of the network.

The hyper-parameter r is a positive integer, chosen to balance the quality and inference time for the applications. Note that although

this interleaving layer increases the number of channels in its output, as we vary r we have chosen to keep the number of channels in subsequent layers of the network fixed. So, for example, the next layer after interleaving will reduce the channels from $C \times r^2$ to some constant number of channels. Thus, the additional computation from the increased number of channels after interleaving affects only the next convolution layer, while reducing resolution at all other layers. Analysis of varying r is given in Section 5.1 — we generally find that performance slightly degrades with increased r , while computation time is substantially reduced.

We observe that the de-interleaving layer is equivalent to the pixel shuffling layer in [Shi et al. 2016]. While the pixel shuffling layer was originally only designed for upscaling images (e.g., for super-resolution problems where the input images are low resolution), we introduce the interleaving layer to intentionally downscale the input high-resolution images before feeding them into convolutional layers. As our contribution, we show that the paired interleaving and de-interleaving layers are effective, real-time alternatives to the classic pooling and unpooling layers. We report comparisons to several classic network architectures in Section 5.1.

3.1.2 Residual Blocks and Skip Connection.

The convolutional layers with residual blocks run at the downsized spatial resolution due to the previous interleaving layer. The number of residual blocks and the number of filters in each layer are selected to balance the quality and the inference time for the applications.

In addition to the residual blocks, the network concatenates its next-to-last layer with the input layer through a skip connection immediately before the last layer. This long-distance skip connection preserves high-frequency details in the imagery by allowing the original image to directly feed into the output layer.

In the following subsections, we demonstrate that our network can be trained and applied to various imagery tasks for computational displays, by only changing the input and output of the network. The different configurations of our network, with inputs, outputs, and loss functions, are summarized in Configurations 1-5.

3.2 Defocus Blur Rendering from RGB-D

As discussed in Section 2.1, rendering accurate defocus blur is required for properly driving accommodation in near eye displays. Existing methods (Section 2.2.2) for faithfully rendering defocus blur are either prohibitively computationally expensive or fail to approximate the blur effects at partially occluded regions.

Our method takes a single, all-in-focus RGB-D image as an input, which is typically available at no additional cost in any real-time

rendering engine, and generates a high-quality defocus blur image as an output. The method is summarized in Configuration 1.

Throughout the paper, we present depth values in units of inverse meters (diopters), abbreviated “D”. In addition to the color and depth maps, $(\mathbf{x}, \mathbf{d})$, we assist the network by also providing a circle of confusion (CoC) map $\mathbf{c}$. The circle of confusion is the shape of the blur on the retina, with diameter (roughly) proportional to the absolute difference (in diopters) between the image depth and the plane of focus. As a result, the CoC map also encodes the desired focus plane of the output. We give details on this CoC calculation below (Section 3.2.1). This feature is, in principle, learnable by the network, but by applying a simple per-pixel preprocessing, we can reduce the complexity of the network.

The output of the network is the defocus blur image at the focal distance provided by the input CoC image. The training loss is a weighted sum of the errors in the output intensity $\mathbf{y}$ and the image gradients $\nabla \mathbf{y}$, as given in Configuration 1. The target defocus image is represented by $\bar{\mathbf{y}}$. The network is trained to support focal distances ranging between 0D and 4D.

3.2.1 Circle of Confusion. To compute $\mathbf{c}$, we adopt the human eye model used in previous work [Matsuda et al. 2017; Mercier et al. 2017; Narain et al. 2015], which assumes a thin lens camera model with aperture diameter $A = 4\text{mm}$, distance between lens and film $s = 17\text{mm}$, and pixel size around 0.68 milliradian. The camera’s focal length f (depending on the focal distance q) and the CoC image $\mathbf{c}$ calculated per pixel are given by:

$$\mathbf{c} = A \left| \frac{f - \mathbf{d}}{f - q} - 1 \right|, \quad \text{s.t.} \quad f = q + \frac{1}{s}. \quad (1)$$

Configuration 1: Synthesize a defocus blur image from RGB-D.

Input: In-focus image $\mathbf{x}$, depth map $\mathbf{d}$, CoC map $\mathbf{c}$

Output: Defocus blur image $\mathbf{y}$

$\mathbf{y} = \text{DeepFocus}(\mathbf{x}, \mathbf{d}, \mathbf{c})$

Training loss $= \log_{10}(\|\mathbf{y} - \bar{\mathbf{y}}\|_1) + 0.5 \cdot \log_{10}(\|\nabla \mathbf{y} - \nabla \bar{\mathbf{y}}\|_1)$

3.3 Rendering for Multifocal Display

Multifocal displays represent 3D scenes by producing images with correct defocus blur effect on the retina when the viewer accommodates to different focal distances. Figure 3 illustrates an example multifocal system used in this paper following Narain et al. [2015], where four display planes are located at 0.2D, 0.8D, 1.4D and 2.0D, and the supported focal distances of viewers ranges from 0.1D to 2.2D continuously.

For multifocal displays, existing decomposition methods [Mercier et al. 2017; Narain et al. 2015] require a focal stack as input, i.e., a sequence of images focused at a range of depths. Assume the multifocal display has M display panels, where $\mathbf{y}_i$ is the image (display parameters) shown on the display panel located at depth p_i for $i = 1, \dots, M$. We aim to match a focal stack with N depths where $\mathbf{k}_{ij}$ is the point spread function of a pixel from display i on the focal plane q_j , and $\mathbf{z}_j$ the retinal image when the eye focuses at depth q_j for $j = 1, \dots, N$. The retinal image $\mathbf{z}_j$ is given by

$$\mathbf{z}_j = \sum_{i=1}^M \mathbf{k}_{ij} * \mathbf{y}_i, \quad \text{s.t.} \quad \mathbf{k}_{ij} = \text{circ} \left(A \left| \frac{q_j}{p_i} - 1 \right| \right), \quad (2)$$

Fig. 3. Viewing a multifocal display. The eye is depicted when focused near (left) and far (right). The images from the four display layers combine on the viewer’s retina to reproduce “near-correct” defocus blur, even when the viewer focuses between two layers.

where circ is the circular averaging kernel of the given diameter, and $*$ is 2D discrete convolution.

3.3.1 Decomposition from Focal Stacks. The key problem of multifocal display algorithms is to solve for the display parameters $\{\mathbf{y}_i\}$ given a target focal stack $\{\bar{\mathbf{z}}_j\}$. State-of-the-art methods [Mercier et al. 2017; Narain et al. 2015] optimize $\{\mathbf{y}_i\}$ by solving the following minimization problem:

$$\begin{aligned} \mathbf{y}_i = \underset{\mathbf{y}_i}{\text{argmin}} \quad & \sum_{j=1}^N \|\bar{\mathbf{z}}_j - \sum_{i=1}^M \mathbf{k}_{ij} * \mathbf{y}_i\|_2^2, \\ \text{s.t.} \quad & 0 \leq \mathbf{y}_i \leq 1, \quad i = 1, 2, \dots, M \end{aligned} \quad (3)$$

Instead of using computationally expensive iterative optimization to solve (3), we employ our network to directly produce the decomposition. Configuration 2 summarizes our method. We train the network to produce display images $\{\mathbf{y}_i\}$ from the input focal stack $\{\bar{\mathbf{z}}_j\}$. The range constraint is strictly satisfied by the use of the TanH activation function and rescaling at the last convolutional layer of the network as described in Section 4.

Note that unlike the loss function in Section 3.2 which penalized the network output directly, we penalize the recovered focal stack images instead (since this is what the user sees). We find that this penalization results in significantly higher quality results than penalizing the network output (the decomposition images) $\mathbf{y}_i$ directly, and importantly, it eliminates the need for generating ground truth decomposition images for training. Since the focal stack images $\mathbf{z}_j$ are linear in the display parameters $\mathbf{y}_i$ (via Equation (2)), we can still backpropagate through this loss function during training.

Configuration 2: Multilayer decomposition from a focal stack.

Input: Target focal stack $\{\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2, \dots, \bar{\mathbf{z}}_N\}$

Output: Display parameters $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$

$\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\} = \text{DeepFocus}(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2, \dots, \bar{\mathbf{z}}_N)$

Recovered focal stack $\mathbf{z}_j = \sum_{i=1}^M \mathbf{k}_{ij} * \mathbf{y}_i$

Training loss $= \sum_{j=1}^N (\log_{10}(\|\mathbf{z}_j - \bar{\mathbf{z}}_j\|_2^2) + 0.5 \cdot \log_{10}(\|\nabla \mathbf{z}_j - \nabla \bar{\mathbf{z}}_j\|_2^2))$

3.3.2 Focal Stack Rendering from RGB-D. Efficiently rendering the input focal stack involves not just the computational cost of producing an accurate defocus blur, but is multiplied by the number of images in the focal stack. For a focal stack of N depths, a naive approach would apply the network of Section 3.2 N times.

Instead, we are able to generate the entire focal stack using our network from RGB-D in a single pass. Configuration 3 summarizes our method. We extend the last layer of our network to have N

Fig. 4. Comparison of defocus blur rendering using RGB-D input only. We compare DeepFocus to prior methods that synthesize defocus blur from a single RGB-D input, including Unity game engine [2018], Nuke off-line renderer [The Foundry 2018], with focal distance 0.1D (top row) and 3.0D (bottom row). The top-right corner of each image shows the PSNR and structural similarity index (SSIM). DeepFocus (PSNR=45.6dB/46.1dB, SSIM=0.999/0.998) produces significantly more physically accurate blur than Unity (PSNR=25.3dB/27.3dB, SSIM=0.887/0.926), Nuke (PSNR=40.1dB/40.4dB, SSIM=0.993/0.994) and Nalbach et al. (PSNR=37.0dB/31.9dB, SSIM=0.989/0.980). Results with focal distance 0.1D and 3.0D are separated with a slash symbol. The reference defocus blur images are rendered with physically accurate, but off-line, accumulation buffering. The pixel-wise SSIMs are shown in Figure 5 for comparison. Additional results with more focal distances are included in the supplemental material.

Fig. 5. Pixel-wise structure similarity index (SSIM) of the defocus blur rendering methods compared in Figure 4. The top and bottom row show the results with 0.1D and 3.0D focal distance, respectively. DeepFocus significantly outperforms prior methods.

output images, each producing an image with fixed focal distance for each of the N equally spaced depths ranging between 0.1D and 2.2D. We no longer require the input CoC map, since the focal distances are fixed and the network learns separate mappings from depth to blur for each of the output channels.

3.3.3 Decomposition Directly from RGB-D. While existing multifocal decomposition methods require focal stacks as input, we can train our network to generate display parameters y_i directly from a single RGB-D image, combining both steps of focal stack rendering

Configuration 3: Synthesize focal stack from RGB-D.

Input: In-focus image x , depth map d

Output: Focal stack $\{z_1, z_2, \dots, z_N\}$

$\{z_1, z_2, \dots, z_N\} = \text{DeepFocus}(x, d)$

Training loss $= \sum_{j=1}^N (\log_{10}(\|z_j - \bar{z}_j\|_2^2) + 0.5 \cdot \log_{10}(\|\nabla z_j - \nabla \bar{z}_j\|_2^2))$

Configuration 4: Synthesize multifocal decomposition from RGB-D.

Input: In-focus image x , depth d

Output: Display parameters $\{y_1, y_2, \dots, y_M\}$

$\{y_1, y_2, \dots, y_M\} = \text{DeepFocus}(x, d)$

recovered focal stack $z_j = \sum_{i=1}^M k_{ij} * y_i$

Training loss $= \sum_{j=1}^N (\log_{10}(\|z_j - \bar{z}_j\|_2^2) + 0.5 \cdot \log_{10}(\|\nabla z_j - \nabla \bar{z}_j\|_2^2))$

and multifocal decomposition. Surprisingly, this does not require increasing the network capacity, which can be explained by the similar nature of the image decomposition process required in both tasks. Configuration 4 summarizes our inputs and outputs for this method.

3.4 Light Field Rendering from RGB-D

Light field displays require, as input, a large number M of *elemental images*, each of which is rendered from a distinct viewpoint. Rendering tens or even hundreds of views interactively is computationally

Fig. 6. Rendering and optimization for multifocal displays. The first four columns show optimized multilayer decompositions, from the furthest to the nearest display plane (see Figure 3 for the display construction). Simulated retinal images are reproduced in the right-most columns. DeepFocus (PSNR=45.6dB/43.4dB, SSIM=0.999/0.997) produces fewer errors than prior optimized blending [Narain et al. 2015] (PSNR=44.9dB/43.0dB, SSIM=0.998/0.996) and linear blending [Akeley et al. 2004] (PSNR=31.4dB/29.1dB, SSIM=0.941/0.909). Results with focal distance 0.8D (far) and 1.7D (near) are separated with a slash symbol.

Fig. 7. Synthesizing multiview imagery for near-eye light field displays. (a) A near-eye light field display based on Lanman and Luebke [2013]. (b) The input view positions (dark squares) for light field synthesis methods in Configuration 5. (c) The final image shown on the display panel generated by resampling the light field synthesized by DeepFocus. Additional explanation of the display setup is given in the supplementary video.

Configuration 5: Synthesize light field from sparse RGB-D inputs.

Input: Image $\mathbf{x}_i$ and depth map $\mathbf{d}_i$ at input views, $i = 1, 2, \dots, N$

Output: Elemental images $\mathbf{y}_j$ at novel views, $j = 1, 2, \dots, M$

$\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\} = \text{DeepFocus}(\mathbf{x}_1, \mathbf{d}_1, \mathbf{x}_2, \mathbf{d}_2, \dots, \mathbf{x}_N, \mathbf{d}_N)$

Training loss = $\sum_{j=1}^M (\log_{10}(\|\mathbf{y}_j - \bar{\mathbf{y}}_j\|_2^2) + 0.5 \cdot \log_{10}(\|\nabla \mathbf{y}_j - \nabla \bar{\mathbf{y}}_j\|_2^2))$

expensive. We use our network to significantly reduce the number of rendered views and synthesize the rest. In this section, we demonstrate that our network can efficiently synthesize all novel views, simultaneously, from a sparse set of RGB-D images.

The near-eye light field display setup is illustrated in Figure 7. The high resolution 2D images shown on the display panel are generated from the 4D light field. The image perceived by a viewer adapts with the viewer’s focal distance by superposing multiple images from the microlens array to approximate the desired blur.

As illustrated in Configuration 5, our method takes, as input, the RGB-D images rendered at a sparse set of views. We experiment with two circumstances: using the 5 views at the corners and center, and 9 views at the corners, center, and sides. The output layer produces the M elemental images directly. For the loss function, we penalize the error of the elemental images compared to the ground truth rendering of the scene from that viewpoint, which is a more demanding constraint than penalizing the final retinal images as perceived by the user.

4 DATASET AND IMPLEMENTATION

Each of the networks described in Section 3 is trained on large datasets generated by a physically based renderer. We create a procedural scene generator in Houdini [Side Effects 2018], which builds scenes with randomized object geometries and with randomized scales, textures, and materials. The objects are stochastically placed in the camera’s field of view to create a complex set of occlusions.

For the networks synthesizing defocus blur in Section 3.2, we rendered 4,900 focal stacks, each of which contains 40 defocus images with focal distances ranging between 0.1D and 4.0D, resulting in 196,000 pairs of RGB-D and ground truth defocus blur image samples. For the network synthesizing multilayer decompositions in Section 3.3, we rendered 5,390 focal stacks and RGB-D images, each of which contains 22 defocus images with focal distances ranging between 0.1D and 2.2D. For the network synthesizing light fields in Section 3.4, we rendered 4,165 light fields, each of which contains 81 RGB-D images at 9×9 views. Each frame has spatial resolution of 128×128 pixels. Approximately 80%, 10% and 10% of each dataset are used for training, validation and testing respectively.

Each convolutional layer of the network sequentially performs 2D convolutions with 3×3 filters, followed by batch normalization [Ioffe and Szegedy 2015], and then applies the exponential linear unit (ELU) activation function [Clevert et al. 2015], with the exception of the last convolutional layer. The last convolutional layer uses the hyperbolic tangent (TanH) activation function followed by a scaling by $f(x) = (x + 1)/2$ to bring the output within the range $[0, 1]$.

We train our networks with TensorFlow [Abadi et al. 2015]. The network weights are initialized following Glorot and Bengio [2010]. The Adam method [Kingma and Ba 2014] with recommended hyperparameters is used for optimization. Each network uses batch size of 16 with 400 epochs of training on the corresponding dataset, and takes 1-2 days on a GPU. After training, we optimize each network inference with 16-bit precision using the Nvidia TensorRT [2018] toolbox and evaluate its performance on a Nvidia Titan V GPU.

5 RESULTS

In this section, we analyze our method and compare to state-of-the-art approaches in three applications: varifocal, multifocal, and light field displays. For quantitative comparisons, we evaluate PSNR for pixel-wise accuracy, and structural similarity index (SSIM) [Wang et al. 2004] (as well as HDR-VDP-2 [Mantiuk et al. 2011] in the supplementary video) for perceptual image quality. Our work enables experiencing real-time rendered blur in the three major classes of near-eye displays for the first time. We leave thorough perceptual user studies for future work.

5.1 Network Complexity and Performance

In Table 1, we summarize the image quality and runtime of DeepFocus networks for each application. For synthesizing defocus blur (Configuration 1) and multilayer decomposition from RGB-D (Configuration 4), we train two versions of DeepFocus networks: a network with 12 layers that processes each color channel in a separate pass, and a faster network with 8 layers that takes all color channels as input and outputs color image results in a single pass, both with interleaving rate $r = 2$. The number of output channels of each convolutional layer is 128, with the exception of the last layer. Although the fast versions produce slightly degraded quality compared to the 12-layer versions (0.6-1.2dB drop in PSNR, 0.001-0.005 drop in SSIM), they are about 5 times faster to evaluate. For synthesizing dense light fields from RGB-D (Configuration 5), we train two networks with 12 and 8 layers respectively, both with interleaving rate $r = 2$ and processing color channels independently. The DeepFocus results reported in all other comparisons in the paper are from the 12-layer networks unless specified otherwise.

In our preliminary search for network architectures, we compared with encoder-decoder networks (“U-Net”) [Ronneberger et al. 2015] and dilated-convolution networks (“Dilated-Net”) [Yu and Koltun 2015; Zhang et al. 2017]. We set the network hyperparameters to have comparable or higher runtime than our DeepFocus networks, and then compare the result quality, taking the defocus blur rendering application (Configuration 1) as an example. The U-Net contains 3 scales and each scale contains 2 convolutional layers, resulting in 10 layers in total. The second layer at each scale, from fine to coarse, contains 128, 256 and 512 output channels respectively (except for

Fig. 8. PSNR (dB) and runtime (milliseconds) comparison of DeepFocus networks (12 layers) for defocus blur rendering, with varying interleaving rate r . In this experiment, the test image resolution is 512^2 . As r increases from 1 to 4, the network runtime reduces significantly (3-10 times), while the result PSNR drops only slightly (0.6-1.7dB).

Table 1. Average PSNR (dB), SSIM and runtime (milliseconds) for each network configuration, denoted by Cn , tested with 10-15 random scenes. Runtimes measured at image resolution 512^2 and 1024^2 are separated with a slash symbol.

Configuration and application	PSNR	SSIM	Runtime
C1: Defocus blur	42.1	0.992	15.0 / 45.9
C1: Defocus blur (fast)	41.5	0.991	3.23 / 9.82
C2: Multifocal from Focal Stack	43.3	0.996	15.4 / 50.3
C3: Focal stack from RGB-D	41.9	0.992	16.1 / 52.1
C4: Multifocal from RGB-D	40.2	0.988	14.6 / 46.2
C4: Multifocal from RGB-D (fast)	39.0	0.983	3.23 / 10.0
C5: Light field	33.7	0.956	19.7 / 65.4
C5: Light field (fast)	33.0	0.950	14.4 / 48.6

Table 2. Quantitative comparison with alternative network architectures for defocus blur rendering on 15 random scenes at image resolution 512^2 . Each scene contains 40 defocus images with focal distances between 0.1D and 4.0D. Average PSNR (dB), SSIM and runtime (milliseconds) are reported.

	U-Net	Dilated-Net	DeepFocus	Dilated-Net (fast)	DeepFocus (fast)
PSNR	41.7	41.5	42.1	38.8	41.5
SSIM	0.9910	0.9906	0.9920	0.9794	0.9906
Runtime	58.2	15.1	15.0	3.7	3.2

Table 3. Average PSNR (dB) and SSIM results of defocus blur rendering methods on 15 random scenes. Each scene contains 40 defocus images with focal distances between 0.1D and 4.0D.

	Nuke	Nalbach et al.	DeepFocus
PSNR	33.9	30.5	42.1
SSIM	0.970	0.955	0.992

the last layer of the network). In addition, the corresponding encoder and decoder layers are concatenated with a skip connection at each scale. This U-Net processes each color channel in a separate pass. For Dilated-Net, we train two versions which have close runtimes to the two versions of the DeepFocus network explained in the previous paragraph: a network with 5 layers that processes each color channel in a separate pass, and a faster network with 4 layers that takes all color channels as input and outputs color images in a single pass, both with 128 output channels at each layer (except for the last one). We choose the dilation rates of the two Dilated-Nets as 1,2,3,2,1 and 1,2,2,1 respectively, following the strategy by Zhang et al. [2017]. All other settings are selected to be the same as DeepFocus, including the activation function, training loss, training and test datasets, and inference optimization with TensorRT. The

Fig. 9. Defocus blur rendering using RGB-D input only. The top row, from left to right, shows the input RGB-D and DeepFocus results with focal distance 0.1D and 3.0D, respectively. The bottom rows show cropped regions from the input RGB-D, Unity (PSNR=35.8dB/35.9dB, SSIM=0.973/0.972), Nuke (PSNR=45.0dB/41.6dB, SSIM=0.997/0.994), Nalbach et al. (PSNR=40.1dB/37.9dB, SSIM=0.995/0.995) and DeepFocus (PSNR=51.3dB/50.5dB, SSIM=0.999/0.999). Results with focal distance 0.1D and 3.0D are separated with a slash symbol. Additional results with more focal distances are included in the supplemental material.

Fig. 10. Average PSNRs of DeepFocus results on 15 random scenes, for defocus blur (with and without MSA) and rendering focal stacks (Configuration 1 and 3, respectively), with the same network complexity. Each scene contains 40 defocus images with focal distances between 0.1D and 4.0D.

test results are reported in Table 2, which shows that DeepFocus outperforms U-Net and Dilated-Net in both quality and runtime.

In Figure 8 we report the performance of DeepFocus with varying interleaving rate r (Section 3.1.1), taking the defocus blur rendering application (Configuration 1) as an example. As r increases, the network runtime is reduced substantially (by a factor of 3 to 10, as r is increased from 1 to 4), while the resulting PSNR decreases slightly (by 0.6-1.7dB). This demonstrates the flexibility of our method: by adjusting the single hyperparameter r , the method can be tuned to support a varying trade-off between the quality and runtime.

5.2 Defocus blur

In Figure 4 and 9, we test the DeepFocus network on scenes from Unity, and compare to the built-in depth-of-field (DoF) rendering engine in Unity [2018], Nuke [The Foundry 2018] (a state-of-the-art off-line DoF renderer taking RGB-D as input), and the recent learning-based method by Nalbach et al. [2017]. The high-quality reference images are generated by the off-line accumulation buffer

Fig. 11. Average PSNRs of images generated by multilayer decomposition methods on 15 random scenes with focal distances between 0.1D and 2.2D.

Table 4. Quantitative comparisons of multilayer decomposition methods on 15 random scenes with average PSNR (dB) and SSIM values on all focal distances. Results without cropping image boundaries are in parentheses. Both Narain et al. and Mercier et al. are run with 100 iterations.

	Akeley et al.	Narain et al.	Mercier et al.	DeepFocus
PSNR	28.81 (28.39)	42.93 (38.04)	41.32 (37.33)	43.45 (43.31)
SSIM	0.8680	0.9956	0.9932	0.9963

Table 5. Runtime comparisons (seconds) of multilayer decomposition methods. Both Narain et al. and Mercier et al. are run with 100 iterations, using the GPU implementation of Mercier et al. [2017].

Resolution	Narain et al.	Mercier et al.	DeepFocus
512 ²	13.3	7.5	0.0154
1024 ²	62.0	25.0	0.0503

Table 6. Equal-time comparisons of multilayer decomposition methods on 15 scenes with average PSNR (dB) and SSIM values on all focal distances. For Narain et al. and Mercier et al., the results of a single iteration are reported.

	Akeley et al.	Narain et al.	Mercier et al.	DeepFocus
PSNR	28.81	35.20	27.56	43.45
SSIM	0.8680	0.9791	0.9296	0.9963

method implemented in Unity. In Figure 5, we compare the pixel-wise SSIM of each method. Our method produces more physically accurate blur and fewer visual artifacts from the same input, leading to better quantitative results. In Table 3, we further report quantitative comparisons on 15 random scenes with high depth complexity at focal distances uniformly sampled between 0.1D and 4.0D.

In Figure 10, we report the PSNR of DeepFocus at each focal distance, averaged over the 15 random scenes that are tested in Table 3. Our method achieves 41-43dB at all focal distances. Note that the PSNR decreases at far focus because missing regions in the RGB-D input (which occur near occlusion boundaries) become both disoccluded and in-focus, resulting in lower synthesis quality.

In addition, in Figure 10, we report our results taking multi-sample anti-aliasing (MSAA) depth input, i.e., the input depth map has 3×3 sub-pixel samples for each pixel color. Our network, specifically the volume-preserving interleaving layer, naturally supports such input by flattening and stacking the sub-pixel samples into the channel dimension as described in Section 3.1.1.

Finally, we evaluate DeepFocus on animation sequences rendered from Unity. Our methods produce high-quality results with no notable temporal artifacts. Comparisons to previous methods are included in the supplemental video, including for PSNR, SSIM, and HDR-VDP-2 metrics.

5.3 Multifocal

In Figure 10, we evaluate our method for focal stack rendering from RGB-D by reporting the PSNR values at varying focal distances on 15 random scenes, and compare to our previous network (with the same model complexity) synthesizing a single defocus image from RGB-D. The results are similar (less than 0.5dB difference), even though this network synthesizes all images in the focal stack simultaneously.

In Figure 6, we show the multilayer decomposition by our method and compare to Akeley et al. [2004] and Narain et al. [2015] for a scene with complex occlusions. Our learned decomposition is visually close to Narain et al. that is based on iterative optimization.

In Figure 11 and Table 4, we include quantitative comparisons with Akeley et al. [2004], Narain et al. [2015], and Mercier et al. [2017] at varying focal distances over 15 random scenes. We run both Narain et al. and Mercier et al. methods for 100 iterations as recommended in their original papers. Our method achieves higher quality at most focal distances. Since Narain et al. and Mercier et al. solve a deconvolution problem, boundary artifacts are introduced in their result images, while our networks learn to reduce errors at image boundaries. The results reported in Figure 11 exclude the boundary artifacts, and the results reported in Table 4 include cases with and without boundaries.

In Table 5, we compare the runtime of each method, on the GPU, with the implementation provided by Mercier et al. Our method is nearly 500× faster than Mercier et al., and around 1,000× faster than Narain et al. Note that the multifocal display setup adopted in this paper follows [Narain et al. 2015]. In order to support a larger focal range, it has higher system complexity than that of Mercier et al. [2017] in the number of display planes and input images, as well as the size of point spread functions k_{ij} in Equation (2).

Narain et al. and Mercier et al. require more time to finish a single iteration than the time for our network to process the entire frame. Therefore, in Table 6, we report equal runtime comparisons to their results, both run with a single iteration.

In Figure 12, we show results of our network generating multilayer decompositions directly from RGB-D. Our method outperforms Akeley et al., which is the only existing method that takes RGB-D as input. The results and comparisons for animated sequences are included in the supplemental video.

5.4 Near-eye light field

As illustrated in Figure 7, we assume a near-eye light field display with a 35-degree field of view, a lenslet pitch of 1.25mm, and a display image resolution of 2560^2 pixels. The target light field contains 9×9 views of elemental images with pixel size of 0.68 milliradian, whose view centers are uniformly distributed in a $1\text{cm} \times 1\text{cm}$ area.

We compare our method to Kalantari et al. [2016] and Wu et al. [2017], both taking sparse view RGB as input and interpolating novel views. In the original paper, Kalantari et al. synthesize each novel view individually from a sparse set of RGB images, by first estimating the depth (disparity) map at the novel view by a CNN, then warping all the input RGB images to the novel view, and finally feeding the warped images together with the estimated disparity map and view position to a second CNN to generate the novel view

Fig. 12. Real-time multilayer decomposition from RGB-D. DeepFocus (PSNR=42.7dB/44.8dB, SSIM=0.994/0.997) significantly outperforms Akeley et al. (PSNR=39.1dB/37.8dB, SSIM=0.984/0.988). Results with focal distance 0.1D and 2.2D are separated with a slash symbol.

image. Their CNNs were trained on a light field dataset captured by Lytro cameras. To make a fair comparison for Kalantari et al., we facilitate their method by warping the input RGB images according to the *rendered ground truth disparity map* at either input views (i.e., forward warping) or novel views (i.e., backward warping), as well as feeding the ground truth disparity map of each novel view to the second CNN, which is then retrained on the same dataset as our method. Note that the backward warping requires rendering depth maps for 72 novel views (81 views minus 9 inputs), which is impractical for real-time display applications. Consequently, we adopt forward warping in most comparisons with Kalantari et al., unless specified. To compare with Wu et al., we run the code provided by the authors and generate the results with tuned parameters.

In Figure 13, we compare the novel view images synthesized by each method with 9 input views on a test scene rendered in Unity. The maximum disparity between adjacent views in the reference light field is above 5 pixels. DeepFocus produces significantly better

Fig. 13. Comparison of synthesized novel view images at view (5,6) (see view grid in Figure 7). DeepFocus (PSNR=36.0dB, SSIM=0.983) produces comparable results to Kalantari et al. with forward pre-warp (PSNR=34.7, SSIM=0.985) and Kalantari et al. with backward pre-warp (PSNR=34.2, SSIM=0.984), both trained on the same dataset as DeepFocus and with additional accurate depth at novel views. Wu et al. produces inferior results (PSNR=22.5dB, SSIM=0.873). Additional results with more views are included in the supplemental material.

Fig. 14. Comparison of estimated retinal images at focal distance 0.5D (far-focus, top row) and 2.2D (near-focus, middle row) from the reference light field, and synthesized light field by Kalantari et al. and DeepFocus respectively. The artifacts in Kalantari et al. are highlighted by the orange arrows. Pixel-wise SSIM comparisons are given in Figure 15.

results than Wu et al., and comparable results to the augmented versions of Kalantari et al., as explained in previous paragraph, while the latter contain large-scale patch artifacts at object edges. The average PSNR and SSIM of all 72 novel views are reported in Table 7.

In Table 8, we report quantitative results of novel view synthesis on 10 random scenes from the Houdini scene generator. Note that DeepFocus generates the whole light field in a single network evaluation pass, and is about 18 times faster than Kalantari et al., which estimates each novel view in a separate pass. Both network inferences are optimized in TensorRT for this runtime comparison, and the warping process required in Kalantari et al. is excluded from its runtime measure. Note that Kalantari et al. requires 9×72 and

Fig. 15. Pixel-wise SSIM comparison of estimated retinal images at focal distance 0.5D (far-focus, top row) and 2.2D (near-focus, bottom row), by Kalantari et al. and DeepFocus using 9 and 5 input views respectively. The relatively high errors at the eye regions are due to the high sensitivity of the SSIM metric at black pixels, by definition.

Table 7. Average PSNR (dB) and SSIM by light field synthesis methods on all 72 novel views of the scene shown in Figure 13.

	Wu et al.	Kalantari et al. (forward)	Kalantari et al. (backward)	DeepFocus
PSNR	19.4	34.3	34.3	34.9
SSIM	0.806	0.981	0.983	0.978

Table 8. Quantitative comparisons of light field synthesis methods on 10 random scenes with average PSNR (dB), SSIM and runtime (milliseconds).

	Kalantari et al. (9 inputs)	Kalantari et al. (5 inputs)	DeepFocus (9 inputs)	DeepFocus (5 inputs)
PSNR	32.63	31.31	33.71	32.99
SSIM	0.960	0.951	0.956	0.950
Runtime	337.7	353.4	19.7	18.9

5×76 warps for the results with 9 and 5 inputs respectively, which could cause non-negligible runtime overhead.

In Figure 14 and 15, we compare the retinal images perceived through the near-eye light field display (Figure 7), computed from the light fields synthesized by DeepFocus and Kalantari et al. The

results demonstrate that DeepFocus outperforms Kalantari et al. both quantitatively and visually. Kalantari et al. produces patch artifacts at object edges (highlighted by the orange arrows), while DeepFocus produces a subtle noise that can be averaged out when the display image is perceived through the lens array. The results of animation sequences are included in the supplementary video.

In this paper, we focus on high-fidelity view interpolation from multi-view RGB-D inputs for real-time display applications. A related thread of view synthesis work, including that of Srinivasan et al. [2017], aims for (typically offline) view extrapolation from a single RGB image for light field photography applications. While the application topic and inputs differ, we have conducted a preliminary experiment comparing with these approaches. Specifically, we modify our DeepFocus network to take a single RGB image as input and retrain it with the dataset provided by Srinivasan et al. [2017]. Our retrained network produces inferior results to Srinivasan et al. This experiment suggests that, for light field photography applications, where multi-view images and accurate depth are unavailable, it may be helpful to further tailor the network design, e.g., to estimate the depth first and generate novel views guided by the estimated depth.

5.5 Limitations and Future Work

As a regression system, a neural network tends to gracefully degrade in quality outside of the known (trained) domain. We have highlighted typical artifacts and quality degradation throughout the paper by using error maps. With RGB-D input, specific failure cases include extrapolations with large disocclusions, and with semi-transparent and mirror objects where no single correct depth values exist. While such objects appear in the figures throughout the paper and supplementary video with plausible results, in Figure 16, we highlight a challenging case with semi-transparent glass. These failure cases require richer input information for physically accurate estimation and we leave this for future work. Other future work includes extending our framework to support chromatic eye aberration [Cholewiak et al. 2017], attenuation-based light field displays [Wetzstein et al. 2012], and holographic displays [Maimone et al. 2017].

6 CONCLUSION

In this paper, we have proposed a unified and flexible computational framework that is capable of producing imagery for a wide variety of accommodation-supporting displays. We have focused our investigations to three of the most well-researched variants: varifocal, multifocal, and light field displays. Together, these systems establish a commonality of rendering and optimization tasks, including synthesizing retinal defocus blur, multilayer decompositions, and novel views from sparse RGB-D light fields. Our resulting framework is based on a single convolutional architecture that allows reconfiguring inputs and outputs based on the needs of each particular display. The output image quality has been quantitatively shown to exceed existing methods that have been tailored for closely related tasks. By accepting common RGB-D inputs, our framework should present few barriers to adoption within existing rendering pipelines.

As computational displays, enabling practical applications of accommodation-supporting HMDs requires innovation beyond just

Fig. 16. An example failure case for defocus blur synthesis, where a statue is partially occluded by a piece of semi-transparent glass. At the overlapped region, there are no single correct depth values and only the foreground depths are returned by the rendering engine. With this limited RGB-D input, DeepFocus produces incorrect defocus blur at the overlapped region compared to the ground truth, for focal distance 2.0D.

the optical hardware. Algorithmic progress will be necessary to experience visually compelling interactive content. DeepFocus shows that these computational needs can be met in a generalizable manner, providing a foundation to overcome practical rendering and optimization limitations for future novel display systems.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments, and to thank Simon Green, Olivier Mercier, and Anton Sochenov for their help with experiments.

REFERENCES

- Martin Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. <https://www.tensorflow.org/>
- Kaan Akgit, Ward Lopes, Jonghyun Kim, Peter Shirley, and David Luebke. 2017. Near-eye Varifocal Augmented Reality Display Using See-through Screens. *ACM Trans. Graph.* 36, 6 (2017), 189:1–189:13.
- Kurt Akeley, Simon J. Watt, Ahna R. Girshick, and Martin S. Banks. 2004. A Stereo Display Prototype with Multiple Focal Distances. *ACM Trans. Graph.* 23, 3 (2004), 804–813.
- Brian A. Barsky and Todd J. Kosloff. 2008. Algorithms for Rendering Depth of Field Effects in Computer Graphics. In *Proc. ICCOMP*. 999–1010.
- Johannes Burge and Wilson S. Geisler. 2011. Optimal defocus estimation in individual natural images. *PNAS* 108 (2011), 16849–16854.
- Chakravarthy R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4 (2017), 98:1–98:12.
- Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Trans. Graph.* 32, 3 (2013), 30:1–30:12.
- Steven A. Cholewiak, Gordon S. Love, Pratul P. Srinivasan, Ren Ng, and Martin S. Banks. 2017. ChromaBlur: Rendering chromatic eye aberration improves accommodation and realism. *ACM Trans. Graph.* 36, 6 (2017), 210:1–210:12.
- Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).
- Robert L. Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed Ray Tracing. *SIGGRAPH Comput. Graph.* 18, 3 (1984), 137–145.
- Joe Demers. 2004. Depth of Field: A Survey of Techniques. In *GPU Gems*, Randima Fernando (Ed.). Addison-Wesley, 375–390.
- Piotr Didyk, Pitchaya Sitthi-Amorn, William Freeman, Frédo Durand, and Wojciech Matusik. 2013. Joint View Expansion and Filtering for Automultiscopic 3D Displays. *ACM Trans. Graph.* 32, 6 (2013), 221:1–221:8.
- David Dunn, Cary Tippetts, Kent Torell, Petr Kellnhofer, Kaan Akgit, Piotr Didyk, Karol Myszkowski, David Luebke, and Henry Fuchs. 2017. Wide Field Of View Varifocal Near-Eye Display Using See-Through Deformable Membrane Mirrors. *IEEE TVCG* 23, 4 (2017), 1322–1331.
- John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. DeepStereo: Learning to Predict New Views From the World's Imagery. In *Proc. CVPR*. 5515–5524.
- Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. Artificial Intelligence and Statistics*. 249–256.
- Paul Haeberli and Kurt Akeley. 1990. The Accumulation Buffer: Hardware Support for High-quality Rendering. *SIGGRAPH Comput. Graph.* 24, 4 (1990), 309–318.

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. CVPR*. 770–778.
- Robert T. Held, Emily A. Cooper, and Martin S. Banks. 2012. Blur and Disparity Are Complementary Cues to Depth. *Current Biology* 22 (2012). Issue 5.
- Robert T. Held, Emily A. Cooper, James F. O'Brien, and Martin S. Banks. 2010. Using Blur to Affect Perceived Distance and Size. *ACM Trans. Graph.* 29, 2 (2010), 19:1–19:16.
- David M. Hoffman, Ahna R. Girshick, Kurt Akeley, and Martin S. Banks. 2008. Vergence-accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of Vision* 8, 3 (2008), 33.
- Xinda Hu and Hong Hua. 2014. High-resolution optical see-through multi-focal-plane head-mounted display using freeform optics. *Optics Express* 22, 11 (2014).
- Hong Hua and Bahram Javidi. 2014. A 3D integral imaging optical see-through head-mounted display. *Optics Express* 22, 11 (2014).
- Fu-Chung Huang, Kevin Chen, and Gordon Wetzstein. 2015. The Light Field Stereoscope: Immersive Computer Graphics via Factored Near-eye Light Field Displays with Focus Cues. *ACM Trans. Graph.* 34, 4 (2015), 60:1–60:12.
- Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*. 448–456.
- Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-Based View Synthesis for Light Field Cameras. *ACM Trans. Graph.* 35, 6 (2016), 193:1–193:10.
- Petr Kellnhofer, Piotr Didyk, Szu-Po Wang, Pitchaya Sitthi-Amorn, William Freeman, Fredo Durand, and Wojciech Matusik. 2017. 3DTV at Home: Eulerian-Lagrangian Stereo-to-Multiview Conversion. *ACM Trans. Graph.* 36, 4 (2017), 146:1–146:13.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- Robert Konrad, Emily A. Cooper, and Gordon Wetzstein. 2016. Novel Optical Configurations for Virtual Reality: Evaluating User Preference and Performance with Focus-tunable and Monovision Near-eye Displays. In *Proc. SIGCHI*. 1211–1220.
- Robert Konrad, Nitish Padmanaban, Keenan Molner, Emily A. Cooper, and Gordon Wetzstein. 2017. Accommodation-invariant Computational Near-eye Displays. *ACM Trans. Graph.* 4 (2017), 88:1–88:12. Issue 36.
- George-Alex Koulouris, Bee Bui, Martin S Banks, and George Drettakis. 2017. Accommodation and comfort in head-mounted displays. *ACM Trans. Graph.* 36, 4 (2017), 87.
- Gregory Kramida. 2016. Resolving the Vergence-Accommodation Conflict in Head-Mounted Displays. *IEEE TVCG* 22, 7 (2016), 1912–1931.
- Jaroslav Krivánek, Jiří Žára, and Kadi Bouatouch. 2003. Fast Depth of Field Rendering with Surface Splatting. In *Proc. Computer Graphics International*. 196–201.
- Douglas Lanman and David Luebke. 2013. Near-Eye Light Field Displays. *ACM Trans. Graph.* 32, 6 (2013), 220:1–220:10.
- Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. 2009. Depth-of-field rendering with multiview synthesis. In *ACM Trans. Graph.*, Vol. 28. 134.
- Sungkil Lee, Gerard Jounghyun Kim, and Seungmoon Choi. 2008. Real-time depth-of-field rendering using point splatting on per-pixel layers. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1955–1962.
- Sheng Liu and Hong Hua. 2010. A systematic method for designing depth-fused multi-focal plane three-dimensional displays. *Optics express* 18, 11 (2010), 11562–11573.
- Gordon D. Love, David M. Hoffman, Philip J.W. Hands, James Gao, Andrew K. Kirby, and Martin S. Banks. 2009. High-speed switchable lens enables the development of a volumetric stereoscopic display. *Optics Express* 17, 18 (2009).
- Andrew Maimone, Andreas Georgiou, and Joel S Kollin. 2017. Holographic near-eye displays for virtual and augmented reality. *ACM Trans. Graph.* 36, 4 (2017), 85.
- Rafat Mantiuk, Kil Joong Kim, Allan G Rempel, and Wolfgang Heidrich. 2011. HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. In *ACM Trans. Graph.*, Vol. 30. 40.
- Belen Masia, Gordon Wetzstein, Piotr Didyk, and Diego Gutierrez. 2013. A survey on computational displays: Pushing the boundaries of optics, computation, and perception. *Computers & Graphics* 37, 8 (2013), 1012–1038.
- Nathan Matsuda, Alexander Fix, and Douglas Lanman. 2017. Focal Surface Displays. *ACM Trans. Graph.* 36, 4 (2017), 86.
- Michael Mauderer, Simone Conte, Miguel A. Nacenta, and Dhanraj Vishwanath. 2014. Depth Perception with Gaze-contingent Depth of Field. In *Proc. SIGCHI*. 217–226.
- Olivier Mercier, Yusuf Sulai, Kevin Mackenzie, Marina Zannoli, James Hillis, Derek Nowrouzezahrai, and Douglas Lanman. 2017. Fast Gaze-Contingent Optimal Decompositions for Multifocal Displays. *ACM Trans. Graph.* 36, 6 (2017), 237.
- O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel. 2017. Deep Shading: Convolutional Neural Networks for Screen Space Shading. *Comput. Graph. Forum* 36, 4 (2017), 65–78.
- Rahul Narain, Rachel A. Albert, Abdullah Bulbul, Gregory J. Ward, Martin S. Banks, and James F. O'Brien. 2015. Optimal Presentation of Imagery with Focus Cues on Multi-plane Displays. *ACM Trans. Graph.* 34, 4 (2015), 59:1–59:12.
- Mark A. A. Neil, Edward G. S. Paige, and Leon O. D. Sucharov. 1997. Spatial-light-modulator-based three-dimensional multiplanar display. *SPIE* 3012 (1997), 337–341.
- Nvidia Corporation. 2017–2018. TensorRT. <https://developer.nvidia.com/tensorrt>.
- Nitish Padmanaban, Robert Konrad, Tal Stramer, Emily A. Cooper, and Gordon Wetzstein. 2017. Optimizing virtual reality for all users through gaze-contingent and adaptive focus displays. *PNAS* 114, 9 (2017).
- Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2536–2544.
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 234–241.
- Kai Selgrad, Christian Reintges, Dominik Penk, Pascal Wagner, and Marc Stamminger. 2015. Real-time depth of field using multi-layer filtering. In *Proc. ISD*. ACM, 121–127.
- Lixin Shi, Haitham Hassanieh, Abe Davis, Dina Katabi, and Fredo Durand. 2014. Light field reconstruction using sparsity in the continuous fourier domain. *ACM Trans. Graph.* 34, 1 (2014), 12:1–12:13.
- Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proc. CVPR*. 1874–1883.
- Takashi Shibata, Joohwan Kim, David M. Hoffman, and Martin S. Banks. 2011. The zone of comfort: Predicting visual discomfort with stereo displays. *Journal of Vision* 11, 8 (2011), 11.
- M. Shinya. 1994. Post-filtering for Depth of Field Simulation with Ray Distribution Buffer. In *Proc. Graphics Interface*. 59–66.
- Shinichi Shiwa, Katsuyuki Omura, and Fumio Kishino. 1996. Proposal for a 3-D display with accommodative compensation: 3DDAC. *J. Soc. Information Display* 4, 4 (1996).
- Side Effects. 1996–2018. Houdini. <http://www.sidefx.com/>.
- Leonard M Smithline. 1974. Accommodative response to blur. *JOSA* 64, 11 (1974), 1512–1516.
- Pratul P. Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. 2017. Learning to Synthesize a 4D RGBD Light Field from a Single Image. In *Proc. ICCV*. 2262–2270.
- Qi Sun, Fu-Chung Huang, Joohwan Kim, Li-Yi Wei, David Luebke, and Arie Kaufman. 2017. Perceptually-guided Foveation for Light Field Displays. *ACM Trans. Graph.* 36, 6 (2017), 192:1–192:13.
- Unity Technologies. 2005–2018. Unity Engine. <http://unity3d.com>.
- The Foundry. 1993–2018. NUKE. <https://www.foundry.com/products/nuke>.
- Marc von Waldkirch. 2005. *Retinal projection displays for accommodation-insensitive viewing*. Ph.D. Dissertation. ETH Zurich.
- Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE TIP* 13, 4 (2004), 600–612.
- Sven Wanner and Bastian Goldluecke. 2014. Variational light field analysis for disparity estimation and super-resolution. *IEEE TPAMI* 36, 3 (2014), 606–619.
- Simon J. Watt, Kevin J. MacKenzie, and Louise Ryan. 2012. Real-world stereoscopic performance in multiple-focal-plane displays: How far apart should the image planes be? In *SPIE Stereoscopic Displays And Applications*, Vol. 8288.
- Gordon Wetzstein, Douglas Lanman, Matthew Hirsch, and Ramesh Raskar. 2012. Tensor displays: compressive light field synthesis using multilayer displays with directional backlighting. *ACM Trans. Graph.* 31, 4 (2012), 80:1–80:11.
- Sven Widmer, David Pajak, André Schulz, Kari Pulli, Jan Kautz, Michael Goesele, and David Luebke. 2015. An Adaptive Acceleration Structure for Screen-space Ray Tracing. In *Proc. High Performance Graphics*. 67–76.
- Gaochang Wu, Mandan Zhao, Liangyong Wang, Qionghai Dai, Tianyou Chai, and Yebin Liu. 2017. Light field reconstruction using deep convolutional network on EPI. In *Proc. CVPR*.
- Wanmin Wu, Patrick Llull, Ivana Tošić, Noah Bedard, Kathrin Berkner, and Nikhil Balram. 2016. Content-adaptive focus configuration for near-eye multi-focal displays. In *IEEE Multimedia and Expo*.
- Yang Yang, Haiting Lin, Zhan Yu, Sylvain Paris, and Jingyi Yu. 2016. Virtual DSLR: High Quality Dynamic Depth-of-Field Synthesis on Mobile Platforms. *Electronic Imaging* 2016, 18 (2016), 1–9.
- Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).
- Fisher Yu and Vladlen Koltun. 2016. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proc. ICLR*.
- M Zannoli, GD Love, R Narain, and MS Banks. 2016a. Blur and the perception of depth at occlusions. *Journal of vision* 16, 6 (2016), 17.
- Marina Zannoli, Gordon D. Love, Rahul Narain, and Martin S. Banks. 2016b. Blur and the perception of depth at occlusions. *Journal of Vision* 16, 6 (2016), 17.
- Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. 2017. Learning deep CNN denoiser prior for image restoration. In *Proc. CVPR*, Vol. 2.
- Zhoutong Zhang, Yebin Liu, and Qionghai Dai. 2015. Light field from micro-baseline image pair. In *Proc. CVPR*. 3800–3809.