2021

Light Field compression and manipulation via residual convolutional neural network

Eisa Hedayati
Michigan Technological University, hedayati@mtu.edu

LIGHT FIELD COMPRESSION AND MANIPULATION VIA RESIDUAL CONVOLUTIONAL NEURAL NETWORK

By

Eassa (Eisa) Hedayati

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Computational Science and Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2021

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Computational Science and Engineering.

Department of Electrical and Computer Engineering

Dissertation Advisor:	*Dr. Jeremy P. Bos*
Committee Member:	*Dr. Timothy C. Havens*
Committee Member:	*Dr. Warren F. Perger*
Committee Member:	*Dr. Scott A. Kuhl*
Department Chair:	*Dr. Glen Archer*

To my dear wife, father, mother, family and friends.

Contents

List of Figures

List of Tables

Acknowledgments

The list of people who have helped me to reach this point in my life is not exhaustive. First, I should thank my advisor Dr. Jeremy Bos, who has supported and guided me throughout this; I have been fortuitous to have him. My wife, Fatemeh, should be recognized here for the hours she spent listening to my presentations and proofreading my drafts of papers and conferences. Also, an especial thanks go to Dr. Daniel Fuhrmann, who had a crucial role in making my studies possible through his supports. Dr. Warren Perger also deserves recognition because of his financial support for part of my studies. Further, Dr. Timothy Havens should be specially recognized because of both his Financial help and academic advice which were essential for part of the projects done in this dissertation. I am also thankful to my committee for all of their helpful suggestion in improving my research. Moreover, I have been lucky to have the opportunity to study at Michigan Technological University that has a unique community that cannot be mirrored anywhere else. Finally, I want to thank my father and mother for all of their mental supports, without which I wouldn't be able to reach this point in my life.

List of Abbreviations

2-D	two-dimensional
3-D	three-dimensional
4-D	four-dimensional
BN	Batch Normalization
CA	Camera Arrays
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
dB	Decibel
DioF	Distance of Focus
DNN	Deep Neural Network
DoF	Depth Of Field
DSSIM	Structural Dissimilarity Index Metric
ELU	Exponential Linear Unit
FFT	Fast Fourier Transform
FLR	Fourier Slice Refocusing
FoV	Field of View
FPC	Focused Plenoptic Camera

fps	Frames per Second
FT	Fourier Transform
GB	Gigabyte
GPU	Graphics Processing Unit
HEVC	High Efficiency Video Coding
IN	Instance Normalization
JPEG	Joint Photography Experts Group
MAE	Mean Absolute Error
ML	Machine Learning
MLA	Microlens Array
MPSNR	Mean of Peak Signal-to-Noise Ratio
MSSIM	Mean Structural Similarity Index Metric
MSE	Mean Squared Error
NN	Neural Network
PC	Plenoptic Camera
PSNR	Peak Signal-to-Noise Ratio
RCNN	Residual Convolutional Neural Network
RELU	Rectified Linear Unit
SAIs	Sub-Aperture Images
SaS	Shift-and-Sum
SPC	Standard Plenoptic Camera

SSIM	Structural Similarity Index Metric

Abstract

Light field (LF) imaging has gained significant attention due to its recent success in microscopy, 3-dimensional (3D) displaying and rendering, augmented and virtual reality usage. Postprocessing of LF enables us to extract more information from a scene compared to traditional cameras. However, the use of LF is still a research novelty because of the current limitations in capturing high-resolution LF in all of its four dimensions. While researchers are actively improving methods of capturing high-resolution LF's, using simulation, it is possible to explore a high-quality captured LF's properties. The immediate concerns following the LF capture are its storage and processing time. A rich LF occupies a large chunk of memory —order of multiple gigabytes per LF—. Also, most feature extraction techniques associated with LF postprocessing involve multi-dimensional integration that requires access to the whole LF and is usually time-consuming.

Recent advancements in computer processing units made it possible to simulate realistic images using physical-based rendering software. In this work, at first, a transformation function is proposed for building a camera array (CA) to capture the same portion of LF from a scene that a standard plenoptic camera (SPC) can acquire. Using this transformation, LF simulation with similar properties as a plenoptic camera will become trivial in any rendering software.

Artificial intelligence (AI) and machine learning (ML) algorithms —when deployed on the new generation of GPUs— are faster than ever. It is possible to generate and train large networks with millions of trainable parameters to learn very complex features. Here, residual convolutional neural network (RCNN) structures are employed to build complex networks for compression and feature extraction from an LF. By combining state-of-the-art image compression and RCNN, I have created a compression pipeline. The proposed pipeline's bit per pixel (bpp) ratio is 0.0047 on average. I show that with a 1% compression time cost and 18x speedup for decompression, our methods reconstructed LFs have better structural similarity index metric (SSIM) and comparable peak signal-to-noise ratio (PSNR) compared to the state-of-the-art video compression techniques used to compress LFs. In the end, using RCNN, I created a network called RefNet, for extracting a group of 16 refocused images from a raw LF. The training parameters of the 16 LFs are set to $(\alpha = 0.125, 0.250, 0.375, ..., 2.0)$ for training. I show that RefNet is 134x faster than the state-of-the-art refocusing technique. The RefNet is also superior in color prediction compared to the state-of-the-art —Fourier slice and shift-and-sum— methods.

Chapter 1

Introduction

The *two dimensional* (2-D) images first might have been developed to capture memories. The plenoptic function [2], however, is the whole ray information of a scene, which is seven-dimensional

$$P = P(\theta, \phi, \lambda, t, V_x, V_y, V_z), \tag{1.1}$$

where θ and ϕ are the polar and azimuthal angles, λ stands for the wavelength, t is the time and V_x, V_y, V_z are the position of each point in space. Using "integral photography" [3], different slices as 2-D images can be extracted from a complete plenoptic function. But, obtaining complete plenoptic function is nearly impossible, at least by today's existing technology. The *light field* LF [4] has a *four dimensional*

4-D representation that provides a portion of the plenoptic function. These two additional dimensions make it possible to change the depth of focus, focus distance, and perspective in postprocessing. Also, limited occlusion removal is possible in captured LF [5] [6].

While the LF's idea has been developed in the early 1900s, capturing LF has always been challenging. The first method of LF capture was using a moving camera [4]. That method has been discontinued for two decades and substituted by *camera array* CAs [7, 8] or *plenoptic camera* PCs [9, 10]. Again, google re-introduced the moving cameras in capturing LF in 2018 [11]. But, the moving cameras are hard to use because the target should remain stationary for a long time.

As already mentioned, both arrays of cameras and PCs can be used for LF capture. Both methods have strengths and weaknesses. Among the available varieties of LF capturing methods, CAs win in terms of spatial resolution, *standard plenoptic camera* SPCs [12] are the winner of the angular resolution and *focused plenoptic camera* FPCs [10] are holding the middle lane. Therefore, the LF capture by an SPC has the least artifacts in extracted features by postprocessing. However, the spatial resolution in SPCs is inversely related to the number angular resolution. This problem, slowly pushed SPCs out of the current commercial market. There were exactly two commercialized SPCs and both have been discontinued. Hence, creating new and rich

datasets of this type of LF is now easy. This is where the simulation can help to overcome the problem while the development of technology might reintroduce optimized SPCs to the market.

Simulating PCs in any rendering software also has some limitations. In conventional rendering engines, creating an *microlens array* MLA will introduce some artifacts. In a computer, a curve is a combination of small straight lines [13]. Where any microlens in an MLA is already very small. Therefore, many leading rendering software comes short in creating high-quality MLAs. The first remedy to this problem is to think of a way to transform SPC into another type of camera where the need for MLA is nonexistent. If an SPC can be transformed to a CA in theory —where I derive this transformation as part of this dissertation—, the physical size of the cameras prevent the CA to be constructed in real-word. In the simulation, however, these constraints can be neglected. Therefore, it is possible to have a CA with both high angular and spatial resolutions when simulating. Such LF, nonetheless, suffers heavily from high space occupation.

A high-resolution LF can easily reach the vicinity of multiple *gigabyte* GBs. Therefore, a need for finding a data compression algorithm that can reduce LF's data size while keeping its features is evident. Another definition of the 4-D LF is essentially the collection of 2-D images taken from different angles from the same scene. Thus, I can expect a great amount of repeated or redundant information in the. Different LF

compression algorithms have been developed but fast and efficient LF compression is still an open problem. In general, there are lossy and lossless algorithms. In this dissertation, I am interested in lossy algorithms. The state-of-the-art video compression algorithms have been used on LF compression with great success [14, 15, 16]. However, all of these techniques are slow in both compression and reconstruction. None of them are suitable for real-time compression or LF video streaming. A fast and high-quality compression and reconstruction algorithm for LF is still an open problem. In case such a problem is solved, extracting interesting features such as refocusing from the LF is slow.

The problem of slow LF postprocessing is more evident in the case of high-resolution LFs. The two most widely used algorithms for refocusing are the *Fourier refocusing* FLR [17] and *shift-and-sum* SaS [7]. FLR has the pre-processing time complexity of $O(n^4 \log n)$ and for the case of SaS, each refocused image has $O(n^4)$ asymptotic time complexity. Therefore, both of them come short in real-time refocused image extraction. For being able to use LF in any real-time detection, we need to have an algorithm for fast LF feature extraction.

1.1 Thesis statement

In this dissertation, I observe that one important challenge in LF postprocessing is the excessive computation time if we have a high-resolution LF. This problem prevents the use of LF in many detection and estimation tasks, *e.g.* autonomous driving because they need real-time processing time. The main contribution of this work is to first create a paradigm for easy simulation of LF in any rendering software and then introduce ML algorithms to address the problem of long processing time. My proposed approaches show significant run-time reduction compared to the state-of-the-art methods.

1.2 Contributions

This dissertation makes the following contributions:

† I present a transformation function that can convert any SPC to a CA. I further show that using my transformation, simulation of PC-like capturing devices will be possible in any physically rendering machine with minimum requirements.

† I introduce a new compression algorithm for LF data compression that is significantly faster than the state-of-the-art LF compression techniques while having

similar quality in reconstruction.

† I present a novel ML technique for extracting a set of sixteen refocused images from a raw light field. I show that by employing learning-based methods, it is possible to obtain refocused images with better color predictions compared to the current state-of-the-art refocusing techniques. Also, this technique can be deployed on a conventional GPU with real-time runtime.

1.3 Outline

The remainder of this dissertation has been organized as follows. In chapter 2, I explain my proposed transformation function between an SPC and a CA. Then, in chapter 3, I present a novel combination of *machine learning* ML with JPEG for LF data compression, and I show how much it decreases the processing time compared to the state-of-the-art. In chapter 4, I show how ML is used to extract refocused images from raw LF. The contents of chapter 2 was published on July 1, 2020, in SPIE Optical Engineering as "Modeling standard plenoptic camera by an equivalent camera array" [18]. The content of chapter 3 has been accepted to publish in International Joint Conference on Neural Networks (IJCNN 2021) [19], and the content of chapter 4 has been submitted to a high profile peer-reviewed computer vision conference and is awaiting a decision [20]. Finally, chapter 5 concludes the dissertation by summarizing

my findings throughout my research and provides some possible future paths to be explored

Chapter 2

Modeling standard plenoptic camera by an equivalent camera array

2.1 Introduction

The light field (LF) is a catch-all term for ways of thinking about imaging, sensing, and displays that include both angular and spatial information. This modality stands in contrast to traditional imaging techniques where only spatial information is captured. Benefits of LF capture range from the artistic to the practical. For example, with

access to the full four-dimensional (4-D) LF, a photographer can change the depth of focus or focus distance in postprocessing. In traditional photography, these effects are introduced by changing the physical configuration of the camera and cannot be changed after the picture is taken. Practically, LF imaging also allows for limited-removal of occlusions [5] [6].

Lippman [3] was the first to suggest the idea of LF capture as "integral photography". However, it was the advent of computer graphics and digital image processing [4] that gave the idea substance. In this new incarnation, the LF was referred to as the plenoptic function [2] and often used camera arrays (CA) [8] for LF capture. Work in the area of plenoptic imaging and LF rendering remained mostly a research novelty until Ng *et al.* demonstrated both a hand-held plenoptic camera (PC) [9] and a way of quickly performing refocusing and other postprocessing using Fourier slice photography [17]. Since that time, work in the area of LF technologies has expanded to include LF displays used in augmented and virtual reality systems [21]. Here, I am interested in modeling different methods of LF capture. As already mentioned, both arrays of cameras and Ng *et al.*'s PC can be used for LF capture. Both methods have strengths and weaknesses. Apart from issues of cost and size, CAs are also severely constrained in their ability to sample the angle space leading to postprocessing artifacts. Ng *et al.*'s PC, also called plenoptic camera 1.0, or the standard plenoptic camera (SPC) by Perwass and Wietzke, [12] is generally cheaper and smaller in size compared to a CA. In an SPC configuration, a microlens array (MLA) is placed at the

focus of the main lens. The f-number of the main lens and MLA elements are also matched, and the imaging sensor is placed at one focal distance from the MLA. This arrangement has the benefit of maximizing the relative angular and spatial sampling and the use of the imaging sensor. A main drawback of the SPC is that the available spatial resolution is limited to the number of MLA elements.

For example, images from an SPC with an $M \times M$ MLA and $M = 256$ would have a resolution of 256×256, which is small by the standard of today's digital cameras. At the same time, the angular resolution is set by the sampling of the image formed under each microlens element; often called a "microlens image." If this image is sampled by an array of 10×10 pixels on the imaging sensor, and assuming no "dead" or "masked" pixels, the pixel count quickly approaches millions of pixels. In response to this limitation, Georgiev and Lumsdaine [10] have suggested the focused PC. His design effectively images from within the aperture creating several overlapping Keplerian telescopes (or Galilean depending on the placement of the microlens with respect to the main lens focal distance). This approach provides improved spatial resolution at the expense of angular sampling. Perwass and Wietzke [12] introduced another version of the PC that does not fall in the category of earlier PCs. They proposed to increase the lateral resolution by spreading microlenses with different focal distances in the MLA with a specific pattern. Anisimov *et al.* [22] developed an LF camera by combining CAs and PCs. Their camera has an array of lenses instead of the single objective lens of the cameras in the array.

While the theoretical model of SPC in any configuration is simple, the practical construction of these devices is difficult. Practical devices require careful calibration and suffer from a variety of postprocessing issues and other artifacts. Still, the fact that these devices exist and are marketed commercially is evidence that the problems are surmountable. The varieties of PC's exist because each can deliver some unique benefits over others by compromising some other properties. Therefore, one should consider the relative trade-offs in spatial/angular resolutions when choosing one configuration over another. However, say I have a problem I want to solve but are unsure what PC approach is most appropriate. One approach would be to simulate a "perfect" LF and introduce practical effects. Here, it worth iterating some of the works performed in simulating LF.

LF simulation is currently an active area of research. Marwah *et al*. [23] used Blender to simulate LFs for the demonstration of their compressive LF model. Honauer *et al*. [24] created a simulated LF dataset by creating a CA in Blender. Michels *et al*. [13] created a PC model in Blender by creating a detailed MLA model. Their camera is designed in such a way that the position of the MLA can be changed to simulate different SPC configurations. Michels *et al*.'s work shows a fundamental problem of modeling SPC in Blender with simulated physical elements. In Blender, each curve is constructed by straight lines between vertices. Because each microlens is very small and the number of vertices is limited, creating accurate models of these physical elements is difficult and time consuming.

One common way of postprocessing SPC LFs is to generate viewpoint images from within the aperture. Conceptually, this suggests that one simple way of simulating an LF would be to place cameras in the world such that they capture these viewpoint images as a CA [25]. However, selecting the correct location, pose, and camera characteristics in order to correctly model the SPC is not obvious. Dansereau suggested placing the CA one focal distance away from its objective lens [26]. Hahne later revealed an error in Dansereau's model [27], showing instead that the cameras should be centered on the main lens of the desired SPC. Hahne used the observed distance of focus (DioF) of in depth of field (DoF) images to demonstrate the accuracy of his model and the resulting improvement [28]. The configuration of each camera element is also not obvious. While the spatial resolution of the cameras should be equal to the number of MLA elements, other parameters, such as the aperture size and field of view (FoV), are unclear but needed to set up a simulation model.

In this chapter, I introduce a transformation between the SPC 1.0 and and equivalent CA, including the exact position, FoV, and aperture size for each camera. I model an SPC 1.0 camera with the main lens with a 100-mm focal length and $f/2$ modeled as a thin lens. The MLA is placed at the focal distance of the objective lens and has the same f-number. The MLA has 500×500 elements. Each microlens image is sampled by 25×25 pixels. I show the equivalency of these two systems by first simulating a pristine LF using the Blender software [29] in the CA configuration. Next, I compare the CA LF's DoF focus distances to those predicted by Hahne's

Plenoptisign software [30]. Finally, I show how, starting with the simulated, pristine LF, practical effects such as vignetting can quickly and easily applied to the entire LF as a 4-D data object. The effects of angular downsampling and vignetting at the main lens and MLA on refocus distance are also discussed.

In Sec. 2.2, I provide some background on LFs, LF cameras, the techniques used for postprocessing the LF to create images, the CA to SPC transformation, and how simple practical effects may be added. In Sec. 2.3, I describe our virtual CA model and simulation set-up in Blender. Section 2.4 provides some example results. Conclusions and directions for future work are outlined in Sec. 2.5.

2.2 Background

2.2.1 The Light Field

In the modern context, the LF or plenoptic function is a ray optics simplification of the visible portion of the electromagnetic field intensity at all points in space [2]. For a single wavelength and optical axis, this function can be described in four dimensions. The most straightforward way of thinking of the LF is to place a plane perpendicular to the optical axis described by the spatial coordinates (x, y). The other two dimensions measure the incident rays from all directions (θ, ϕ). A more common

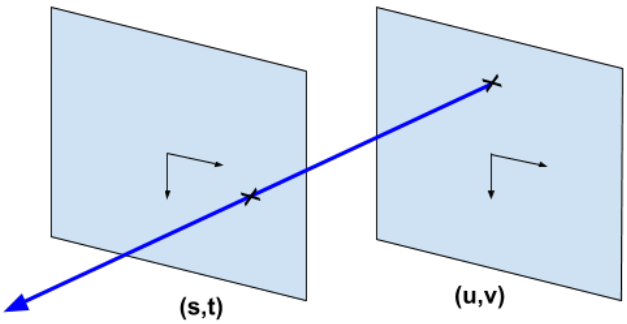


Figure 2.1: Two-plane parameterization of the LF as popularized by Levoy.

representation found in the literature uses a two-plane parameterization popularized by Levoy and Hanrahan [4]. Here, as shown in Fig. 2.1, the LF is described by the intersection of rays with the (s, t) plane, which represents the spatial dimensions and the (u, v) plane responsible for the angular dimensions. Thus, the LF on a given optical axis is described by the function $L(s, t, u, v)$. According to Ng *et al.* [9], the conventional camera's image is derived from the LF function as

$$I(s,t) = \frac{1}{D^2} \iint L(s,t,u,v) A(u,v) \cos^4 \theta du\, dv, \tag{2.1}$$

D is the distance between sensor plane and aperture, and the aperture function is referred as $A(u, v)$. θ is the angle between the incident rays and the sensor. The Eq.(2.1) directly suggests that by changing the aperture function A, different DoF images will be extracted. The pinhole image ($I_{pinhole}$) at each view point (u', v') is

extracted as

$$I_{u',v'}(s,t) = L(s,t,u=u',v=v'). \tag{2.2}$$

Though other processing techniques such as depth estimation [31] and partial occlusion removal [6] exist, the last one I will expound upon here is image refocusing. The LF can be refocused by integrating along a slope defined by dividing the focal depth, FD (the distance of the real image of the scene behind the objective lens), by the focal length of the main lens, F or $\alpha = \frac{FD}{F}$. Specifically,

$$I_{FD}(s,t) = \iint L(u + \frac{s-u}{\alpha}, v + \frac{t-v}{\alpha}, u, v) A(u,v) \cos^4\theta du\, dv. \tag{2.3}$$

2.2.2 Methods of LF Capture

So far, I have described a method by which a user may postprocess an LF to produce certain effects. Missing are methods for actually capturing the LF and their practical limitations. While a full discussion is beyond the scope of this paper, I begin by providing a brief overview of Ng *et al.*'s SPC [9] and the CA.

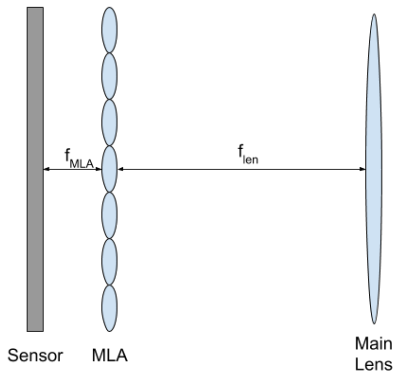


Figure 2.2: Configuration of an SPC

2.2.2.1 Standard Plenoptic Camera

As shown in Fig.2.2 the configuration of the SPC involves a primary lens, L, an MLA placed at f_{len}, and the detector at f_{MLA}. The MLA is assumed to have $M \times M$ elements. Likewise, $N \times N$ sensor elements, or pixels, are arranged under each MLA element. In this configuration, there are N^2 viewpoint images with spatial resolution $M \times M$. So, I see that the number of MLA elements limits the spatial sampling, whereas the angular sampling is limited by the number of samples under each MLA element. In this chapter, I will follow the convention of referring to the images formed under each MLA element as "microlens" images. Here, the LF is sampled as $L(u, s)$ where $u = -\frac{N}{2}$ to $\frac{N}{2} - 1$ and $s = -\frac{M}{2}$ to $\frac{M}{2} - 1$.

In this configuration, the image sampled by the SPC sensor has $(M \times N, M \times N)$

samples. To extract the 4-D LF from the plenoptic camera's 2-D sensor (L_p)

$$L(s,t,u,v) = L_p\Big(u + \big((s-1) \times N\big), v + \big((t-1) \times N\big)\Big), \tag{2.4}$$

s, t are in range $(1, M)$ and u, v are in range $(1, N)$.

2.2.2.2 Camera Array

CAs also been used for LF capture [8]. Indeed, they were the first devices applied to the task. In this case, I assume an $N \times N$ CA and that each camera captures an image with $M \times M$ spatial resolution. Each camera is assumed to have an identical configuration. The use of the same notation is intentional to make clear both the similarities and differences between the two methods. Ignoring diffraction, the spatial resolution of the CA is limited by the pixel pitch of the camera imaging system. The angular resolution, however, is governed by the pitch of the CA. Immediately, I can see a practical weakness of this approach: the minimum separation (pitch) between camera centers is limited by the size of the aperture support equipment (the camera or the sensor's dimension, *etc.*). For this reason, LFs captured using a CA tend to be severely undersampled in the angle space.

2.2.2.3 Practical Issues Concerning LF Capture: Calibration

Another issue facing both LF capture techniques is calibration. As alluded to above, the SPC captures the LF as a series of microlens images sampled under each MLA element. Practically, images under MLA elements that are farther from the optical axis will be displaced rather than falling directly under each element partially due to the thickness of the main lens. Dansereau calculated parts of these displacements and called it directionally dependent radial distortion [32]. Even on-axis, small misalignments can result in uncertainty in the location of the microlens image in pixel space. Though an aspect of Ng *et al.*'s work [9], Dansereau [32] first published techniques for calibrating the SPC. Measuring the pose of each CA element introduces an entirely different set of calibration challenges. These challenges are arguably more difficult to address. Without going into detail, calibration requires one to know precisely the relative orientation of each camera to another in six dimensions (three spatial dimensions plus roll, pitch, and yaw).

2.2.3 Transformation of Camera Array to Standard Plenoptic Camera

From the discussion above, both the CA and the SPC can be used for LF capture. Here, I am concerned with simulating the LF. From a simulation perspective, the CA is particularly appealing. For example, in simulation, I can have overlapping apertures or sensors. I can also use a series of pinhole cameras without concern regarding the available light. Intuitively, based on the discussion above, it also follows that for any given SPC setup there is an equivalent CA. In the simulation, I can also perfectly control CA's geometry. Our goal is to create a CA that can record with the same spatial and angular sampling as a given SPC. To do this, I need to specify the correct placement and orientation of each camera in the array.

To create a transformation from SPC to CA, I need to know the number of cameras in the array and two general quantities: the properties of each camera in the array and the placement of each relative to the common optical axis of both devices. The important properties of each camera are the aperture size, the FoV, and the spatial resolution. I also need to know the precise six degrees of freedom pose of the CA and each camera in it relative to the main optical axis, where the optical axis is defined to be the optical axis of the SPC's objective lens. To correctly find these quantities, I trace the light rays inside an SPC by employing geometrical optics rules

as demonstrated in Fig. 2.3(a).

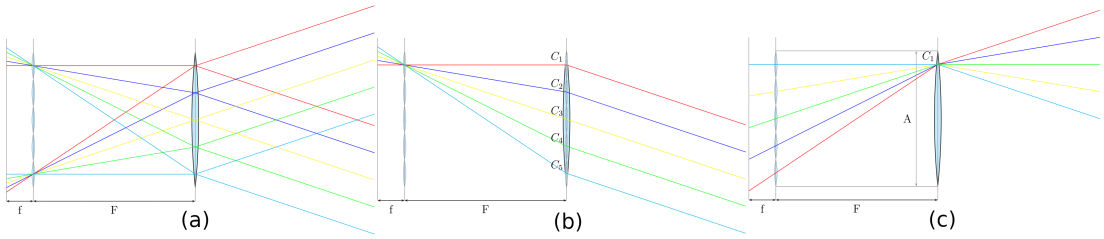


Figure 2.3: Using geometric optics ray tracing principles to understand the equivalence of the SPC and CA. (a) The rays of the microlens' at the boundaries of the MLA. (b) Rays traced from each pixel under a microlens through the main lens. (c) Rays traced from a proposed CA element center through each element in the MLA.

Looking at either the top or bottom microlens in Fig. 2.3(a), I see that the microlens separates the rays coming from different positions on the main lens. Rays with the same color intersect at the camera locations at the main lens and correspond to viewpoint images available from the LF. As I mentioned in Sec. 2.1, I have specified these locations and I must also specify the properties of each element in the CA. These properties are minimum FoV, the aperture, and minimum pixel resolution or maximum pixel pitch of the image sensor of each camera in the array.

2.2.3.1 Transformation of Camera Array Properties from a SPC Model

Figure 2.3(b) shows the limitation of FoV for the top camera in the array, which demonstrates that no light outside of the given boundaries in 2.3(a) can be recorded

by SPC. Therefore, the FoV of the top camera is governed by the same limitation and will be

$$FoV_{C_1} = 2 \times \arctan\left(\frac{A}{2F}\right), \tag{2.5}$$

where A is the aperture size and F is the objective lens's focal distance of the SPC. As shown in Fig. 2.3(a) all of the cameras in the array have the same FoV. Therefore, Eq. (2.5) holds for all of the cameras in the array.

The number of pixels under each microlens governs the number of cameras in the array. Figure 2.3(c) shows that each pixel under a microlens should be captured by a different camera. Therefore, the number of cameras in the array is equal to $N \times N$. Because the pixels in the sensor have similar size, using the rules of similar triangles I find that the aperture is equally divided and each pixel integrates the irradiance of one of the divisions. Thus, the size of each camera's aperture is calculated by

$$A_{ca} = \frac{A}{N \times N}, \tag{2.6}$$

where A_{ca} is the aperture size of each camera in the array, A represents the SPC's main aperture size.

Finally, the resolution or pixel pitch on each camera's sensor is governed by the number of microlenses in the MLA ($M \times M$). Each camera in the array should have the resolution equal to ($M \times M$).

2.2.3.2 Constructing LF

Our goal is to obtain the same 4-D sampled LF or $L(s,t,u,v)$ from the CA as its corresponding SPC. In the CA, each camera is sampling the s,t plane while u,v denotes the plane where the cameras in the CA are placed. Therefore, the correct order of filling the $L(s,t,u,v)$ matrix is

$$L(s,t,u,v) = I_{C_{u,v}}(s,t), \tag{2.7}$$

$I_{C_{u,v}}(s,t)$ is the captured image by each camera positioned at (u,v).

2.2.4 Adding Practical Effects

In any optical imaging system, from telescopes to conventional cameras, where there are lenses and mirrors involved, there will be practical limitations, such as vignetting, turbulence, aberrations, *etc*. Such effects will also be found in an SPC that may consist of any number of optical elements in addition to the main lens and MLA. Rather than try to model these effects as part of the simulation, my approach is to add them via postprocessing to a pristine LF with the desired sampling.

Here, I use vignetting as an example of how practical effects can be introduced to

a pristine LF. I use the Hadamard product (elementwise product [33]) scaling the intensity of each pixel in the LF as

$$L_r = R \circ L. \tag{2.8}$$

In Eq. (2.8) $\circ$ denotes the Hadamard product, L_r is the LF with added effects, R is any mask that can be expressed by scalar multiplication of positive real numbers, and L is the pristine LF.

2.2.4.1 Main Lens and MLA Mask creation

We model vignetting using functions $\mathcal{F}$ that operate either on the entire LF or a subset of the LF usually a microlens image. In the case of main lens vignetting, the vignetting effect applies to the entire LF as

$$R'_M = \mathcal{F}_M(J_{s\times u, t\times v}), \tag{2.9}$$

where $\mathcal{F}_M$ is the desired vignetting effect, J is the matrix of ones of size $(s \times u, t \times v)$ [33], and R'_M is a mask that has similar size to the SPC's sensor. I use Eq. (2.4) to transform the 2-D R'_M to the 4-D R_M which can be applied directly the 4-D LF as outline in Eq.(2.8) without requiring reshaping the LF.

For vignetting introduced by the MLA, the mask, R_{MLA} is applied instead to each microlens image

$$R_{MLA_{s,t}} = \mathcal{F}_{\mathcal{MLA}}(J_{u,v}), \tag{2.10}$$

where $\mathcal{F}_{\mathcal{MLA}}$ is vignetting for each microlens (s,t). Again, these masks can be applied by Eq. (2.8) to the pristine LF as a whole rather than operating on each microlens image.

2.2.5 Vignetting

We use a simple two parameter vignetting model for both main aperture and MLA vignetting described using the piecewise function, F, as

$$F(d,r,a) = \begin{cases} 0 & d > r + a \\ 1 & d < r \\ 1 - \left(\frac{d-r}{a}\right)^2 & r \leq d \leq r + a \end{cases}, \tag{2.11}$$

where d is the distance of each pixel from the middle of the image sensor. r represents the main lens aperture radius and a is the fading attenuation distance.

2.3 Method

We used Blender to render two scenes based on SPC 1.0 using a thin lens approximation and verified our generated LF's similarity to the converted SPC's LF by comparing the refocusing distance of the extracted DoF from the simulated LF with the expected DioF derived from Hahne's method [34] for the desired SPC and applied simple vignetting effects by postprocessing.

In Blender, I used Cycle [35] a physically based ray tracing engine. I used the Blender add-on tool generated by Honauer *et al.* [24] on our desired scenes to create the camera grids. I used the infinite focus option in the tool to keep the camera grid in agreement with our described conversion.

2.3.1 Camera Array Configurations

We use one camera configuration in Blender for simulating two scenes. The CA in our simulation model consists of 25×25 cameras. Each camera has an aperture of 2 mm and a focal distance of 4 mm and a resolution of 500×500. This configuration, according to Sec. 2.2.3, will transform to an SPC with an aperture of 50 mm, main lens focal distance of 100 mm, MLA of 500×500, and 25×25 pixels under each

microlens.

For each simulated scene, I also generated two downsampled LFs: one to 12×12 and another to 5×5 angular samples. These LFs are used for demonstrating the effects of vignetting and to validate our transform model, respectively. The corresponding SPCs to these two downsampled models have the same aperture size, 50 mm, main lens focal distance, 100 mm, and MLA elements 500×500 but with the specified microlens image sizes. Downsampling is accomplished by averaging adjacent view point images.

2.3.2 Simulated Scenes

For this work, I simulated two different scenes. The first is a line of wooden seahorses in a black background. This scene will be useful for model validation. The second is a realistic scene of a pavilion in Barcelona [36]. This scene demonstrates that our model can be used on realistic scenes and to better demonstrate practical effects, vignetting in this case.

2.3.3 Transformation Validation

We demonstrate that our transformation from CA to SPC is valid by comparing the DioF of the LF obtained by our CA simulated in Blender and the estimated DioF by Hahne's Plenoptisign [30] software. Hahne's software is capable of returning the near and far DioF boarders and DoF for a given α by entering the physical properties of the SPC of interest. Therefore, to test the correctness of our transformation, I fed the properties of our transformed SPC driven from Sec. 2.2.3 to Plenoptisign for $\alpha = 0.1, 0.2, 0.3, 0.4$ and extracted the DioF values.

To validate our model, I constructed a scene made up of 10 seahorse models [37] and numbers from 0 to 9 adjacent to each seahorse. Each number indicates the distance of each seahorse from the Cartesian coordinate origin of the Blender model. The center of the CA is located at $(x, y, z) = (0, -5, 1.5)$ in the simulation world and units are in meters. The CA optical axis is oriented along the positive x direction. Figure 2.4 (a) shows the center view simulated by the middle camera in the array. Given the placement of the CA, the zeroth seahorse is $\sim$ 5.2 m away from the center of our CA. The distance to each seahorse is then the number adjacent to the seahorse plus 5.2. Seahorses in our model are separated by 1 m. Therefore, all distance estimates have a relative uncertainty of $\pm$0.5 m. For each value of alpha, I evaluate the DioF and DoF by examining the range of seahorses that are in focus and compare them to

Plenoptisign estimates.

(a) (b)

Figure 2.4: The Blender rendered models. (a) The middle view of the seahorse scene. (b) The middle view of the Barcelona Pavilion scene.

2.3.4 Application of Vignetting

While the seahorse scene is valuable for demonstrating the DioF, it is difficult to observe vignetting effects because of the black background. To better visualize vignetting, I simulate a scene called "Barcelona Pavilion" architected by Mies in Blender [36]. In this model, the CA's optical center is at $(x, y, z) = (-25, -5, 2)$ and the array is looking at the main structure of the pavilion. Figure 2.4(b) is the center view of our simulated LF. I explore the effects of vignetting on the DoF images extracted from the pristine and vignetted LFs.

2.4 Results and Discussion

In this section, I first demonstrate the validity of our transformation. Then, I illustrate the introduction of practical effects via postprocessing (vignetting in this case).

2.4.1 Validation

In this section, our goal is to show agreement of our simulated DoF's DioF with those indicated by the Plenoptisign software.

In Fig. 2.5, I present DoF images for four different values of the refocusing parameter, α. Images were extracted from the simulated LF with a 25×25 angular samples. Referring to Table 2.1, I can see that for $\alpha = 0.33$ the DoF predicted by Plenoptisign is 2.4 with the near DioF border at 5.7 m. In corresponding subfigure, Fig. 2.5(c), the second and third seahorses are sharpest, and the fourth seahorse is still a little sharper than others. The distances of these two seahorses are 6.2 and 7.2 m, respectively, and the fourth seahorse is located at 8.2 m away from the CA. This agrees with the expected DioF's extracted from the Plenoptisign software and shows that our model results are within our measurement uncertainty. Other α's results shown in Fig. 2.5 are also within the boundary of the estimated error compared to Table 2.1.

One of the benefits of simulating an LF is I can easily change sampling by down-sampling a high-resolution LF. The effect of angular resolution on the LF and LF processing can be unintuitive. Therefore, in building an LF capture system, it is useful to understand how much resolution is necessary to create the desired effect or extract relevant information. In Fig. 2.6, I show two DoF images with the same value of the refocusing parameter, α but with two different angular sampling rates. Referring to Table 2.1, I see that all of the DoF properties have changed. These changes are also obvious in the figure where the furthest seahorses are in focus for the lower angular resolution image. In contrast only the closest seahorse is in focus for the higher angular resolution image.

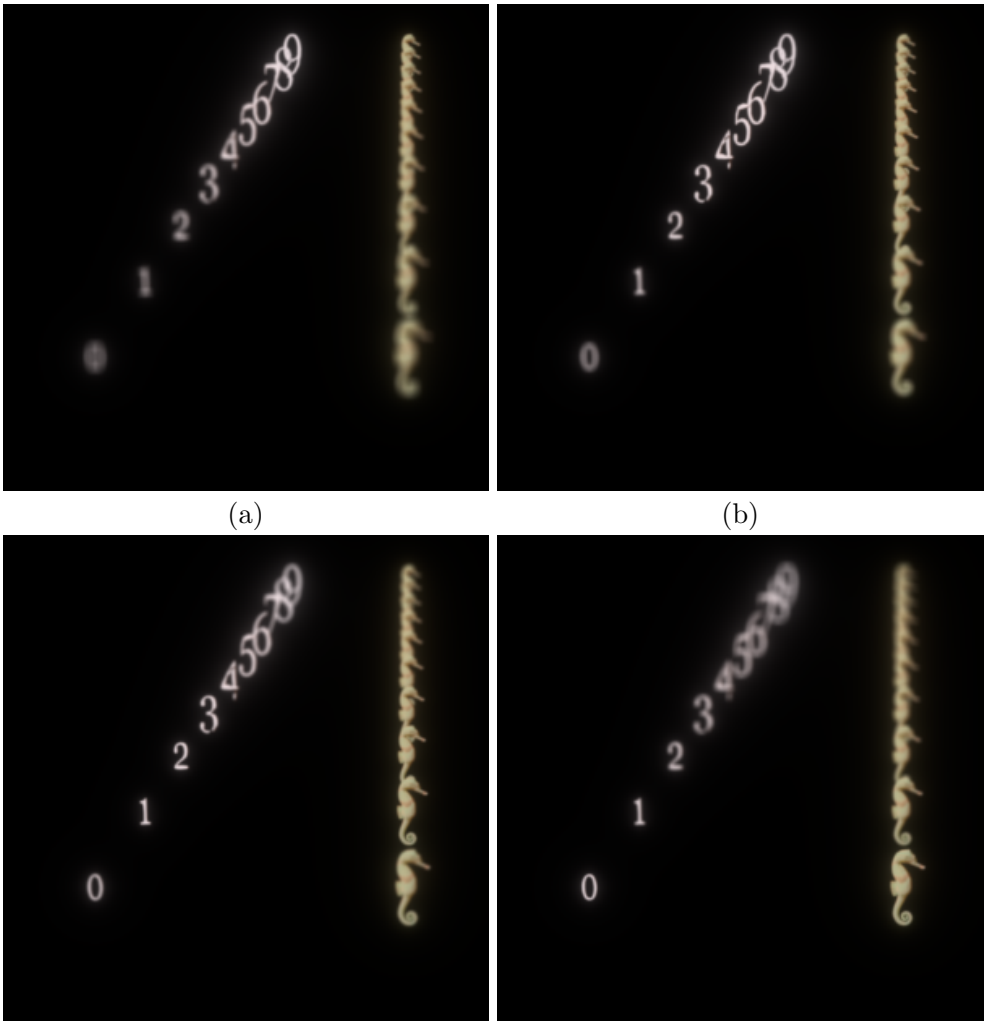
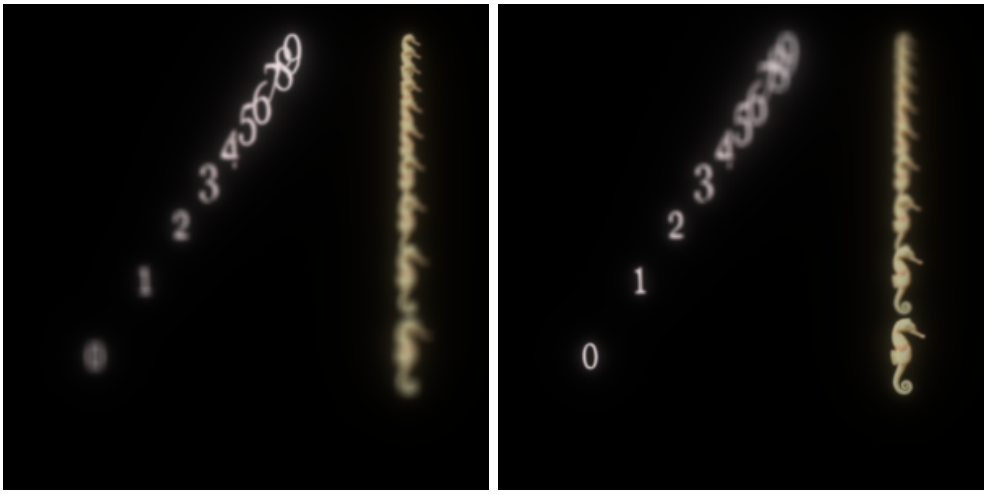


Figure 2.5: Different focus points DoF images of seahorses with 25×25 camera in the array. (a) Refocused with $\alpha = 0.11$. The sharpest seahorse is the last two where they are more than 13 m away. (b) Refocused with $\alpha = 0.2$. I can see that from the third seahorse (8.2 m away) to eights one (13.2 m away), I have sharper seahorses compared to the rest. (c) Refocused with $\alpha = 0.3$ and the second and third seahorses are the sharpest of all. (d) Refocused with $\alpha = 0.4$. I can see that the first seahorse is the sharpest one. These results can be compared to the predicted focus distances in Table 2.1.



(a) (b)

Figure 2.6: The same refocusing parameter $\alpha = 0.4$ results in different focusing points because of a difference in angular resolution. (a) The CA is 5×5 and the focus point is 11.6 m so the last two seahorses are within the DoF. (b) The array size is 25×25. Here, the nearest seahorse is only within the DoF because the far DioF is 5.9 m, and the second seahorse is placed at 6.2 m away from the CA.

Array size	α	**Far DioF Border(m)**	**Narrow DioF border(m)**	**Point of Focus**
25×25	0.1	35.7	13.6	24.6
25×25	0.2	13.2	8.1	10.7
25×25	0.3	8.1	5.7	6.9
25×25	0.4	5.9	4.5	5.1
5×5	0.4	82.5	11.6	25.0

Table 2.1
Corresponding DoF distances for each value of refocusing parameter, α, and for each CA.

2.4.2 Adding Practical Effects:Vignetting

Here, I follow the explanation in Sec. 2.2.4 to create and add the main lens and MLA vignetting to LF. The LF used for demonstrating these effects is the Barcelona Pavilion model with a downsampled angular resolution of $(12, 12)$.

2.4.2.1 MLA Vignetting

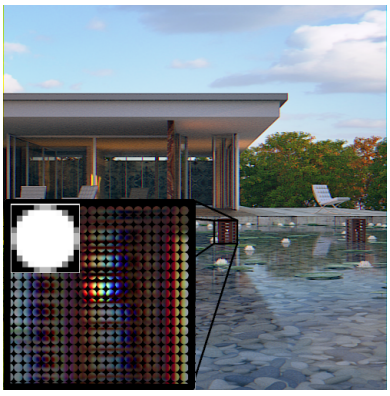


Figure 2.7: MLA mechanical vignetting added to Barcelona Pavilion with angular resolution of $(12, 12)$. The middle chandelier is enlarged to see the effects of the vignetting. A sample MLA aperture used to create this vignetting effect is depicted on top left of the enlarged chandelier.

An MLA vignetting filter was created using Eq. (2.10) and applied it to the LF via Eq. (2.8). The resulting detector image is shown in Fig. 2.7. An inset shows both a close up view and the MLA vignetting mask.

Figure 2.8(a) shows the LF viewpoint images with MLA vignetting applied. As I

pointed out in Sec. 2.1, our intuition tells us that the CA and SPC are equivalent. Here, I see one benefit of implementing practical effects as a separate step. I can see that MLA vignetting will result in losing the view point images at the corners. By applying this mask before rendering, I can safely eliminate rendering those viewpoints complex, compute-intensive scenes.

From inspection, I can see that MLA vignetting does not reduce the spatial resolution. However, referring again to Fig. 2.6, I know that reduced angular resolution will have a deeper depth or longer DoF compared to the pristine LF. Figure 2.9 further illustrates this concept. Comparing subfigure (a) to (c), I see that the effect of applying MLA vignetting is to reduce the DoF.

2.4.2.2 Main Lens Vignetting

Main lens mechanical vignetting is added by applying the mask generated by Eq. (2.9) to the pristine LF via Eq. (2.8). Figure 2.8(b) shows the resulting tiled viewpoint images. I can see that compared to Fig. 2.8(a) the angular resolution is not reduced by main lens vignetting. Intuitively, the effect of main lens vignetting is to only reduce the spatial information in the LF.

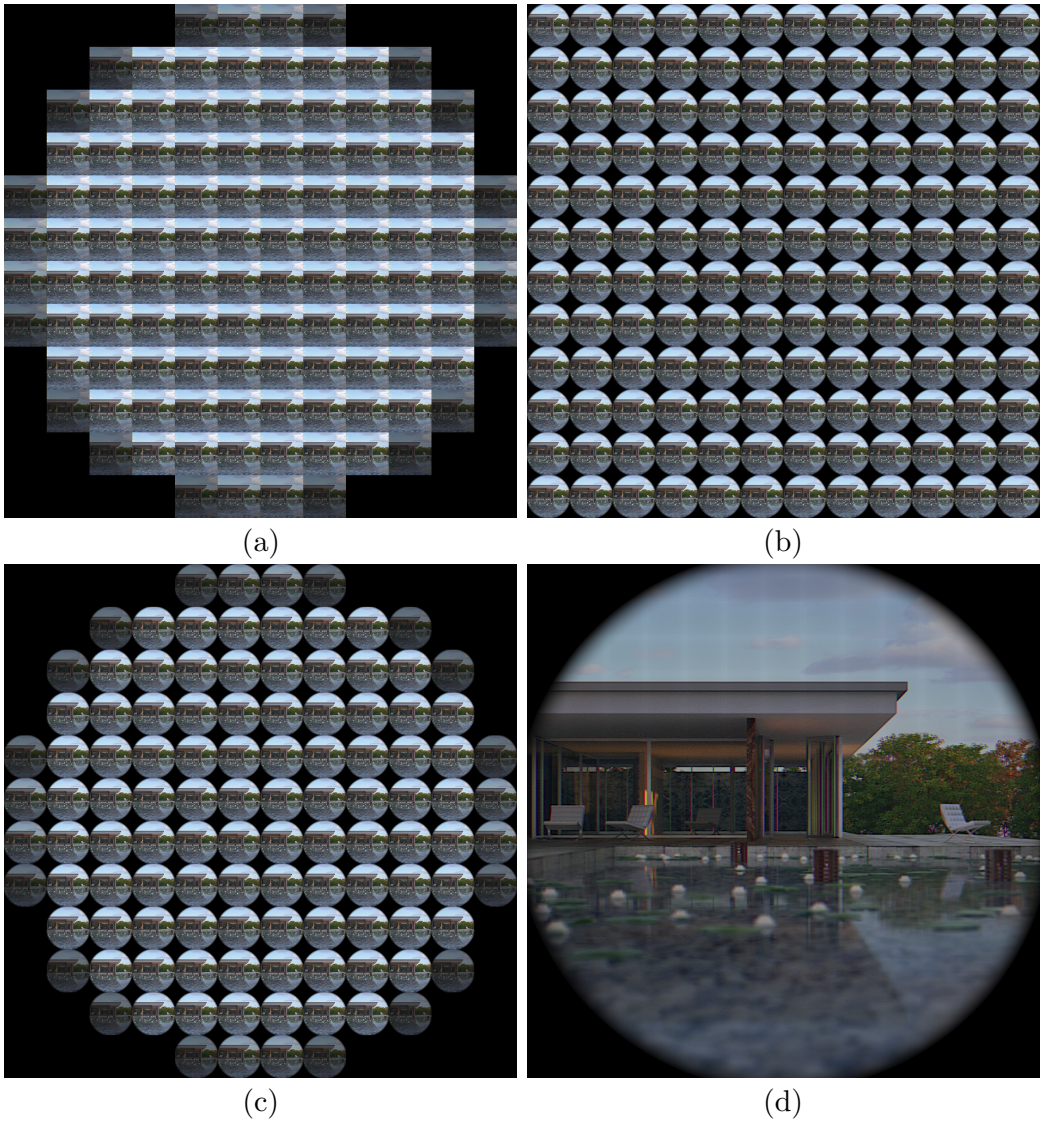


Figure 2.8: Applying vignetting mask to the Barcelona's LF. (a) The tile view of applying the MLA mask to the LF. I can see that some of the views are lost. (b) The tile view of adding main lens vignetting to the pristine LF. The spatial resolution reduction is evident, but none of the views are lost. (c) The tile view of the LF with both main and MLA vignetting added. (d) The LF with both vignetting with the shape of PC's sensor.

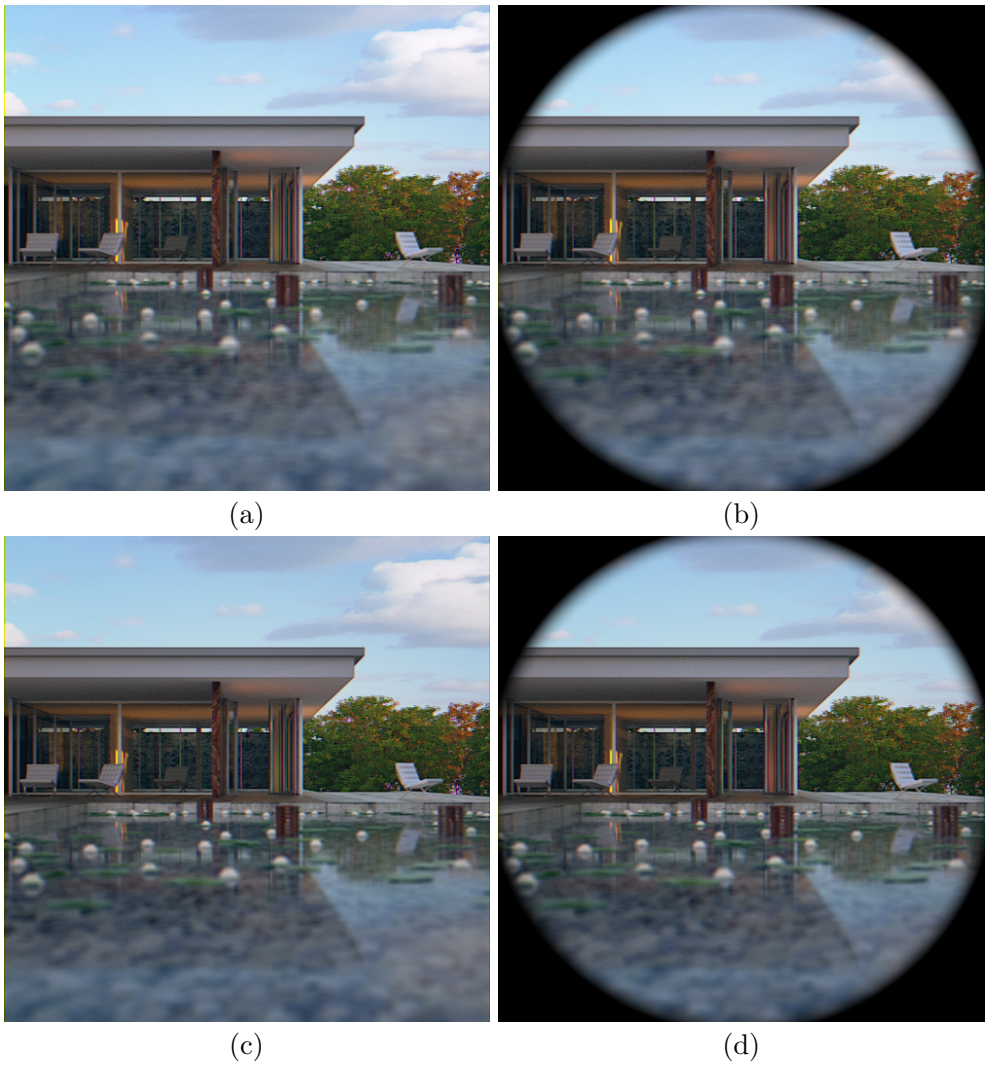


Figure 2.9: (a) The DoF image derived from the pristine LF with angular resolution of 12×12. (b) DoF from the same LF but with the main lens vignetting added, note that the DoF is unchanged. (c) DoF from the same LF, with MLA vignetting. I can see (a) has shallower DoF compared to (c). (d) The DoF of LF with applied MLA and main lens vignetting. This DoF is as the same as (c).

2.4.2.3 MLA and Main Lens Vignetting

By applying both the main lens vignetting and the MLA vignetting, I can examine both effects both in the sensor plane image [Fig. 2.8(d)] and the resulting viewpoint

images [Fig. 2.8(c)]. Likewise, I also produce a DoF image as in Fig. 2.9(d). This DoF here is the same as Fig. 2.9(c) because the main aperture does not change the angular sampling.

2.5 Conclusions and Future Work

I have presented a transformation model allowing us to find the CA geometry for a given SPC configuration. This transform includes in the the relative position of each camera with respect to the principal optical axis of the SPC. Also, in contrast to previous works, I define the FoV and aperture size of each camera. While physically building such a CA configuration with proper calibration can be difficult, it is trivial in a simulation model. Further, simulating a CA is much simpler than an SPC that may potentially have multiple lens elements in addition to the MLA. To verify our model, I evaluated the focusing distance of our simulated LFs. These distances were compared to those estimated by the model described by Hahne [34] and were shown to be in agreement within the expected measurement error.

Furthermore, I have presented a way of adding simple practical effects to a pristine LF by postprocessing. By considering MLA and main lens vignetting separately, I am able to effectively show that the MLA vignetting reduces the minimum DoF available in LF's DoF images. Likewise, that main lens vignetting filters only spatial

information. While known theoretically, our approach to including practical effects allows designers of LF capture systems to quickly and easily evaluate the effect of design choices on the LF. In contrast, simulation approaches that model the entire SPC may require extensive changes for each option evaluated. Our approach to applying vignetting also has the benefit that it can be applied directly to the 4-D LF data object without reshaping the matrix or extracting viewpoint images.

Extensions to this work would include exploring other PC geometries such as the focused PC and irregular sampling geometries. Similarly, I plan to explore other practical effects including intrinsic effects such as imaging system aberrations and extrinsic effects such as motion blurring and environmental effects such as rain and fog.

Chapter 3

Light Field Compression by Residual CNN-Assisted JPEG

3.1 Introduction

Light fields (LF), as compared to conventional images, have two extra dimensions which represent angular information of the scene [2, 4, 9]. Hence, LFs contain a relatively large volume of data that makes storing and portability time consuming and costly. Also, decompression of LF video with a high angular resolution at acceptable *frames per second* (fps) for streaming is challenging. I aim to address these challenges by predicting the entire LF from its JPEG compressed center view.

Direct application of standard image compression techniques, such as JPEG, PNG, *etc.*, on an LF does not take advantage of existing redundancies between LF views. Video compression techniques, however, achieved better success in compressing LFs. To use video compression algorithms on LFs, a sequence of images is built from LF views, which is called pseudo-sequence[14]. A combination of *machine learning* (ML) methods, capable of predicting LF views, and video compression techniques was explored in [38]. In this chapter, I present a combination of JPEG compression with ML view predictions. LF synthesis techniques have shown the possibility of estimating the entire LF from a single view or a set of sparse views. Here, I show that there is enough information in the JPEG compressed center view—as well as a group of *sub-aperture images* (SAIs)—to predict the entire LF with a quality comparable to the use of the state-of-the-art video compression techniques on the LF. I test the success of our method by comparing it against state-of-the-art methods in LF compression that use the existing HEVC compression. Some extensions using deep-learning and other techniques are applied to improve the quality of HEVC application to the LF. But the only compression, so far, that is possible to apply to any variation of LF is still HEVC. For this reason, HEVC, while being the base model, is the state-of-the-are.

Our method is faster in compression and decompression by 100x and 10x, respectively, compared to the direct use of HEVC. This speed up means a set of 30 LFs with a spatial resolution of $(375, 540)$ and angular resolution of $(7, 7)$ can be decompressed on a typical gaming GPU in less than 0.02 seconds, while HEVC-based methods require

more than 0.39 seconds. With increases in spatial or angular dimensions, HEVC-based methods reconstruction soon takes more than one second. Thus, streaming will not be possible without pre-decompression. Furthermore, while speeding up the process, I have maintained and, in most cases, improved the quality of reconstruction at the same *bit per pixel* (bpp) ratio. I have used the *mean of peak signal-to-noise* (MPSNR) ratio over all of the views and *mean structural similarity index metric* (MSSIM) to compare the reconstructed LFs of my model with those that use HEVC. I show that while the MPSNR of my method is comparable to the direct employment of HEVC, my model can achieve higher MSSIM. This results in fewer artifacts and better quality in the extracted synthetic aperture *depth of field* (DoF) images. Note that I built my model to be fully convolutional, thus, it can be used on LFs with any spatial resolution. Also, my model works in the RGB channel. This is an advantage compared to other techniques, which are using YUV channel, because most of the available LF datasets are in RGB and digital conversion between RGB and YUV is not lossless [39].

The contributions of this paper are as follows. I achieved compression speed-up of more than 100x and decompression speed-up of more than 10x compared to the use of HEVC on pseudo-sequences of LF views. At an average bpp of .0047, the LF's DoF reconstructed with my method improved the SSIM by 0.31% on average over the test dataset compared to the direct use of HEVC. Finally, I introduce a small, fast, and efficient *convolutional neural network* (CNN) for enhancing JPEG images for use in

LFs. This network also boosts the SSIM of the final decompressed LF.

Our code and trained network can be found at: https://github.com/ehedayati/LFCompressionByRCNN-JPEG

3.2 Related Work

3.2.1 Light Field View Synthesis

Linear view synthesis by Levin and Durand [40] and depth of field extension and super-resolution by Bishop and Favaro [41] are among the earliest works on LF view synthesis and reconstruction. Flynn *et al.* [42] proposed a deep learning method to predict novel views from a sequence of images with wide baselines. LF view synthesis became more popular after Kalantari *et al.* [1] showed in their work that an LF can be synthesized from its corner SAIs with high quality. Building on the work of Kalantari *et al.*, Yeung *et al.* used different sets of views to reconstruct dense LFs [43]. Srinivasan *et al.* demonstrate the possibility of estimating the entire LF from its center view by extrapolating using machine learning methods [44]. Choi *et al.* extended the extrapolation to an LF taken with arrays of cameras [45]. LF fusion [46] and depth-guided techniques [47] have been popular in reconstructing an LF from a single or a sparse set of SAIs. Hu *et al.* [48] aimed for a faster LF reconstruction

method by using hierarchical features fusion.

The backbone of nearly every view synthesis method enumerated here is the depth-map estimation. The current work is categorized as single view LF reconstruction. my method is unique because I use a lossy compressed JPEG view from which to estimate the entire LF. I use residual learning methods to guess the possible artifacts from the JPEG compressed version of the center view to assist the main network for accurately estimating the depth map.

3.2.2 Light Field Compression

Lossless and lossy compression methods have been investigated extensively in the literature. For the lossless model, Perra [49] proposed an adaptive block differential prediction method and Helin *et al.* [50] described a sparse modeling with a predictive coding for SAIs of the LF.

The lossy models can be classified in to sub-categories of: i) standardized image/video compression techniques and ii) machine learning assisted compression techniques.

3.2.2.1 LF compression by standardized image/video compression methods

Standardized image and video compression techniques (especially HEVC) have been directly used to address the problem of the bulkiness of the LF, see e.g., [14, 15, 16]. Other methods, such as homography-based low-rank models [51] and Fourier disparity layers [52], have been used to reduce the angular dimension of the LF. In another work, the LF depth was segmented into 4-D spatial-angular blocks, which were used for prediction, followed by encoding the residue using JPEG-2000 [53].

3.2.2.2 Machine learning assisted compression techniques

Followed by the breakthrough in synthesizing LF views from its four corners using CNN learning techniques introduced by [1], another work introduced a compression technique by using the same method and compressing the four corner views by HEVC [54]. In another work, the authors proposed to keep half of the views and encode them by HEVC and synthesize the other half by a CNN [38]. A CNN based epipolar plane image super-resolution algorithm was used in cooperation with HEVC to compress LF as well [55]. Wang *et al*. proposed a new LF video compression technique by deploying view synthesis methods from multiple inputs while encoding the input views by a proposed region-of-interest scheme [56]. Generative adversarial network based

methods have been used in cooperation with video codec techniques to compress LF in [57, 58]. There has been multiple works on LF compression with use of standard video compression codecs in combination with learning based view synthesis [59, 60]. Unfortunately, I could not find any public version of these codes, or trained networks for the purpose of comparing my results with them. Hence, I have chosen the pseudo-sequence HEVC compression method for comparison because of its easy implementation and availability of the HEVC codec.

To the best of my knowledge, because the view extrapolation is ill posed, LF reconstruction from a lossy compressed single input (specifically, JPEG) has not been explored before my work.

3.2.3 JPEG Compression Artifact Reduction

For several decades, different researchers addressed the JPEG compression artifact reduction generally in three main groups: prior knowledge-based, filter-based, and learning-based approaches. Here though, I am interested in learning-based approaches. The basic intention of learning-based methods is to find a non-linear mapping between the JPEG compressed image—compressed at different compression ratios—to the ground truth uncompressed image. To the best of my knowledge, the first deep learning model to address this problem was created by Dong *et al.*

[61], where they showed the possibility of enhancing the reconstructed JPEG image by a relatively shallow CNN. Since then, multiple researchers have gradually improved the performance of learning-based methods by introducing new networks such as: dual-domain representations [62], deep dual-domain based fast restoration [63], encoder-decoder networks with symmetric skip connection [64], CAS-CNN [65], one-to-one networks [66], DMCNN [67], and dual-stream multi-path recursive residual network [68]. While deeper networks and state-of-the-art architectures have improved the task of JPEG artifact reduction, I am not focused solely on this task here. The ultimate goal of my JPEG-Hance network is to improve the estimated depth-map from the JPEG compressed center image of an LF. JPEG artifact reduction is the natural first step for extracting better depth-maps.

3.3 The Proposed Method

Here I describe my compression and decompression pipeline. The compression pipeline is simply extraction of the center view of the LF, compression by JPEG at 50% quality, followed by discarding of all other views. The decompression pipeline has the following steps:

1. JPEG decompression of the center view c_J

2. Enhancing c_J by JPEG-Hance to c_E,

$$c_E = J(c_J). \tag{3.1}$$

3. Estimating depth map $d(x, u)$ of every view u from c_E,

$$d = D(c_E). \tag{3.2}$$

4. Reconstructing LF by

$$L(x, u)_{u_0 \to u} = L(x + (u - u_0)d(x, u), u_0), \tag{3.3}$$

where L is the approximated LF and u_0 is the middle view index. Variables x and u are spatial and angular indices.

3.3.1 Networks architectures

3.3.1.1 JPEG-Hance

The main goal of my JPEG-Hance network is to assist the Depth-Net in providing better depth map estimation. In doing so, it is certainly beneficial to improve the

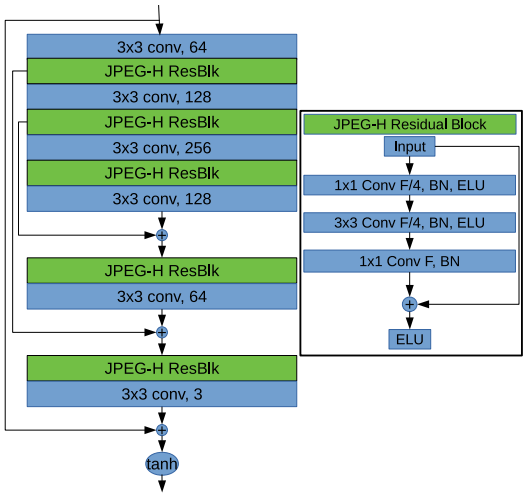


Figure 3.1: JPEG-Hance detailed structure

overall quality of the JPEG decompressed image by reducing the error between uncompressed ground truth images and the lossy compressed ones. However, the goal of my network is not general JPEG artifact reduction; instead, JPEG-Hance should learn to enhance the parts of the image which have the most effect on improving depth information extraction. To achieve this task, JPEG-Hance also needs to find correspondence information from the extracted depth maps. Therefore, it is trained in two phases: first, it learns to enhance any typical JPEG decompressed image, then again as part of the whole depth estimation pipeline. The architecture of JPEG-Hance is shown in Fig. 3.1. Inspired by ResNet50's *bottleneck* building blocks structure [69], I have designed my JPEG-Hance as residual blocks. I added a *batch normalization* (BN) layer after each convolution followed by an *exponential linear unit* (ELU). The *ELU* followed by a last layer *tanh* seems to be the most promising activation pair of functions when dealing with regression of image data scaled to the interval $[-1, 1]$.

JPEG-Hance is pre-trained by minimizing the mean squared error of each pixel value in the RGB channels. Then it is added to the training pipeline for full reconstruction of LFs.

3.3.1.2 Depth-Net

Multiple images provide geometry information which can be used for LF reconstruction. A single image does not provide such information. Therefore, such information needs to be extracted by other methods. Machine learning techniques, particularly CNNs, showed a promising potential for estimating geometry from a single image [1, 43, 44]. Thus, for the problem of depth estimation from my enhanced center image, I use a residual CNN.

Our Depth-Net, depicted in Fig. 3.2, is responsible for estimating the depth map (disparity map) for all 49 views from the middle JPEG compressed view. Depth-net has three variants of residual blocks. The first variant is a down-sampler which uses a 2-D convolution with strides of (2×2), halving the spatial dimension of the input image. This block is used just before the first Depth Residual Block; each time the feature size is increased. The second type of block, the Depth Residual block, is the main residual block. This block is used the most and extracts most of the features. The structure of the Depth Residual block mimics the bottleneck structure of ResNet50 with added instance normalization after each of the first two convolution

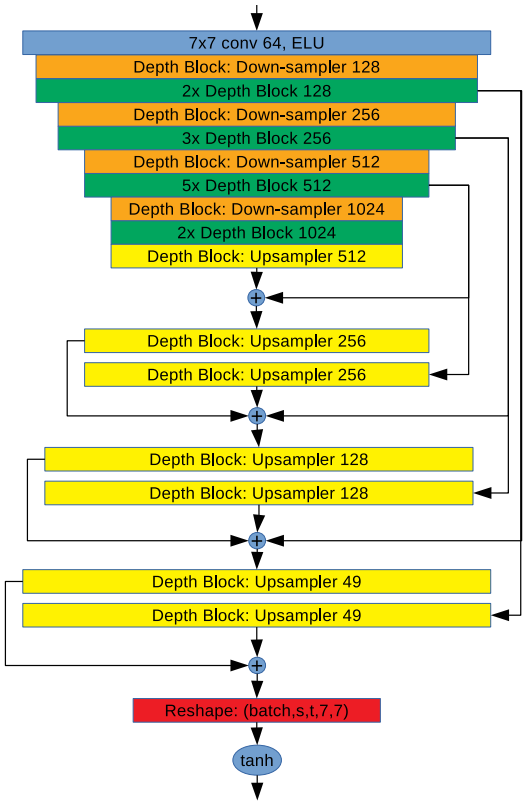


Figure 3.2: Depth-Net detailed structure

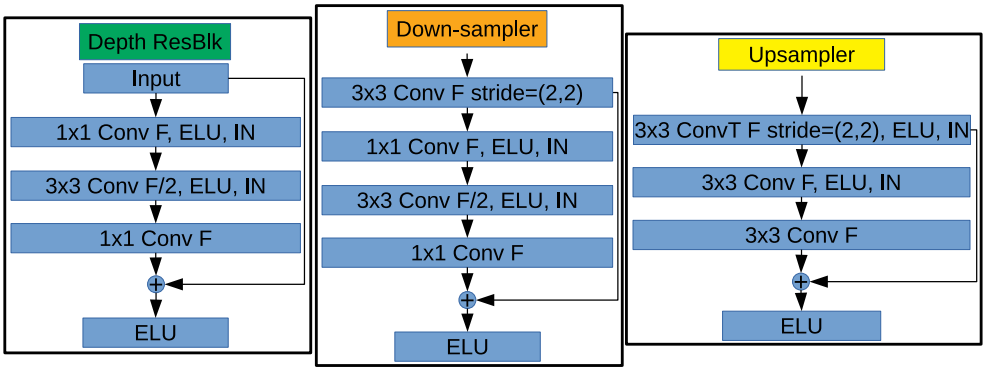


Figure 3.3: Depth-Net residual blocks

layers. Last, the Upsampler block is constructed to have a 2-D deconvolution (transposed convolution) layer and two 2-D convolution layers with kernel size of (3×3). The deconvolution layer's stride is set to (2×2). These blocks are shown in Fig. 3.3.

Because I am training the Depth-Net on the actual LF data and not the ground truth depth maps, my loss functions have to be designed to train the network in an unsupervised manner. my Depth-net is predicting the LF's depth while I do not have the ground truth depth to supervise the training. Thus, I define the Depth-Net pre-training loss function to be a weighted sum of four sub-functions: i) photometric loss L_p, ii) defocus loss L_r, iii) depth-consistency loss L_c, and iv) DoF loss L_d. The total loss is simply

$$L_{depth} = \alpha L_p + \alpha_1 L_r + \alpha_2 L_c + \alpha_3 L_d, \tag{3.4}$$

where $\alpha, \alpha_1, \alpha_2$ and α_3 are chosen to be 2, 100, 0.02, and 10 in my conducted experiments, which were empirically found to work well overall.

The image quality comparison sub-function ψ [47] is constructed by combining mean absolute difference of pixels and image structural dissimilarity (DSSIM) that is derived from the *structural similarity index metric* (SSIM) [70]:

$$\psi(I_1, I_2) = \beta \frac{1 - SSIM(I_1, I_2)}{2} + (1 - \beta) \left\| I_1 - I_2 \right\|_1, \tag{3.5}$$

where I_1, I_2 are two images that are being compared and $\beta \in [0, 1]$, which I empirically found that 0.15 yields the best training results. Using the sub-function ψ, photometric

loss is defined as [47]

$$L_p = \sum_u \big[\psi(L(x,u)_{u_0 \to u}, L(x,u)) + \psi(L(x,u)_{u \to u_0}, L(x,u_0))\big]. \tag{3.6}$$

Because I am training an unsupervised Depth-Net, the more prior knowledge I can give the network, the better will be the training quality. Zhou *et al.* [47] introduced defocus cue loss

$$L_r = \psi\big(L(x,u_0), \frac{1}{N}\sum_u L(x,u)_{u \to u_0}\big). \tag{3.7}$$

Also depth consistency (left-right or forward-backward) has been shown in the literature [71, 72, 73] to be a promising regularizer for LF view synthesis purpose, where

$$d_{u_0 \to u}(x) = d_{u_0}\big(x, (u-u_0)d(x,u)\big), \tag{3.8a}$$

$$L_c = \sum_u ||d_u(x) - d_{u_0 \to u}(x)||_1. \tag{3.8b}$$

Finally I have included *depth of field* (DoF) loss to further assist the network in

learning depth information, where

$$DoF = \frac{1}{u}\sum_{u} L(x,u), \tag{3.9a}$$

$$L_d = \psi\big(DoF, DoF_{u_0 \to u}\big). \tag{3.9b}$$

3.4 Experiments

In this section, I describe my method's implementation details. Then, I use public data sets [1, 44] to evaluate my method and investigate the impact of different parts of my network on the performance of my model.

3.4.1 Data sets

I have conducted my experiments over the two public data sets: *Flowers* [44] and *30 Scenes* [1]. Both of these data sets are captured by a Lytro Illum camera. The angular resolution of these data sets is 14×14 views and the spatial resolution is variable between 375×540 and 376×541 pixels. The LF from these data sets are cropped to the size of $7 \times 7 \times 375 \times 540$ to have a consistent size and vignetting.

3.4.2 Implementation details

Our pipeline was trained in multiple steps. I have implemented my model with Tensorflow 2.2 in python 3.7 on a workstation with an Intel Xeon W-2223 3.60 GHz, 64GB DDR4 memory, and NVIDIA Quadro RTX 5000.

3.4.2.1 JPEG-Hance

Our JPEG-Hance was pre-trained on the *30 Scenes* training data set, which contains 100 scenes. The center views of these 100 scenes were extracted and used for training. In the training phase, the spatial dimensions of the JPEG-Hance were set to 128×128. First a training pool of images with dimensions of 128×128 was created by cropping the center views of the 100 scenes at 8 pixels steps. Therefore, the training pool had $150,000$ different crops which I found sufficient for the JPEG-Hance network to be trained without over-fitting or under-fitting. The learning rate was set to 0.0004 which was empirically found to have sufficiently good results for pre-training step. The JPEG-Hance has a relatively small network: only $202,435$ trainable parameters. The pre-training phase took about 90 minutes to converge.

3.4.2.2 Depth-Net

Our Depth-Net also has a pre-training step. The Depth-Net was pre-trained on the *Flowers* data set, which has $3,343$ flowers. During the pre-training phase, the JPEG-Hance was used for enhancing center images while only Depth-Net parameters were trained. The input pipeline of the flowers contains random croppings to 128 and data augmentation with 50% selection rate for the original data, 15% chance for random contrast change between $[0.1, 0.5]$, 15% chance that the brightness was changed randomly up to $0.4\times$ original brightness, and the remaining 20% where the hue was randomly changed by up to $0.4\times$. During the pre-training phase, 16 random crops were extracted for each epoch, and the network was trained for 10 epochs. The learning rate was 0.0004 that again proved empirically its sufficiency in our experiment. Depth-Net is the main network responsible for extracting the depth map; thus, it has more trainable parameters: about 38.2 million. The pre-training phase takes about 7 hours to converge.

A sample estimated depth is depicted in Fig. 3.4. This illustration demonstrates that the edges are not very sharp. This is because I have used the highly compressed lossy JPEG on center view, which adds blur to the edges, to estimate the disparity map. Thus, the resulting depth map is somewhat blurry.

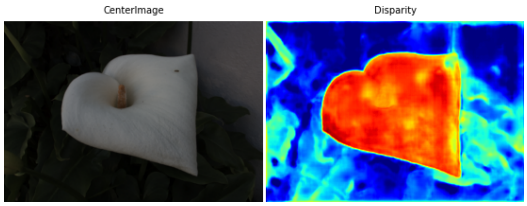


Figure 3.4: An estimated depth map. I can see that my network estimate is correct for most parts of the image. The map indicates that the flower is the nearest object to the LF camera and the leaves are just behind the flower and the wall is at the background, which is very realistic.

3.4.2.3 Training the entire pipeline

After pre-training the two networks, I can now train the entire pipeline. I add 100 scenes from the *Flowers* data set pool and use the same input pipeline as the one used for Depth-Net. The entire pipeline was trained for 45 epochs, gradually decreasing the learning rate from 0.0001 to 0.000001. The learning rates schedule defined in a way that the first 10 epochs was trained by 0.0001, then it was halved for the next 10 epoch. From epoch 20 to 40, each 5 epochs, the learning rate was halved. Finally, the last 5 epochs were trained by setting the learning rate to 0.000001. These learning rates were chosen based on observing the training procedure and the it was decreased when no improvement was observed. I have stopped training after epoch 45 due to not seeing any improvement afterward.

The last fine-tuning step includes training the pipeline on the data sets with input spatial dimensions of 375×540. Here the augmentation selection is 25% original, 25% random contrast, 25% random brightness, and 25% random hue. Because of the

structure of the Depth-Net network, the input images have to be zero-padded and the resulting LFs should be cropped to the correct size. The input dimension of the Depth-Net is 384×544. The fine tuning phase takes 40 epochs for the network to converge with gradually decaying learning rate from 0.00005 to 0.000001. The learning rate scheduling here had the same terminology, except that the halving started at epoch 10 for each 5 epoch, and epoch 35 to 40 trained with 0.000001 as the learning rate. The fine-tuning phase took around 20 hours to converge, while all other pre-training phases took less than 10 hours cumulatively.

3.4.3 Performance comparison

We compared my compression-decompression results with a pseudo sequence method using the HEVC video compression Codec. I chose a raster sequence over spiral because raster had slightly better performance. The 30 scenes data set was used for comparing my method with HEVC. I use MSSIM and MPSNR metrics as well as SSIM and PSNR of the extracted DoF from LFs to compare the results, where

$$MSSIM = \frac{1}{M} \sum_{u} SSIM(LF, \tilde{LF}), \tag{3.10}$$

$$MPSNR = \frac{1}{M} \sum_{u} PSNR(LF, \tilde{LF}). \tag{3.11}$$

To have a fair comparison, I tuned the QP factor of HEVC for each LF to reach approximately similar bpp between HEVC compressed LF and my method's compressed representative. The average bpp for both methods on the 30 scenes data is 0.0047. Fig. 3.5 shows that the LFs reconstructed by my method have very similar MPSNR and MSSIM to those decompressed by HEVC. By carefully examining Fig. 3.5 it is evident that, while my method outperforms HEVC in MSSIM, it is slightly inferior in MPSNR performance. Fig. 3.6 plots the PSNR and SSIM metrics for extracted DoFs from the reconstructed LFs. Here, my method meaningfully outperforms HEVC in SSIM metrics while further reducing the gap in PSNR. Because of this dual behavior in SSIM and PSNR metrics between my method's results and HEVC's, I have conducted experiment on the refocused images to find out which method is more reliable.

The PSNR comparison between my model and HEVC over the test set for near and far focus in shown in Fig. 3.7. These results show that in some cases my model is superior and for other cases HEVC performs better. The mean PSNR of the HEVC for the test set is greater than mine by 1.3db for the near focus and 0.6db or the far focus. But for the case of the SSIM metric over the same test set, depicted in Fig. 3.8, I can see that in both near and far focuses, my model is performing better. my model has 0.4% greater SSIM for near focus and 1.8% for far focus.

While the quantitative results look nearly the same between my method and the

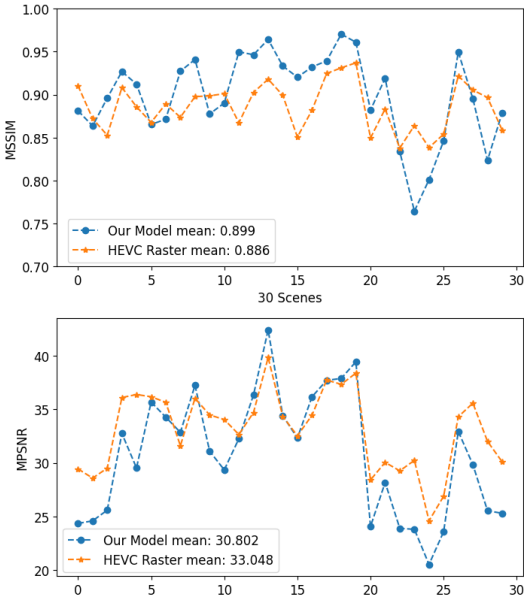


Figure 3.5: The top figure is showing the MSSIM for each reconstructed LF by my method and HEVC. The bottom figure is the MPSNR calculated for each reconstructed LF.

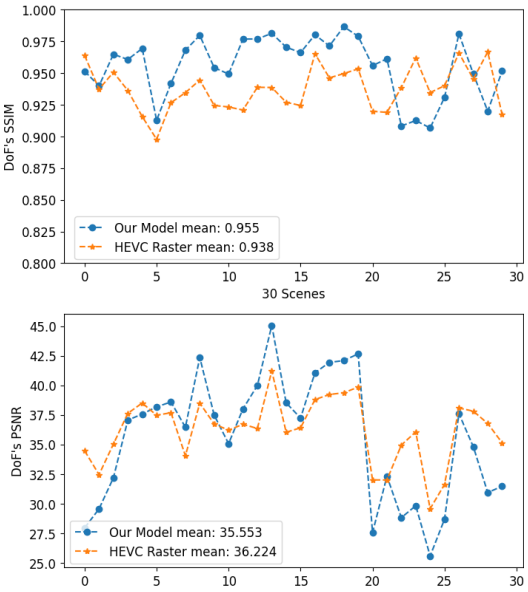


Figure 3.6: The DoF images extracted from ground truth LFs and the reconstructed LFs are compared using SSIM and PSNR metrics. The top plot is representing SSIM and the bottom one is showing PSNR for each LF in the 30 scenes.

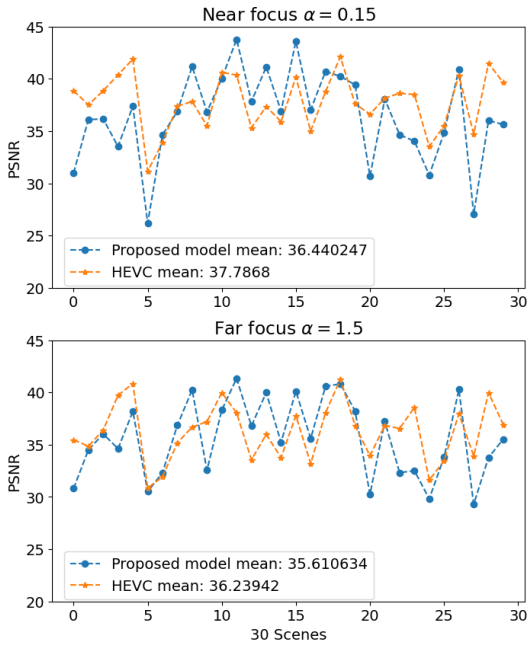


Figure 3.7: The reconstructed LFs from my model and HEVC is used to extract refocused images with refocusing parameters of $\alpha = 0.15, 1.5$, The top plot is showing the PSNR of the near focus images, and the bottom on is the far focus PSNRs.

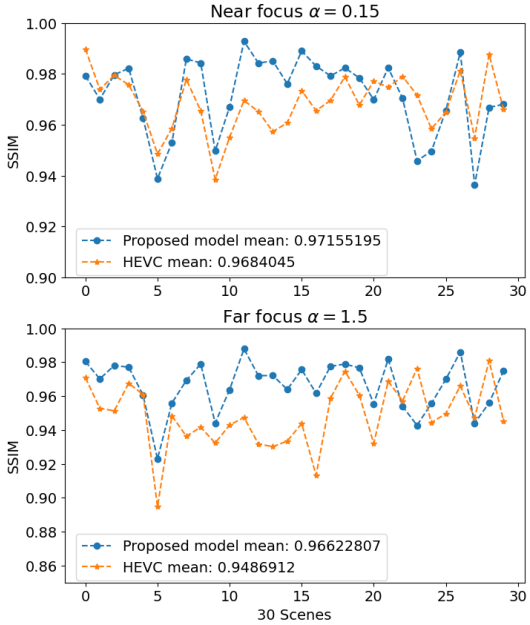


Figure 3.8: The SSIM of the near and far focus images extracted from HEVC and my proposed model is calculated and plotted.

HEVC compression technique, the reconstructed images show the real differences. Fig. 3.9 shows the reconstructed view of a statue from the LF. It is the 25th LF in the 30 Scenes data set. By looking at Fig. 3.6, I can see that HEVC's MPSNR for this LF is more than $3dB$ greater than my model. Yet, Fig. 3.9 shows that the reconstructed LFs from my model are producing a visually better representation of the ground truth image. The HEVC reconstructed LF generally has more blur all over the image. This blur is, to some extent, caused by severe data loss. Fig. 3.10 illustrates another example, where the second leaf behind the front one is not reconstructed in HEVC decompressed LF. Last, I can see more details and better texture in the extracted DoF from my model, demonstrated in Fig. 3.11. In the refocused images extracted from my model, they have the same depth to the reference image and are refocused to the same focal plane as the ground truth. Images reconstructed from HEVC seem to lose the focal plane, especially in the one focused on the car in Fig. 3.9. Here, it becomes clear that my model is more successful in retaining the LF physical information. On the other hand, this finding indicates that the available quantitative metrics do not tell the whole story in comparing the two LF reconstruction methods. It is worth noting that for quality assessment, the SSIM metric is showing more agreement with the qualitative comparison than PSNR.

Figure 3.9: Two different focus points of an LF. The refocused DoFs in the top row are focused with $\alpha = 0.1$ to the nearest flower, and the bottom row DoFs are focused at the car with $\alpha = 1.5$. The images in the left most column are ground truth images. The middle column shows DoF images extracted from the HEVC reconstructed LF. The rightmost column contains the results from the LF reconstructed by my model.

Figure 3.10: In this figure, a small slice from 3.9 shows HEVC loses more physical information compared to my model. The second leaf just behind the front leaf is not visible in the HEVC reconstructed LF's DoF.

3.4.4 Speed Gain

Table 3.1 shows that the compression time of my proposed method is more than 100 times faster than that of HEVC, on the same computational hardware. This is

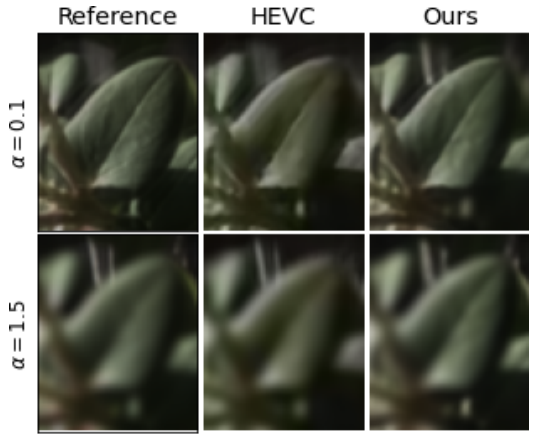


Figure 3.11: For better texture comparison, a small leaf from the 3.9 is magnified in this figure.

Table 3.1
The time takes to compress all 30 LFs in the 30 scenes data set using my method and the HEVC. I can see an speed up of more than 102 times.

Method	CUDA	Comp time (s)
HEVC	No	43.53
JPEG-Hance + Depth-Net	No	**0.42**

because my compression pipeline is more efficient, which is just a JPEG algorithm on a fraction of the LF (1/49 in my case with 49 views). The HEVC algorithm processes all of the views.

For decompression, my method is 18 times faster than HEVC; see Table 3.2. To give a fair comparison, I used the NVIDIA optimized HEVC codec using the GPU's video decoder. So on the same hardware, this will be the fastest implementation of HEVC.

Overall, these results indicate that my model is suitable for compressing LF videos with high angular resolution. This is because my method can decompress in near real time inside the GPU without the barrier of transferring high volumes of data from

Table 3.2
The reconstruction time on all LFs from the 30 scenes data set by my method and HEVC. I can see that my method is 18 times faster in decompressing.

Method	CUDA	Rec time (s)
HEVC	yes	0.399
JPEG-Hance + Depth-Net	yes	**0.022**

host to GPU. The bandwidth used from host to GPU is equal to the size of only the center view of the LF.

3.5 Conclusions

I have designed a machine-learning assisted LF compression technique. It contains two sequential custom-designed CNNs: JPEG-Hance and Depth-Net. I showed that there is enough information in a highly compressed LF center view to estimate the depth-map of the LF and then use it to reconstruct the whole LF. Also, compression and decompression are faster with my method. I have used the public *Flowers* and *30 Scenes* data sets to conduct my experiment and also to evaluate my model. I have achieved more than 100 times speedup during compression and about 18 times faster reconstruction compared to using HEVC on LF pseudo sequences. Comparing to HEVC, the reconstructed LFs using my method have superior MSSIMs, and they have comparable MPSNRs. Furthermore, the visual quality of the focal plane images reconstructed using my method are superior. For future work, I will try to add

other views, with varied relative compression ratios, to further improve the quality of reconstruction. I will also explore options, such as improving the network architecture, other loss functions, larger training data sets, *etc.*, to enhance the MPSNR. I am also looking forward to deploying my method on an actual LF video to explore the achieved compression ratio and streaming capabilities.

Chapter 4

A Machine Learning Method for Light Field Refocusing

4.1 Introduction

In order to have a sharp image with the desired information about the capture scene, the choice of aperture size and focus point are very important. But if one can obtain a complete *light field* (LF) from the scene, one can synthesize the aperture of choice at any desired focus point by postprocessing. By summing the views inside the aperture of choice, one can quickly obtain a synthetic-aperture image from the LF. In the case of a plenoptic camera, one captures two extra dimensions in addition to the

regular two spatial dimensions. This 4-D LF enables limited refocusing and synthetic-aperture *depth of field* (DoF) images by post-processing, which are not available in conventional cameras [2, 4, 9]. The shift-and-sum method [7] and Fourier refocusing [17] are two widely used methods for LF refocusing.

The shift-and-sum refocusing method requires 4-dimensional integration for each new refocused image, which is time-consuming. While Fourier slice photography [17] can perform refocusing faster, it needs an initial 4-D Fourier transform, which is again slow; also, the quality of the Fourier refocused images are less than that of the slower shift-and-sum method because that discrete Fourier and inverse Fourier transform on the same data is not lossless. Also, both Fourier slice method and the shift-and-sum method are predicting each refocused image by interpolating some of the missing points; thus, they are simply estimates of the reality. Both methods sometimes have color and brightness prediction problems, which may need to be fixed by re-calibration.

While *Fourier slice photography* [17] can be used to extract multiple refocused images from LFs faster than the shift-and-sum method by sacrificing a little quality, real-time refocusing of the LF, especially without time-consuming pre-processing, remains a pertinent challenge. my goal is to address this problem by predicting the refocused images using a machine learning algorithm. By employing my model with a conventional desktop GPU, I can extract multiple focus-points from LFs in real-time

with higher quality refocused images than both of the conventional refocusing methods. Real-time processing of the LFs can enable LF cameras to be used for detection tasks as well, possibly with better accuracy than obtained with conventional cameras. For example, LF cameras can be used for improved object detection because such algorithms can find objects from different focus points in LFs [74]. In the case of conventional cameras, just a single focus point is available for detection, and blurred parts of the image need another capture with a different focus to be revealed. This problem can be solved by employing an LF instead of a conventional image.

In this chapter, I introduce a residual network to estimate a set of 16 refocused images on different focal planes with refocusing parameters of $\alpha = \{0.125, 0.250, 0.375, ..., 2.0\}$ [1] from an LF with 7×7 angular resolution. We show that my approach can predict refocused images in real-time, with superior color prediction, as compared to both of the conventional refocusing methods.

The code to accomplish my refocusing method will be publicly available upon acceptance.

[1] One can use any set of 16 refocusing parameters during the training phase.

4.2 Related Work

4.2.1 Refocusing techniques

Here I review some of the previously developed refocusing methods in the literature. Isaksnen *et al*. introduced some early algorithms to extract features like refocusing, synthetic apertures, and occlusion removals from LFs. Their refocusing algorithm is usually referred as brute force refocusing or reparametrization of LF [75]. This method has time complexity of $O(n^4)$ and making it efficiently parallel is difficult. This method is slow, particularly because it needs to apply a homography to the whole LF for each requested focal plane. The shift-and-sum LF refocusing method was first introduced by Vanish *et al*. [76] and later extended by Levoy *et al*. [7]. This method first derives a mapping for the LF capturing device and the refocused images are extracted by warping the views in the LF using the calculated map. The quantized form of the shift-and-sum method was first derived by Ng *et al*. [9]. But the shift-and-sum method still has an asymptotic time-complexity of $O(n^4)$ and is not useful for real-time refocusing. *Fourier slice refocusing* (FLR) was introduced by Ng [17] in an attempt to speed-up the refocusing. Refocusing of the LF using FLR has the time complexity of only $O(n^2) \log n$, which is almost ideal for real-time refocusing. But the problem with this method is that it needs to calculate the 4-D

Fourier transform of each LF. In the discrete world, the 4-D fast Fourier transform's asymptotic time complexity is $O(n^4 \log n)$. Because of this pre-processing step, the LF cannot be refocused in real-time immediately after capture. Also, the refocused images extracted by the FLR method is inferior to the shift-and-sum method at some focus points because taking $FT^{-}1(FT(x))$ is not always equal to x in discrete world. Nava *et al.* [77] addressed the problem of FLR's quality by using generalization of the discrete Radon transform in the process. Their discrete focal stack transform produces better quality refocused images compared to FLR, but at the cost of reducing the number of possible refocusing focus points. In both shift-and-sum method and FLR, the camera array or the physically accurate conversion of the Plenoptic camera to camera array [18] should be on a single plane. If the cameras are placed on different planes the tilt-shift method can be used for refocusing [7, 78]. Other focus based features of LFs include multi-focus image extraction [79] and super-resolved refocused images [80, 81]. While these scenarios are not the focus of this paper, my method can be easily applied to these situations as well.

4.2.2 ML Assisted Refocusing

Depth estimation from single [18, 44] or sparse LF views [1] and LF reconstruction followed by using conventional refocusing techniques to demonstrate the quality of the LF have been thoroughly investigated. Deep learning has been used to extract

refocused images using a single image as well [82, 83]. In stereo cameras, because of sparsity in the angular domain, the number of focal planes for refocusing are limited. Neural network have been employed to extract and enhance the refocused images for stereo cameras too [84, 85]. However, I could not find any work in the literature targeting the particular problem of refocusing LFs captured by plenoptic cameras using machine learning.

4.3 The Proposed Method

In this section, my procedure for extracting refocused images from an LF is described. I have introduced a network for this purpose called RefNet. RefNet has the ability to extract 16 refocused images at different but static, predetermined focal planes.

4.3.1 RefNet

Our network is constructed by employing 11 layers of modified ResNet50 *bottelneck* building blocks [69], or ResBlocks. These blocks, shown in Fig. 4.1, have been used for the main feature extraction. For filter dimensionality change, 2-D convolutions with (1×1) kernels have been used. The overall structure of the network is depicted in Fig. 4.2.

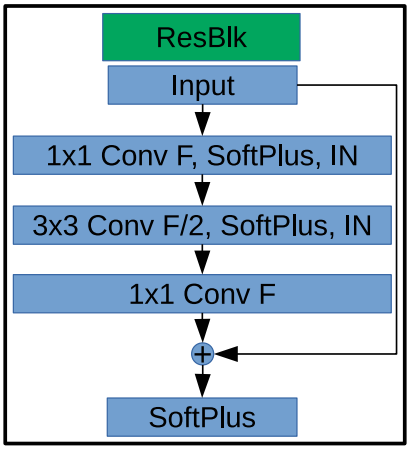


Figure 4.1: Structure of each ResBlock. Conv is short for 2-D convolution. The kernel size of each convolution is noted in the beginning of each cell. F represents the number of filters of each convolution. IN is the instance normalization.

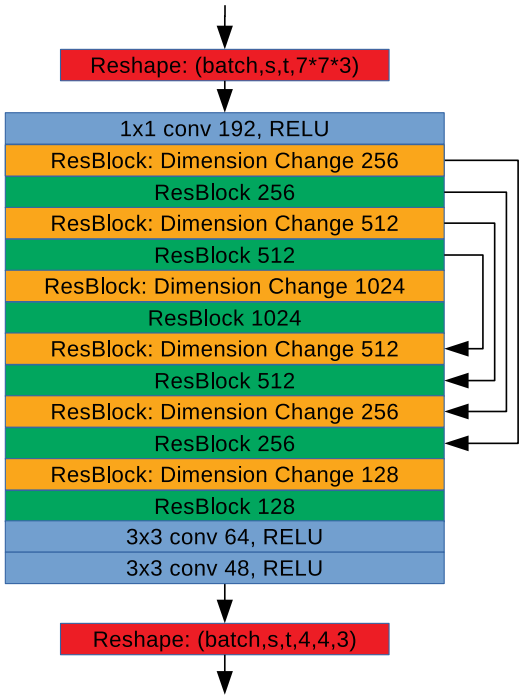


Figure 4.2: The detailed structure of RefNet. First the LF is reshaped to $s, t, u \times v \times 3$, where in my experiment $u, v = 7$. I used a 2-D convolution with 192 filters as the first layer of the network followed by a 11 ResBlocks. Then with two additional convolution with RELU activation the final dimensionality is reached. Finally, with a reshape, 16 estimated refocused image extracted.

Fig. 4.1 illustrates that each ResBlock is composed of a 2-D convolution with kernel size of (3×3) and $F/2$ channels sandwiched between two 2-D convolution layers with (1×1) kernels and F channels. Each of the first two convolutions are followed by a *Softplus* activation function and an instance normalization. The input has a skip connection to the last convolution and then a Softplus activation is applied to the summed result. If the number of channels of the input differs from F, it will go through an additional 2-D convolution with (1×1) kernel and F channels, then, it's result will be added to the last convolution.

For a detailed illustration of the RefNet structure, see Fig. 4.2. First the LF is reshaped to $s, t, u \times v \times 3$. I used a 2-D convolution with 192 channels as the first layer of the network with an RELU activation; this is followed by 11 ResBlocks. The network then has two additional 2-D convolution layers with RELU activation. Finally, after a reshape, 16 refocused images are estimated as the network output.

The RefNet is trained in supervised manner. I use a weighted loss function for the purpose of training. The training output labels are the refocused images extracted from the LFs in my learning set by either the Fourier slice or shift-and-sum methods. Next I describe the loss function.

4.3.2 Loss function

Our loss function is a weighted sum of the *mean squared error* (MSE) and ℓ_1-norm between the predicted and true images (as provided by the conventional reconstructions), and also the image quality losses as measured by *structural similarity index metric* (SSIM) [70] and *peak signal to noise ratio* (PSNR).

The first component of the loss function the appearance matching loss [86], which combines the SSIM with the ℓ_1-norm,

$$\psi_1(I_p, I_g) = \beta \frac{1 - SSIM(I_p, I_g)}{2} + (1 - \beta) \left\| I_p - I_g \right\|_1, \tag{4.1}$$

where I_p and I_g are the predicted and ground truth refocus images, respectively, and β is a tuning parameter. The second component is the inverse PSNR,

$$\psi_2(I_p, I_g) = \frac{1}{PSNR(I_p, I_g)}. \tag{4.2}$$

These two components are then added to MSE to form the final version of the loss function,

$$L = MSE(I_p, I_g) + \psi_1(I_p, I_g) + \gamma \psi_2(I_p, I_g). \tag{4.3}$$

The tuning parameter γ allows the magnitude of the inverse PSNR in ψ_2 to be tuned

relative to the quantities in MSE and ψ_1, which I found to be helpful in overall performance. I provide the values of β and γ I used in my experiments in Section 4.4.

4.3.3 Metrics

For quantitative evaluation of the predictions, I used the SSIM and PSNR metrics. The average PSNR and SSIM over all of the focus points of one LF are calculated as

$$MSSIM = \frac{1}{M} \sum_{u} SSIM([I_p]_u, [I_g]_u), \tag{4.4}$$

$$MPSNR = \frac{1}{M} \sum_{u} PSNR([I_p]_u, [I_g]_u), \tag{4.5}$$

where $[I_p]_u$ represents the uth refocused image extracted by my model and $[I_g]_u$ is the respective ground truth. M is the total number of extracted refocused images.

4.4 Experiments

In this section, I describe my method's implementation details for the experiments. Then I use public LF data sets [1, 44, 87] to evaluate my method and investigate the impact of different parts of my network on the performance of my model.

4.4.1 Data sets

I have conducted my experiments over three publicly available data sets: *Flowers* [44], *30 Scenes* [1], and *Stanford Lytro Light Field Archive* [87]. I chose these data sets because all three of them are captured by Lytro Illum cameras, which is the most widely used LF camera. I cropped the LF from these data sets to the size of $7 \times 7 \times 375 \times 540$ to have a consistent size and vignetting. We chose 648 specific LFs from the Flowers data set, which were chosen to span a diverse set of viewing angles and scenes of the flowers. I did this because I noticed a lot of the LFs of the flowers were captured from the same scene with minimal camera movement. Another reason was to balance the number of flower scenes with other categories to have a balanced training set. Overall, my training data set contains $1,101$ LFs: 648 flowers, 100 LFs from the *30 Scenes*, and the 353 scenes from the *Stanford archive*. I used the 30 LFs of the *30 Scenes test set* to evaluate my results; note that these 30 LFs are typically used as a benchmark set for LF reconstruction and are not contained in the training data.

4.4.2 Implementation details

We used the Fourier slice theorem and shift-and-sum method to pre-process the training and test sets. Two new data sets of refocused images were created by extracting 16 refocused images with $\alpha = \{0.125, 0.250, 0.375, ..., 2\}$ from each LF. During the training phase, the LFs and the corresponding refocused images were randomly cropped to $[192, 192, 7, 7]$. This cropping technique was used as an augmentation to the training data to prevent over-fitting the network during the training phase. I used a batch size of 4 crops of each LF during a single epoch. RefNet has about 12.5 million trainable parameters. The networks were trained by minimizing the loss function using the ADAM optimizer [88] for 90 epochs in my experiments. Each epoch took approximately 8 minutes to complete; hence, the total amount of time spend for training was 13 hours. In the training phase, $\beta = 0.65$ was empirically found to yield the best results. And $\gamma = 500$ performed well for training the network.

We implemented my model with Tensorflow 2.3 in python 3.7 on a workstation with an Intel Xeon W-2223 3.60 GHz, 64GB DDR4 memory, and an NVIDIA Quadro RTX 5000.

4.4.3 Performance

We compared my model's result with shift-and-sum and the Fourier slice method. Note that none of these methods can be claimed to be the actual ground truth refocused images. To obtain actual ground truth images, physical cameras would have to be used to capture images at different focus points. But such data are not available; hence, I compared my results with these benchmark approaches and evaluated the results visually as well. First I compared the my results quantitatively using PSNR and SSIM metrics, then I provided the refocused images for qualitative comparison as well. Please look at the electronic version of the paper for better quality assessment of images. In my supplementary materials, the full results of refocusing on the whole test set is available.

The MSSIM and MPSNR values for the predicted refocused images of my model compared to the Fourier slice and shift-and-sum methods are plotted in Fig. 4.3. When the Fourier slice method is used as the ground truth for training, I can see that, most of the mean MSSIM values are more than 92.5%, and there are only two of them with a value less than 90%. I will show that RefNet's refocused images for these two LFs are better estimates compared to the FLR. For the case of MPSNR, the results are a bit worse than MSSIM in term of fluctuations. The majority of the MPSNR values are more than 30dB; however, at least seven trials show MPSNR values

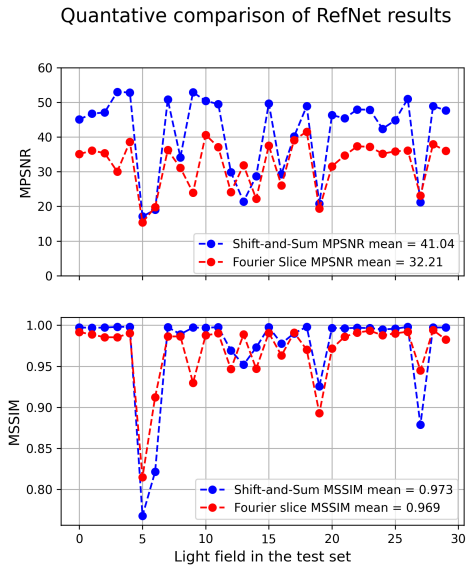


Figure 4.3: The MPSNR and MSSIM of the refocused images extracted from RefNet, compared to the shift-and-sum and Fourier slice methods. Blue points represent the MPSNRs and MSSIMs when the shift-and-sum refocused images are used as ground truth, while the red points represent the MPSNR and MSSIM of the RefNet refocused images predicted by the RefNet when it is trained on the Fourier slice method's refocused images. The set of refocused images for calculating the MPSNR and MSSIM are 16 refocused images with different focus points.

of less than 25dB. Similar behavior is observed for when the ground truth images are shift-and-sum refocused images. In this case, the average MSSIM and MPSNR is higher than the Fourier slice method, but there are still some trials that seem like outliers. By carefully looking at Fig. 4.3, I can see that some of the trials have high MSSIM and low MPSNR. This behavior—low MPSNR and high MSSIM—suggests a large color mismatch between the two sets but with a high structure similarity, which means they are focused roughly at the same focus point. Figs. 4.4 and 4.5 demonstrate this impression.

For qualitative comparison of the refocused images, I have compared refocused images extracted from RefNet trained with both the Fourier slice and shift-an-sum images for three different focus points, $\alpha = (0.125, 1.0, 2.0)$. See Figs. 4.4 and 4.5. The chosen LFs are number 5 and 16 in the test set. These LFs are selected because they are among the 5 trials in Fig. 4.3 where MPSNR is around or below 20dB, indicating that the RefNet and ground truth are very different in either brightness or color. LF number 5 shows significant brightness differences and LF number 16 shows color differences. Similar figures that compare the refocused images for all of the 30 LFs in the test set are available in the supplementary materials.

In both of these figures, I can see that RefNet and the ground truth methods seemingly have the same DoF and focus point but they have distinction in color and brightness. Fig. 4.4 shows that the refocused images from the Fourier slice method are significantly brighter than the DoF image without refocusing, while the brightness is not consistent among the three different focus points. For the shift-and-sum method, the brightness is consistent between different focus points but all of them are still brighter than the DoF image without refocusing while it has better predictions compared to the Fourier slice method. The RefNet trained on Fourier slice labels is performing almost similar to the shift-and-sum refocused images in brightness prediction and consistency. I can see that the RefNet with shift-and-sum labels has the best performance visually and the brightness of the RefNet refocused images are almost identical to the DoF image without refocusing. So, in terms of robustness in predictions, both trained

RefNet networks are performing better than the Fourier slice method in terms of visual quality, while being able to produce these refocused images in real-time.

In Fig. 4.5 I see that the RefNet approach is superior in terms of color reconstruction. The only difference is that the color of the predicted refocused images from RefNet that is trained by shift-and-sum labels is not identical to the DoF without refocusing, but it is the closest compared to other methods.

4.4.4 Speed comparison

A truly fair speed comparison of FLR and RefNet is out of scope of this paper, as FLR requires a 4-D *fast Fourier transform* (FFT) while the available GPU based FFTs are at most 3-dimensional. Thus, I cannot implement the FLR method on GPU without tweaking the available tools and further optimizing for 4-D FFTs; though, I recognize this could be accomplished. Nor did I find an available GPU based implementation of the FLR. Hence, it was not possible to compare the speeds of the algorithms on the exact same GPU.

For the FLR method, the average time spent for 4-D FFT using NumPy's fftn [89] over 1,000 samples, when the LF was loaded into CPU RAM, was 0.5137 seconds over just the R channel. So the whole RGB LF would need 1.5411 seconds just for the pre-processing step of the FLR. For RefNet, when the network is loaded into the

GPU's memory and the LF is in CPU RAM (i.e., not yet loaded to GPU), a single forward propagation of one LF with all 3 RGB channels which extracts 16 refocused images takes 0.0209 seconds. This is at least 73× speed up compared to the pre-processing step of the FLR. If the LFs are grouped in batches, the speed gain of RefNet will be further improved. Using RefNet, a batch of 4 LFs can be processed, with 16 refocused images for each, in 0.046 seconds. This is over 134× faster than FLR. The significance of this speedup enables the possibility of real-time processing of LF videos concurrently with capturing. It is worth noting that the whole process of reading each LF from the hard drive to the final extraction of refocused images takes ~ 0.2 seconds per LF, which is still about an order of magnitude faster than just the pre-processing step of FLR, where, if the LF is being read from the hard drive, takes 1.85 seconds.

4.5 Conclusions and Future Work

In this chapter I have proposed a machine learning technique, called RefNet, to extract refocused depth of field images from a light field captured using a plenoptic camera. I demonstrated that by using my method, a light field can be refocused in real-time without pre-processing on conventional desktop GPUs. Furthermore, my method was demonstrated to be more visually pleasing in terms of color prediction when compared to the Fourier slice and shift-and-sum methods. RefNet is capable of extracting

16 refocused images with refocusing parameters of $\alpha = 0.125, 0.250, 0.375, ..., 2.00$, though other refocusing parameter values could be trained for given proper training data. Possible future work includes exploring methods for training a network that can refocus light fields for any refocusing parameter, perhaps even using training data composed of other refocusing parameter values. Another avenue that can be explored is to extract multi-focus images from the light field using machine learning, which could enable 3-D object detection or reconstruction techniques.

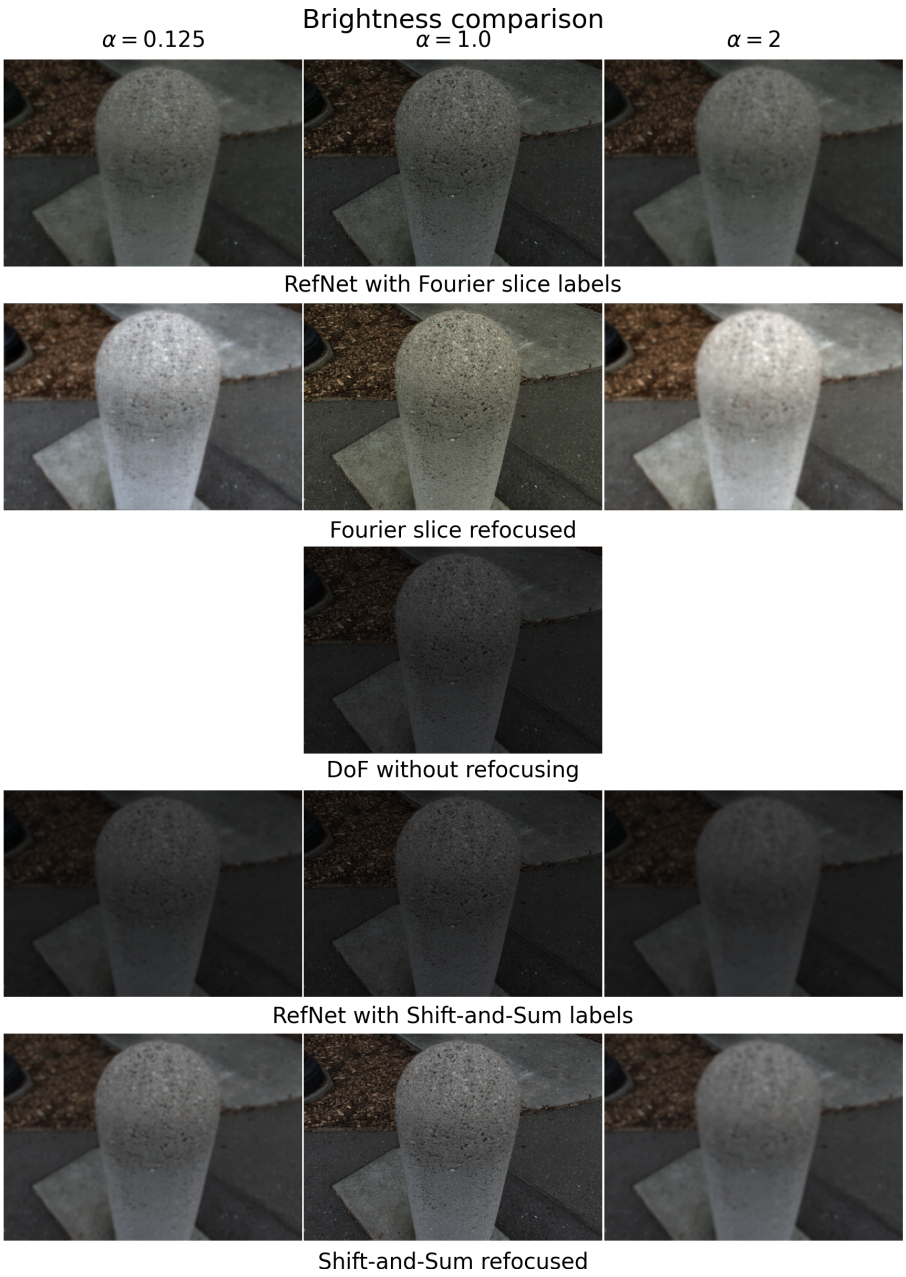


Figure 4.4: In this figure, three refocused images from each refocusing method are shown. Each row contains one method's refocused images with refocusing parameters of $\alpha = 0.125$, $\alpha = 1.0$ and $\alpha = 2.0$ from left to right. In the middle, an unfocused DoF image is provided for visual comparison of the predicted brightness. From top to bottom, the used methods for refocusing are RefNet with Fourier slice ground truth, Fourier slice method, RefNet with shift-and-sum ground truth, and shift-and-sum method. The LF used for this figure is the fifth LF in the 30 Scenes data set [1].

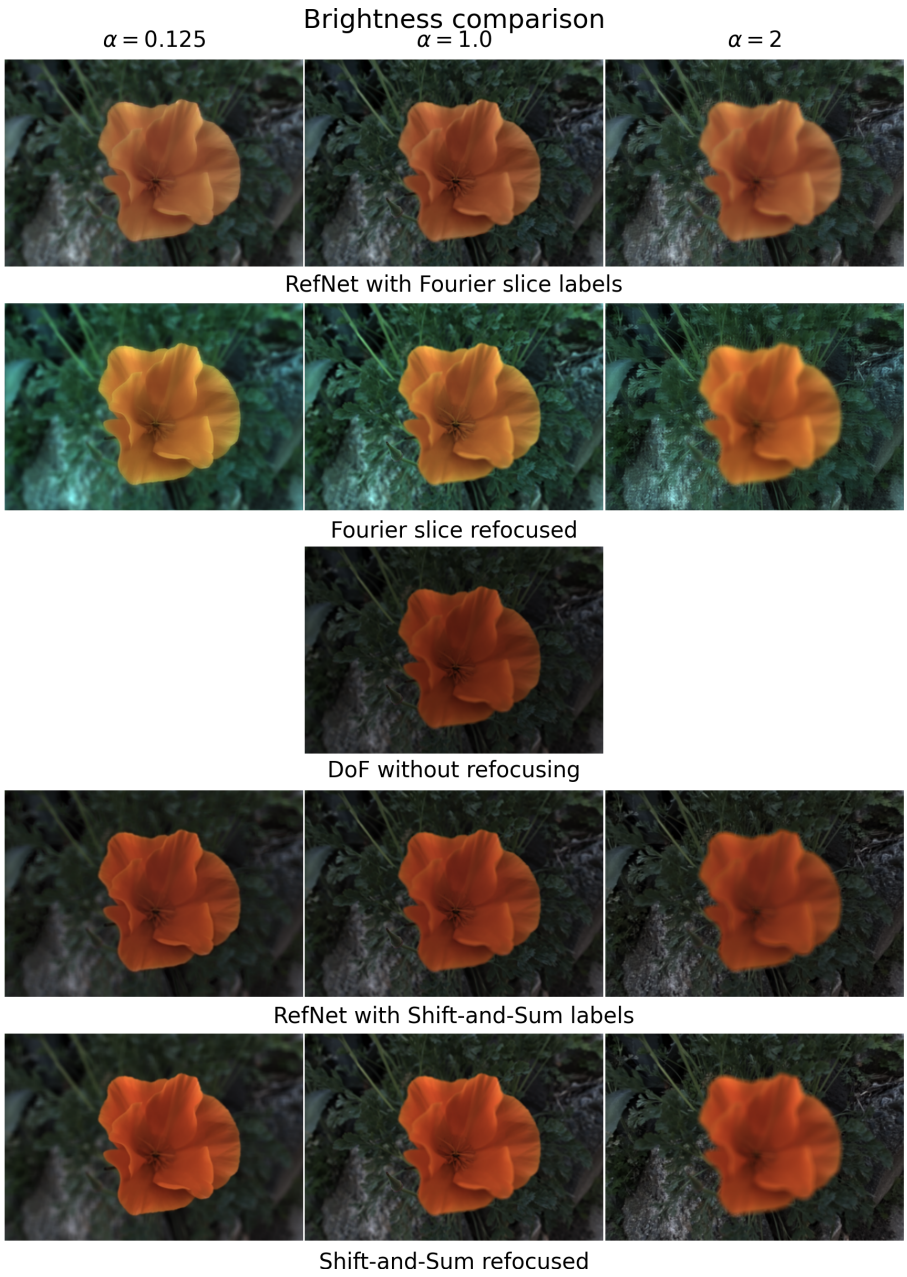


Figure 4.5: In this figure, three refocused images from each refocusing method are shown. Each row contains one method's refocused images with refocusing parameters of $\alpha = 0.125$, $\alpha = 1.0$ and $\alpha = 2.0$ from left to right. In the middle, an unfocused DoF image is provided for visual comparison of the predicted colors. From top to bottom, the used methods for refocusing are RefNet with Fourier slice ground truth, Fourier slice method, RefNet with shift-and-sum ground truth, and shift-and-sum method. The LF used for this figure is the 16th LF in the 30 Scenes data set [1].

Chapter 5

Conclusion

In this dissertation, it was shown that Machine Learning Methods are very efficient LF Compression and refocusing. A transformation function between PCs and CAs is physically derived in chapter 2. Using that transformation function, simulation of high angular and spatial resolution LF in any physically rendering software is possible. These high-resolution simulated LFs have large are very bulky. We provide an ML algorithm for lossy compression and decompression of bulky LFs In chapter 3 to overcome this problem. Finally, in chapter 4 a real-time LF refocusing algorithm has been developed by use of residual CNNs. My new algorithm, increased the quality of refocused images while being faster than state-of-the-art. The function and algorithms described in this dissertation can be used by researchers to improve the quality of LFs both in capture and postprocessing. Also, the refocusing methods may be used for

the detection task of choice by making some changes to the dataset.

5.1 Summary of findings

The body of this dissertation provides the following key results. First, a transformation model was physically derived by the use of geometrical optics that allows finding the placement of cameras in the array to build a CA with the capability of capturing an LF similar to one that is captured by the SPC of choice including how to apply vignetting to the pristine simulated LF. This model is developed by considering the thin lens approximation. The model is validated by first conducting simulations in Blender software. Then, using the refocusing parameter, some refocused images are extracted. The refocusing distance of them was compared to the estimated distances derived from the Plenoptisign software [34] and proofed to have similar distances within the expected measurement error. Second, an ML decompression network is trained that is capable of estimating the entire LF from a single-center view of LF that is compressed by the JPEG lossy algorithm at 50% quality. This algorithm is faster in compression by 100 times while being at least 18 times faster in decompression compared to the state-of-the-art video encoder used compression techniques when the quality and the bit rate are kept the same. On the other hand, it is shown that the PSNR and SSIM of the postprocessing extractions like refocused images from the proposed algorithm's reconstructed LF has higher values compared to the

state-of-the-art. This superiority has been visually confirmed as well. Third, a novel ML technique is proposed that uses residual CNN types of network to extract a set of 16 refocused images with refocusing parameters of $\alpha = 0.125, 0.250, 0.375, ..., 2.00$ from a raw LF in real-time without pre-processing. The set of refocused images have better color prediction compared to the other available refocusing techniques while being faster. The dataset used for training and testing RefNet, JPEG-Hance, and Depth-Net are the public datasets captured specifically by the Lytro Illum camera.

5.2 Possible future extensions

While I have been able to address some of the problems in LF processing to some extent, there are still so many unanswered problems that remained untouched. Also, further improvements to the proposed algorithm are possible. As an example, the transformation function proposed in this dissertation for conversion of SPC to CAs can be further improved to take compound lenses into account and go beyond the thin lens approximation. Also, the transformation can be expanded to other types of PCs like FPCs.

While I have been able to propose a fast LF compression algorithm, the quality of reconstructed LF needs to be improved while maintaining the speed. Using the same idea, LF video compression can be explored too. The proposed network structure and

loss functions also can be improved. Using simulated LFs and LF videos, the dataset can be expanded for achieving a better result.

My refocusing algorithm is the first ML method for refocusing LFs captured by a PC. The next step would be to remove the constraint of predefined refocusing parameters. Also, the current network can refocus any light field with angular resolution of 7×7. One important extension is to make it suitable for any number of angular resolution. On the other hand, generalization to other types of LF capture is also an important avenue that should be explored. Other future paths that could be explored are extracting multi-focus compositions from LF using ML methods. Such refocused images can be included in an object detection pipeline to improve the detection quality.

References

[1] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi, "Learning-based view synthesis for light field cameras," *ACM Trans. Graph.*, vol. 35, no. 6, 2016.

[2] E. H. Adelson and J. R. Bergen, *The Plenoptic Function and the Elements of Early Vision*, vol. 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1991.

[3] G. Lippmann, "La photographie intégrale," *Comptes-Rendus, Académie des Sciences*, vol. 146, pp. 446–551, 1908.

[4] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), p. 31–42, Association for Computing Machinery, 1996.

[5] V. Vaish, M. Levoy, R. Szeliski, C. L. Zitnick, and Sing Bing Kang, "Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust

measures," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2331–2338, June 2006.

[6] S. McCloskey, "Masking light fields to remove partial occlusion," in *2014 22nd International Conference on Pattern Recognition*, pp. 2053–2058, Aug 2014.

[7] M. Levoy, B. Chen, V. Vaish, M. Horowitz, I. McDowall, and M. Bolas, "Synthetic aperture confocal imaging," *ACM Trans. Graph.*, vol. 23, no. 3, p. 825–834, 2004.

[8] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High performance imaging using large camera arrays," *ACM Trans. Graph.*, vol. 24, pp. 765–776, July 2005.

[9] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, P. Hanrahan, *et al.*, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, no. 11, pp. 1–11, 2005.

[10] T. G. Georgiev and A. Lumsdaine, "Focused plenoptic camera and rendering," *Journal of Electronic Imaging*, vol. 19, no. 2, p. 021106, 2010.

[11] R. S. Overbeck, D. Erickson, D. Evangelakos, and P. Debevec, "The making of welcome to light fields vr," in *ACM SIGGRAPH 2018 Talks*, pp. 1–2, ACM, 2018.

[12] C. Perwass and L. Wietzke, "Single lens 3d-camera with extended depth-of-field," in *Human Vision and Electronic Imaging XVII*, vol. 8291, p. 829108, International Society for Optics and Photonics, 2012.

[13] T. Michels, A. Petersen, L. Palmieri, and R. Koch, "Simulation of plenoptic cameras," in *2018 - 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–4, June 2018.

[14] D. Liu, L. Wang, L. Li, Zhiwei Xiong, Feng Wu, and Wenjun Zeng, "Pseudo-sequence-based light field image compression," in *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–4, 2016.

[15] C. Conti, P. Nunes, and L. D. Soares, "Hevc-based light field image coding with bi-predicted self-similarity compensation," in *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–4, 2016.

[16] Y. Li, R. Olsson, and M. Sjöström, "Compression of unfocused plenoptic images using a displacement intra prediction," in *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–4, 2016.

[17] R. Ng, "Fourier slice photography," in *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, (New York, NY, USA), p. 735–744, Association for Computing Machinery, 2005.

[18] E. Hedayati and J. P. Bos, "Modeling standard plenoptic camera by an equivalent camera array," *Optical Engineering*, vol. 59, no. 7, pp. 1 – 14, 2020.

[19] E. Hedayati, T. C. Havens, and J. P. Bos, "Light field compression by residual cnn assisted jpeg," 2020.

[20] E. Hedayati, T. C. Havens, and J. P. Bos, "Machine learning method for light field refocusing," *arXiv preprint arXiv:2103.16020*, 2021.

[21] A. Jones, I. McDowall, H. Yamada, M. Bolas, and P. Debevec, "An interactive 360 light field display," in *ACM SIGGRAPH 2007 emerging technologies*, p. 13, ACM, 2007.

[22] Y. Anisimov, O. Wasenmüller, and D. Stricker, "A compact light field camera for real-time depth estimation," in *Computer Analysis of Images and Patterns* (M. Vento and G. Percannella, eds.), (Cham), pp. 52–63, Springer International Publishing, 2019.

[23] K. Marwah, G. Wetzstein, Y. Bando, and R. Raskar, "Compressive light field photography using overcomplete dictionaries and optimized projections," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 46, 2013.

[24] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke, "A dataset and evaluation methodology for depth estimation on 4d light fields," in *Asian Conference on Computer Vision*, Springer, 2016.

[25] E. Hedayati and J. P. Bos, "Simulation of light fields captured by a plenoptic camera using an equivalent camera array," in *Laser Communication and Propagation*

through the Atmosphere and Oceans VII, vol. 10770, p. 107700R, International Society for Optics and Photonics, 2018.

[26] D. G. Dansereau, *Plenoptic Signal Processing for Robust Vision in Field Robotics*. Doctor of philosophy ph.d., Graduate School of Engineering and IT; School of Aerospace, Mechanical and Mechatronic Engineering, 2013-08-27.

[27] C. Hahne, A. Aggoun, S. Haxha, V. Velisavljevic, and J. C. J. Fernández, "Baseline of virtual cameras acquired by a standard plenoptic camera setup," in *2014 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–3, July 2014.

[28] C. Hahne, A. Aggoun, V. Velisavljevic, S. Fiebig, and M. Pesch, "Baseline and triangulation geometry in a standard plenoptic camera," *International Journal of Computer Vision*, vol. 126, pp. 21–35, Jan 2018.

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

[30] C. Hahne and A. Aggoun, "Plenoptisign: An optical design tool for plenoptic imaging," *SoftwareX*, vol. 10, p. 100259, 2019.

[31] C. Perwaß and L. Wietzke, "Single lens 3D-camera with extended depth-of-field," in *Human Vision and Electronic Imaging XVII* (B. E. Rogowitz, T. N. Pappas, and H. de Ridder, eds.), vol. 8291, pp. 45 – 59, International Society for Optics and Photonics, SPIE, 2012.

[32] D. G. Dansereau, O. Pizarro, and S. B. Williams, "Decoding, calibration and rectification for lenselet-based plenoptic cameras," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1027–1034, 2013.

[33] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[34] C. Hahne, A. Aggoun, V. Velisavljevic, S. Fiebig, and M. Pesch, "Refocusing distance of a standard plenoptic camera," *Opt. Express*, vol. 24, pp. 21521–21540, Sep 2016.

[35] Blender Foundation, "Blender cycle renderer." `https://www.cycles-renderer.org/`.

[36] eMirage, "Blender demo files." `https://www.blender.org/download/demo-files/`.

[37] D. Wittmann, "Blender models." `https://www.turbosquid.com/`.

[38] Z. Zhao, S. Wang, C. Jia, X. Zhang, S. Ma, and J. Yang, "Light field image compression based on deep learning," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2018.

[39] M. Podpora, G. P. Korbas, and A. Kawala-Janik, "Yuv vs rgb-choosing a color space for human-machine interaction.," in *FedCSIS (Position Papers)*, pp. 29–34, 2014.

[40] A. Levin and F. Durand, "Linear view synthesis using a dimensionality gap light field prior," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1831–1838, 2010.

[41] T. E. Bishop and P. Favaro, "The light field camera: Extended depth of field, aliasing, and superresolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 972–986, 2012.

[42] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[43] H. Wing Fung Yeung, J. Hou, J. Chen, Y. Ying Chung, and X. Chen, "Fast light field reconstruction with deep coarse-to-fine modeling of spatial-angular clues," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[44] P. P. Srinivasan, T. Wang, A. Sreelal, R. Ramamoorthi, and R. Ng, "Learning to synthesize a 4d rgbd light field from a single image," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[45] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, and J. Kautz, "Extreme view synthesis," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[46] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Trans. Graph.*, vol. 38, no. 4, 2019.

[47] W. Zhou, G. Liu, J. Shi, H. Zhang, and G. Dai, "Depth-guided view synthesis for light field reconstruction from a single image," *Image and Vision Computing*, vol. 95, p. 103874, 2020.

[48] Z. Hu, Y. Y. Chung, W. Ouyang, X. Chen, and Z. Chen, "Light field reconstruction using hierarchical features fusion," *Expert Systems with Applications*, vol. 151, p. 113394, 2020.

[49] C. Perra, "Lossless plenoptic image compression using adaptive block differential prediction," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1231–1234, IEEE, 2015.

[50] P. Helin, P. Astola, B. Rao, and I. Tabus, "Sparse modelling and predictive coding of subaperture images for lossless plenoptic image compression," in *2016 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–4, IEEE, 2016.

[51] X. Jiang, M. Le Pendu, R. A. Farrugia, and C. Guillemot, "Light field compression with homography-based low-rank approximation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 7, pp. 1132–1145, 2017.

[52] E. Dib, M. L. Pendu, and C. Guillemot, "Light field compression using fourier disparity layers," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 3751–3755, 2019.

[53] I. Tabus, P. Helin, and P. Astola, "Lossy compression of lenslet images from plenoptic cameras combining sparse predictive coding and jpeg 2000," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 4567–4571, 2017.

[54] M. L. P. X. Jiang and C. Guillemot, "Light field compression using depth image based view synthesis," in *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 19–24, 2016.

[55] J. Zhao, P. An, X. Huang, L. Shan, and R. Ma, "Light field image sparse coding via cnn-based epi super-resolution," in *2018 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, 2018.

[56] B. Wang, Q. Peng, E. Wang, K. Han, and W. Xiang, "Region-of-interest compression and view synthesis for light field video streaming," *IEEE Access*, vol. 7, pp. 41183–41192, 2019.

[57] C. Jia, X. Zhang, S. Wang, S. Wang, and S. Ma, "Light field image compression using generative adversarial network-based view synthesis," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 177–189, 2019.

[58] D. Liu, X. Huang, W. Zhan, L. Ai, X. Zheng, and S. Cheng, "View synthesis-based light field image compression using a generative adversarial network," *Information Sciences*, vol. 545, pp. 118 – 131, 2021.

[59] J. Hou, J. Chen, and L. Chau, "Light field image compression based on bi-level view compensation with rate-distortion optimization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 517–530, 2019.

[60] J. Wang, Q. Wang, R. Xiong, Q. Zhu, and B. Yin, "Light field image compression using multi-branch spatial transformer networks based view synthesis," in *2020 Data Compression Conference (DCC)*, pp. 397–397, 2020.

[61] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[62] J. Guo and H. Chao, "Building dual-domain representations for compression artifacts reduction," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 628–644, Springer International Publishing, 2016.

[63] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, and T. S. Huang, "D3: Deep dual-domain based fast restoration of jpeg-compressed images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[64] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 2802–2810, Curran Associates, Inc., 2016.

[65] L. Cavigelli, P. Hager, and L. Benini, "Cas-cnn: A deep convolutional neural network for image compression artifact suppression," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 752–759, 2017.

[66] B. Zhang, J. Gu, C. Chen, J. Han, X. Su, X. Cao, and J. Liu, "One-two-one networks for compression artifacts reduction in remote sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 184 – 196, 2018. Deep Learning RS Data.

[67] X. Zhang, W. Yang, Y. Hu, and J. Liu, "Dmcnn: Dual-domain multi-scale convolutional neural network for compression artifacts removal," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 390–394, 2018.

[68] Z. Jin, M. Z. Iqbal, W. Zou, X. Li, and E. Steinbach, "Dual-stream multi-path recursive residual network for jpeg image compression artifacts reduction," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2020.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[70] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[71] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[72] Y. Wang, Y. Yang, Z. Yang, L. Zhao, P. Wang, and W. Xu, "Occlusion aware unsupervised learning of optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[73] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[74] M. Ren, R. Liu, H. Hong, J. Ren, and G. Xiao, "Fast object detection in light field imaging by integrating deep learning with defocusing," *Applied Sciences*, vol. 7, no. 12, 2017.

[75] 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, (USA), p. 297–306, ACM Press/Addison-Wesley Publishing Co., 2000.

[76] V. Vaish, G. Garg, E. Talvala, E. Antunez, B. Wilburn, M. Horowitz, and M. Levoy, "Synthetic aperture focusing using a shear-warp factorization of the viewing transform," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, pp. 129–129, 2005.

[77] F. P. Nava, J. G. Marichal-Hernández, and J. M. Rodríguez-Ramos, "The discrete focal stack transform," in *2008 16th European Signal Processing Conference*, pp. 1–5, 2008.

[78] C. Xiao, J. Yang, Y. Wang, and W. An, "Reprojection-based method for camera arrays of refocusing onto arbitrary focal surfaces," in *Fifth Conference on Frontiers in Optical Imaging Technology and Applications* (J. Chu, W. Liu, and H. Jiang, eds.), vol. 10832, pp. 237 – 242, International Society for Optics and Photonics, SPIE, 2018.

[79] S. Sugimoto and M. Okutomi, "Virtual focusing image synthesis for user-specified image region using camera array," in *2008 19th International Conference on Pattern Recognition*, pp. 1–4, 2008.

[80] F. Pérez, A. Pérez, M. Rodríguez, and E. Magdaleno, "Fourier slice super-resolution in plenoptic cameras," in *2012 IEEE International Conference on Computational Photography (ICCP)*, pp. 1–11, 2012.

[81] T. Georgiev, G. Chunev, and A. Lumsdaine, "Superresolution with the focused plenoptic camera," in *Computational Imaging IX* (C. A. Bouman, I. Pollak, and P. J. Wolfe, eds.), vol. 7873, pp. 232 – 244, International Society for Optics and Photonics, SPIE, 2011.

[82] P. P. Srinivasan, R. Garg, N. Wadhwa, R. Ng, and J. T. Barron, "Aperture supervision for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[83] P. Sakurikar, I. Mehta, V. N. Balasubramanian, and P. J. Narayanan, "Refocusgan: Scene refocusing using a single image," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[84] H. Chou, K. Shih, and H. Chen, "Occlusion-and-edge-aware depth estimation from stereo images for synthetic refocusing," in *2018 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–6, 2018.

[85] Z. Nian and C. Jung, "Cnn-based multi-focus image fusion with light field data," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 1044–1048, 2019.

[86] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[87] D. G. Dansereau, "Stanford lytro light field archive," 2016.

[88] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[89] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[90] M. Ren, R. Liu, H. Hong, J. Ren, and G. Xiao, "Fast object detection in light field imaging by integrating deep learning with defocusing," *Applied Sciences*, vol. 7, no. 12, 2017.

[91] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[92] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[93] H. G. C. Werij, J. E. M. Haverkort, and J. P. Woerdman, "Study of the optical piston," *Phys. Rev. A*, vol. 33, pp. 3270–3281, May 1986.

[94] D. Dansereau and L. Bruton, "Gradient-based depth estimation from 4d light fields," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol. 3, pp. III–549, May 2004.

[95] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[96] C. Hahne, *The Standard Plenoptic Camera: Applications of a Geometrical Light Field Model*. PhD thesis, Univ. of Bedfordshire, January 2016.

[97] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz, "Realistic modeling and rendering of plant ecosystems," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, (New York, NY, USA), pp. 275–286, ACM, 1998.

[98] Matt Pharr, Wenzel Jakob, and Greg Humphreys, "Pbrt v2 scenes." `https://pbrt.org/scenes-v2.html`.

[99] Z. Zhang, Y. Liu, and Q. Dai, "Light field from micro-baseline image pair," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Appendix A

Letters of Permission

A.1 Optical Engineering Journal

Dear Eisa,

Thank you for seeking permission from SPIE to reprint material from our publications. SPIE shares the copyright with you, so as author you retain the right to reproduce your paper in part or in whole.

Publisher's permission is hereby granted under the following conditions:

1. the material to be used has appeared in our publication without credit or acknowledgment to another source; and

2. you credit the original SPIE publication. Include the authors' names, title of paper, volume title, SPIE volume number, and year of publication in your credit statement.

Best wishes on your dissertation. Please let me know if I may be of any further assistance.

Best,

Katie Sinclair

Editorial Assistant, Publications

SPIE – the international society for optics and photonics

katies@spie.org

1 360 685 5436

A.2 IEEE Article Sharing and Posting Policies

IEEE generally allows authors to reproduce their work in their own thesis or dissertation, explained here.