<table>
<thead>
<tr>
<th>Title</th>
<th>4D light field reconstruction and scene depth estimation from plenoptic camera raw images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Shan, Xu</td>
</tr>
<tr>
<td>Publication Date</td>
<td>2016-05-16</td>
</tr>
<tr>
<td>Item record</td>
<td><a href="http://hdl.handle.net/10379/6006">http://hdl.handle.net/10379/6006</a></td>
</tr>
</tbody>
</table>

Some rights reserved. For more information, please see the item record link above.
4D Light Field Reconstruction and Scene Depth Estimation from Plenoptic Camera Raw Images

by

Shan Xu

Supervisor: Dr. Nicholas Devaney

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

School of Physics, Science Faculty, National University of Ireland, Galway

April 2016
Abstract

Microlens-array-based light-field cameras (also known as plenoptic cameras), which are capable of recording both angular and spatial information of light, are already commercially available as consumer commodities. Due to the fact that we are still using the 2D planar sensor to record the 4D light field, extra optical design and post-processing considerations are required to reconstruct a high-quality light field.

In this thesis, three research problems are investigated: light-field camera simulation, light field reconstruction, and depth from light field. First, we propose a framework for computational camera simulation utilising computer graphic rendering software. Using this simulation framework, we analyse image formation in the microlens-array-based light-field camera. Second, we present a reconstruction algorithm that is able to reconstruct the 4D light field from a portable plenoptic camera without the need for calibration images. Our quality assessment shows that our proposed approach achieves comparable results to the state-of-the-art algorithms that require calibration images. Third, we present theoretical analysis of depth resolvability for the microlens-array-based light-field camera. A simple and efficient implementation of depth from light field algorithm, in particular, a novel and effective refinement scheme for Lytro camera is proposed. We demonstrate that our approach outperforms the state-of-the-art algorithms and commercial software.
I would like to thank my supervisor, Dr. Nicholas Devaney for his guidance and support throughout this thesis work. Your consistent encouragement, advice and patience have inspired me to finish my thesis. Also, I would like to thank Professor Chris Dainty. Your series of lectures on optics and imaging are as fantastic as the most popular ones from Coursera.

My sincere thanks to all my colleagues in the Applied Optics Group, NUIG: Dr. Mingzhou Chen, Dr. David Mackey, Mr. Joseph Gallagher, Dr. Huanqing Guo, Mr. Colm Lynch, Mr. Conor Sheil, Ms. Niamh Fitzgerald, to name a few. My special thanks to my former colleagues Dr. Barry Mccullagh, Dr. Alexandru Drimbarean, Dr. Mariusz Madry and Dr. Piotr Stec in Fotonation. Another special thanks go to the lovely Irish lady, Ms. Geraldine Shaughnessy who provided me not only a permanent place but also a warm family to stay with during my hard times.

My sincere thanks to Professor Bastian Goldluecke for offering me the academic visiting opportunity in the light field analysis group, Heidelberg University where I spent a memorable summer in 2013. And my sincere thanks to Professor Guijin Wang for offering me the academic visiting opportunity in the Visual Computing Group, Tsinghua University in 2014. I am also very grateful to Dr. Zhiliang Zhou and Mr. Xuanwu Yin. During my visit, we had so many pleasant and inspirational discussions.

I must acknowledge my wife, without whom I would have never even considered pursuing a Ph.D. degree. There would be little chance to accomplish this thesis without your and our son’s accompany. Last but not the least, I would like to thank our parents for everything they have done for us.
The way ahead is long and has no ending;
yet high and low I'll search with my will unbending.

Qu Yuan  (340–278 BC)
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>ii</td>
</tr>
<tr>
<td>Preface</td>
<td>1</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>3</td>
</tr>
<tr>
<td>1.1 Light Field Imaging Theory</td>
<td>4</td>
</tr>
<tr>
<td>1.1.1 Light Field Transport</td>
<td>5</td>
</tr>
<tr>
<td>1.1.2 Light Field Projection</td>
<td>6</td>
</tr>
<tr>
<td>1.2 Light Field Acquisition Devices</td>
<td>7</td>
</tr>
<tr>
<td>1.3 Applications of Light Field Imaging</td>
<td>14</td>
</tr>
<tr>
<td>1.3.1 Photography</td>
<td>14</td>
</tr>
<tr>
<td>1.3.2 Depth Sensing</td>
<td>16</td>
</tr>
<tr>
<td>1.3.3 Microscopy</td>
<td>16</td>
</tr>
<tr>
<td>1.3.4 Optical Aberration Correction</td>
<td>18</td>
</tr>
<tr>
<td>1.3.5 Object Recognition</td>
<td>20</td>
</tr>
<tr>
<td>1.3.6 Light Field Probe</td>
<td>21</td>
</tr>
<tr>
<td>1.3.7 Particle Image Velocimetry</td>
<td>21</td>
</tr>
<tr>
<td>1.4 Other Related Technology</td>
<td>22</td>
</tr>
<tr>
<td>1.4.1 Wavefront Sensing</td>
<td>22</td>
</tr>
<tr>
<td>1.4.2 Holography</td>
<td>23</td>
</tr>
</tbody>
</table>
Preface

Recent advances in computing power and optical fabrication provide us with the opportunity to record the world in a more efficient and effective manner than the conventional imaging system. Computational imaging has emerged as a vibrant and cross-disciplinary field of research. As a branch of computational imaging, light field imaging is attracting increasing attention. Specifically, the development of microlens-array-based light-field cameras makes capturing an instant light field possible and more importantly, convenient and inexpensive.

Motivation

The motivation for this thesis is threefold. First, for the purpose of understanding light field image formation, we develop a simulation framework to evaluate the performance of optics and algorithms for computational cameras. Second, due to the presence of a microlens array and extra information is embedded into the 2D raw image data, the calibration of a microlens-array-based light-field camera requires a more sophisticated process; we present a novel, efficient calibration algorithm enabling no reference image 4D light field reconstruction. Third, since depth estimation is one of the most important applications of light field imaging, we propose a fast and accurate depth estimation framework.

Synopsis

In Chapter 1, we briefly describe the background, development, applications of light field imaging, and a detailed introduction to the Lytro camera, from which the raw data has been used throughout the thesis. In Chapter 2, we implement a computa-
tional camera simulation framework and give three application examples. The framework enables joint evaluation of optics and algorithms. In Chapter 3, we introduce a novel approach to reconstruct the light field without the need for the uniform illuminated light field images. We also propose to use joint local and global fitting to model the vignetting for the plenoptic raw images. In Chapter 4, we first study how to extract the depth information from light field. Next, we present a novel and efficient method of estimating disparity from light field. In Chapter 5, we summarize the work of the whole thesis and discuss possible further work.

Publications


Chapter 1

Introduction

Looking back at the history of photography, the earliest development of an imaging system was a camera obscura, a device dating back to the ancient Chinese and ancient Greeks over two thousand years ago. Since the invention of photographic film in the 19th century, imaging technology has been continuously evolving. Today, digital cameras are ubiquitous, ranging from mobile phones to surveillance cameras. However, the principle of the traditional camera remains fundamentally the same; the photograph is a 2D projection of the 3D scene.

In contrast to conventional imaging, computational imaging is a vibrant and cross-disciplinary field which enables a broad range of novel imaging applications. In recent decades, the advances of technology, especially optical engineering and computation power, provide us the opportunity to record the world in a more efficient and effective manner than conventional imaging systems. By jointly designing an optical system and post computational algorithms, we are able to optically encode information such as high dynamic range and scene geometry in the capture process and decode it afterward with computation.
1.1 Light Field Imaging Theory

The photograph produced by either a pinhole camera or a conventional camera is a 2D projection of the 3D objects inside a pyramid as illustrated in Fig. 1.1. It is more intuitive to observe the projection relation between the plane $\Pi$ in object space, which is a mirror image of the plane $\Pi'$, and the object plane $\Omega$. For the lens camera, the intensity value of point $p'$ on the image plane $\Pi'$ is the irradiance integration of rays passing through the lens (finite aperture) from the corresponding point $o$ on the object plane $\Omega$. The lens camera achieves a better signal to noise ratio than a pinhole camera but at the expense of losing the angular information of the incoming rays. In contrast, light field imaging is an imaging method which aims to preserve the angular information of each individual light ray.

![Figure 1.1: A typical pinhole camera diagram. The conjugate image plane $\Pi$ is the projected plane of the 3D world encapsulated in the pyramid with perspective point $E$.](image)

**Representation** In order to completely characterize a light ray in 3D space, Adelson and Bergen [1] proposed a 7D plenoptic function which includes the ray origin $(x,y,z)$, direction $(\theta,\psi)$, wavelength $\lambda$ and the time $t$. Without considering time and wavelength, McMillan and Bishop [2] reduce it to a 5D function for the purpose of image-based rendering. Since in geometrical optics, rays travel along straight lines and remain constant in free and transparent space, the representation of the light field can be further simplified to a 4D light field [3][4]. The way of representing light fields in 4D significantly simplifies the computational complexity.

As illustrated in Fig. 1.2, two planes $XY$ and $UV$ are parallel to each other and have a unit length separation. $XY$ and $UV$ are usually referred to as the spatial coordinates...
Chapter 1. Introduction

and angular coordinates respectively. Mathematically, the 4D light field is a map,

$$L : XY \times UV \mapsto \mathcal{R} \quad (x, y, u, v) \mapsto l(x, y, u, v)$$

(1.1)

Since the separation between the two planes is a unit of length, the \((u, v)\) coordinates are equal to the tangent of the ray angles with respect to the optical axis. This is the reason why \((u, v)\) coordinates are usually regarded as the angular coordinates. In Fig.1.2 to visualize the ray transport, rays are coloured with a gradually changing colour palette. Furthermore, we assume that the rays emanate from the XY plane with uniform radiance within a solid angle. In the following, for the sake of simplicity, we only discuss 2D light fields. The 2D light field spectrum is denoted as \(\hat{l}(\Omega_x, \Omega_u)\) which is the 2D Fourier transform of its spatial domain counterpart \(l(x, u)\).

1.1.1 Light Field Transport

Using the 4D light field representation to characterize the light rays offers a new perspective to analyse how the light rays transport through the imaging system and form an image on the sensor plane. Durand et al. [5] introduced a 4D light field transport analysis in the frequency domain for the purpose of efficient 3D rendering. The framework was later extended to image formation [6] and performance analysis [7,8]. Typically, as illustrated in Fig.1.3 there are three types of light field transport: free

1 spatial frequency of the intensity distribution, not the frequency of the light waves.
Chapter 1. Introduction

space propagation, occlusion, and refraction.

![Diagram of light field transport](image)

Figure 1.3: (a) Free space propagation. (b) Occlusion. Some light rays are blocked by the mask (aperture). (c) Refraction. Light rays are bent by the thin lens.

These three types of light field transport are illustrated in Fig.1.4 in both the spatial domain and the frequency domain. In the spatial domain, the propagated light field is a horizontally sheared version of the original light field, whereas, in the frequency domain, the propagated light field spectrum is a vertical shearing of the original light field spectrum. In contrast to free space propagation, the refracted light field is a vertically sheared version of the original light field, whereas, in the frequency domain, the propagated light field spectrum is a horizontal shearing of the original light field spectrum. For the occlusion case, the original local light field is trimmed by a binary mask in the spatial domain. As a result, the spectrum of the original light field is replicated (convolved) by the spectrum of the binary mask.

### 1.1.2 Light Field Projection

The image formation process can be modeled as a projection of the 4D light field onto a 2D planar sensor. For simplicity let us consider the 2D light field. For the conventional camera, with the 2D light field representation, the intensity of the pixel is simply a vertical integration of all the rays that have the same spatial location as illustrated in Fig.1.5(b). As opposed to the conventional camera which is only able to capture the 2D projection of the 4D light field, the benefit of having the light field is that we can artificially manipulate the light field (discussed in the previous section) and produce synthetic images. For example, the refocusing effect can be obtained by propagating the local light field to a certain distance before the projection. This is equivalent to using the projection along a slant line instead of vertical integration, as illustrated in Fig.1.5(a),(c).
Chapter 1. Introduction

1.2 Light Field Acquisition Devices

Although the idea of capturing light fields dates back to more than a century ago by the Nobel prize winner, Gabriel Lippmann [9], commercially available products have only appeared on the consumer market recently, produced by the companies Lytro and Raytrix. This is because a light field device not only relies on a high resolution sensing device and high precision optical components but also requires a high density memory storage and powerful computing capability. In general, according to the multiplexing schemes, the light field acquisition devices can be classified into three categories: temporal, spatial, and frequency multiplexing. This is summarized in
Table 1.1

<table>
<thead>
<tr>
<th>Multiplexing Scheme</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Programmable aperture camera [12] [13] [14]</td>
</tr>
<tr>
<td></td>
<td>Mobile phone [15]</td>
</tr>
<tr>
<td>Spatial multiplexing</td>
<td>Camera array [16]</td>
</tr>
<tr>
<td></td>
<td>MLA-based light-field camera [17] [18]</td>
</tr>
<tr>
<td></td>
<td>Miniaturized camera array module [19]</td>
</tr>
<tr>
<td></td>
<td>Compressive light-field camera [20]</td>
</tr>
<tr>
<td></td>
<td>Add-on module [21]</td>
</tr>
<tr>
<td></td>
<td>Lens-prism light-field camera [7]</td>
</tr>
<tr>
<td></td>
<td>Spherical Catadioptric light-field camera [22]</td>
</tr>
<tr>
<td>Frequency multiplexing</td>
<td>Heterodyned light-field camera [23]</td>
</tr>
</tbody>
</table>

Table 1.1: Light field acquisition devices with different multiplexing schemes.

**Temporal Multiplexing**  Probably the simplest way to capture 4D light fields with a 2D image sensor is to take a sequence of images with different perspectives using for example a camera gantry [3] capturing light fields. Kim et al. [11] also proposed to use a professional camera with a motorized slider to acquire high spatial-angular resolution light fields. Leveraging the rapid development of mobile phone technology, Davis et al. [15] proposed to interactively acquire and render light fields by moving an off-the-shelf mobile phone around the object. In their design, users are able to check if enough views have been taken from real-time feedback on the mobile phone. For
Chapter 1. Introduction

each shot, the camera pose and orientation are estimated using a Simultaneous Localization and Mapping (SLAM) library designed for augmented reality applications.

Instead of moving the camera, Liang et al. [12] proposed to change the shape of the aperture, which is equivalent to shifting the perspective of the viewpoint, to capture the full resolution light fields. This is known as the programmable aperture camera. Rather than only opening a small region of the aperture each time, they optimized the aperture pattern to improve the light efficiency by Hadamard Coding. For the post processing, a novel photometric calibration algorithm is employed to reduce the vignetting effect across different views. In their prototype, a low-cost passive liquid crystal display (LCD) module with removed reflective layer is inserted into the light path to electronically control the shape of the aperture. A limitation is that the transmission of the LCD pixel is low, and it cannot be turned off completely. The alternative options are Liquid Crystal on Silicon (LCoS) [13] and Digital Mirror Devices (DMD) [14]. However, these components are expensive and need a more complicated optical setup to be integrated into the light path as shown in 1.7.

![Image removed due to copyright issues](image)

**Figure 1.7:** Left, a LCoS based coded aperture camera prototype [13]. Right, the optical schematic. A beam splitter and relay lenses are used to integrate a reflective LCoS into the optical path.

The temporal multiplexing scheme enables the capture of high-quality light fields, but it also requires a static scene. The images need to be registered accurately in order to reconstruct the light field.

**Spatial Multiplexing** To capture a dynamic scene or a light field video, a camera array was built by Wilburn et al. [16]. Their camera array consists of 10 by 10 cameras with a 640 by 480 spatial resolution for each camera. Each image sensor is connected
to a field programmable gate array (FPGA) and the video stream is transferred to PC through a FireWire® 1394 interface. In fact, this camera array supports flexible physical configurations such as physical separation between each camera, exposure time and shutter trigger synchronization. With different configurations, it is able to not only capture light fields but also high speed, high dynamic, high resolution, wide-angle images. They also demonstrated seeing objects behind partial occlusions such as brushwood by adaptively selecting rays that do not intersect the occluders; this is known as synthetic aperture photography [24]. However, in order to record a high quality light field a camera array needs careful calibration including individual camera geometric calibration (orientation and position), radiometric calibration and colour calibration to compensate for variations among the lenses and sensors. In contrast to the cumbersome camera array, a miniaturized camera array was recently developed by Pelican Imaging [19]. It enables capture of a high quality 4D light field in a snap shot using mobile devices. To make their miniaturized camera array module as thin as possible, a 4 by 4 lens array is used to replace the conventional lens stack and the auto-focus actuator is absent. Since their camera array module is not able to adjust focus, the aperture size of each lens is deliberately reduced to have a large depth of field from a few millimeter to infinity. The side effect of using small apertures is that signal-to-noise ratio is low and the image quality of each individual camera is limited by diffraction. Using a super-resolution algorithm for a camera array similar to the one proposed by Carles et al. [25], they claimed a factor of 2.4 overall image quality improvement by fusing all 4 by 4 sub-aperture images into one high-resolution and low-noise image. However, this does not overcome the diffraction limit of the individual lenses.

A single-shot portable light-field camera was first proposed by Adelson and Wang [18] in 1992. This is the first modern optical device to record light fields by placing an MLA behind the main lens inside a conventional camera. Their work was focused on estimating the depth of objects in the scene and they described their device as a single lens stereo camera. A decade later, Ng et al. [17] prototyped a portable light-field camera specifically for photographic applications e.g. digital refocusing. They used a customized MLA of 296 by 296 microlenses. A customized aluminum lens holder was designed to ensure precise separation between the MLA and the image sensor. This is the early prototype of the Lytro camera which has been recognized as a milestone in the evolution of photography. The major drawback of this type of light field device is the trade-off between spatial and angular information which is caused by the nature of the spatial multiplexing scheme. The spatial resolution is limited by the size of the microlenses, which leads to the quality of the light field reconstructed image being
inferior to even a low-cost mobile phone camera. This type of MLA-based light-field camera is referred as to the plenoptic 1.0 camera.

In order to overcome this critical drawback of the MLA-based light-field camera, a few improved optical setups and advanced light field reconstruction algorithms have been proposed. Ng [26] proposed a generalized light-field camera setup. He proposed that the spatial and angular resolution of the MLA-based light-field camera can be dynamically alternated by mechanically changing the separation between the MLA and the image sensor. For example, when the separation between MLA and sensor is zero, theoretically we have the maximum spatial resolution at the expense of losing all angular resolution. However, repeatably accurate alignment between microlens array and image sensor is difficult. Alternatively, Lumsdaine and Georgiev [27] proposed to shift the MLA from the focal plane of the main lens to the image plane of the main lens. This optical setup, which is referred to as the plenoptic 2.0 camera, can be seen as a micro camera array focusing on the image plane of the main lens. The Raytrix camera, which is based on this optical configuration, uses microlenses with three different focal lengths for the purpose of further extending the depth of field [28]. Fig.1.8 shows the optical setup of plenoptic 1.0 and plenoptic 2.0 cameras. The plenoptic 1.0 camera directly captures the 4D light fields which are characterized by the exit pupil plane and the microlens array plane. In contrast to the plenoptic 1.0, the plenoptic 2.0 camera indirectly captures the light field. Therefore, Wanner et al. [29] proposed an algorithm to reconstruct the light field captured by the plenoptic 2.0 camera. The spatial resolution of the plenoptic camera 2.0 is increased at the expense of losing angular resolution. A detailed discussion of spatio-angular trade-off of plenoptic cameras can be found in [7]. As shown in Fig.1.9, the raw image of plenoptic 1.0 camera looks blurry, whereas the raw image of plenoptic 2.0 looks sharper and contains fine features. Recently, Boominathan et al. [30] used a hybrid light-field camera system which includes a Lytro camera and a Canon 18 megapixel DSLR camera to acquire high-quality light fields. The result is plausible, but the proposed system is not feasible for common use due to cost and complexity of operations. From the algorithm point of view, Bishop and Favaro [31] theoretically derived the PSF model for a generalized MLA-based light-field camera. To reconstruct the light field, they proposed a deconvolution algorithm under a variational Bayesian framework with statistical priors including Lambertianity and texture statistics.

As illustrated in Fig.1.10, similar to the existing multiplexing methods imposed on image sensor pixels, such as the Bayer filter [32] or spatially varying exposure pixel [33], the angle sensitive pixel device was proposed by Hirsch et al. [34].
Figure 1.8: Plenoptic camera designs. (a) Plenoptic 1.0 camera setup \([17]\). The whole exit pupil is imaged by a microlens and a single pixel corresponds to a subregion of the exit pupil. (b) Plenoptic 2.0 camera setup \([27]\). The image plane of the main lens is re-imaged by the microlens (blue shading). \(f_1\) and \(f_2\) are the focal lengths of the microlenses of the plenoptic camera 1.0 and 2.0, respectively. In practice, \(d_1\) is smaller than \(d_2\) and \(a,b\) satisfies the lens formula so that the microlens focuses at the focal plane of the main lens.

Figure 1.9: (a) The ISO 12233 chart is the object in the scene. (b) A crop of plenoptic 1.0 camera raw image captured by us using our Lytro camera. The detailed specification of Lytro camera will be introduced in the following section. (c) A crop of Plenoptic 2.0 camera raw image captured by us using our Ratrix camera R5 with a 35mm focal length lens. The size of each crop is 640 by 640.
Chapter 1. Introduction

Figure 1.10: Left is the Bayer filter sensor layout. R, G, B represents the red, green and blue colour filter respectively. Middle is the interleaved pixel HDR sensor layout. S, L represents the short and long exposure respectively. Right is angle sensitive pixel sensor layout. Each pixel is only sensitive to a certain range of projection angles of the incoming light.

placing a phase grating with different orientations on top of the sensor, each pixel is directionally sensitive to the incoming light. The light field is reconstructed with a compressive sensing algorithm. Alternatively, Marwah et al. [20] applied the compressive sensing algorithm on a LCoS coded-aperture camera to reconstruct the 4D light field. Rather than modifying or adding elements inside a conventional camera, Georgiev [7] built a light-field camera by placing a lens-prism array in front of the camera. Similarly, Taguchi et al. [22] proposed a wide-angle light-field camera using a conventional camera looking at a spherical mirror array.

Frequency Multiplexing  Borrowing the idea of 1D signal transmission in telecommunications, where the base-band signal can be modulated by a high frequency carrier at the transmitter side and then demodulated with a reference carrier at the receiver side, Veeraraghavan et al. [23] prototyped a light field capturing device named Dappled Photography or heterodyne light-field camera. They assumed the light field signals have a narrow spatial and angular frequency band so that it is possible to replicate angular frequency components to the high spatial frequency regions. Unfortunately, in the real world, the spatial signal contains high frequencies. In the experiment, to achieve a plausible result they used a flatbed scanner, of which the spatial resolution is much higher than today’s high-end professional cameras. Another drawback is that the mask reduces the light efficiency and introduces diffraction effects. From the algorithm side, the appropriate light field prior was studied in their paper for the purpose of reducing ringing effects introduced by the frequency manipulation.
1.3 Applications of Light Field Imaging

Light field imaging expands the dimensional space of conventional imaging. This enables a variety of novel applications, both photographic and scientific. In the following, several popular and emerging light field applications will be discussed. In addition, over the last couple of years, as the twin technology of light field imaging, the light field display [35–39] has been drawing increasing interest from both academia and industry, specifically leveraging compressive sensing technology [40, 41]. The rapid development of light field imaging provides the light field display with solid support in terms of media source. We can imagine a scenario which might become reality in the near future; a doctor performs surgery with a light field endoscope observing the interior of the patient while using the real-time augmented light field visualization with a light field display.

1.3.1 Photography

**Digital Refocusing** Since the 4D light field of the scene is digitally captured by the light-field camera, we can propagate the recorded 4D light field and render an image at a synthetic plane by a 2D projection; this operation is referred to as digital refocusing. In other words, having the 4D light field allows us to simulate the optical refocusing digitally. However, this is sometimes confused with the image processing based approach, such as the Android’s app Lens Blur provided by Google Camera [42], which selectively blurs the content of the image according to the depth information to produce bokeh and shallow depth effect. The depth knowledge is normally estimated from a focal stack image sequence [43, 44] or multiple-view images [45] of the static scene. In contrast, light field digital refocusing only requires a snapshot with a single camera.

Digital refocusing may be appealing to amateur photographers. It is a common problem when people try to capture a moment without accurate focus adjustment, so that the object of interest is blurred in the photo. Ng et al. [17] first demonstrated digital refocusing using the 4D light field data captured from an MLA-based light-field camera. His work has finally resulted in a consumer product, the Lytro camera for which the main feature is interactive refocusing. As shown in the Fig. 11, the 2D images reconstructed from the Lytro camera can be focused digitally by simply clicking the mouse button on the region of interest. With a light-field camera, refocusing becomes a post-processing step and an interactive entertainment process.
Chapter 1. Introduction

Figure 1.11: Left, digital refocus at the near object. Middle, digital refocus at the far object. Right, all-in-focus synthetic image. Above results are produced using Lytro Desktop 5.0.1.

For the same optical settings, such as aperture size and focal length, Ng et al. [17] experimentally demonstrated that the light-field camera can provide a larger depth of field (DOF) than the conventional camera provided that the light-field camera and the conventional camera have same spatial sampling rate. The DOF extension is achieved by manipulating a focal stack of images which is obtained by digitally projecting the 4D light fields to the 2D images with varying refocusing parameters. We demonstrate this comparison quantitatively in Section 2.2.3. To artificially synthesize an extended depth of field image, a graph-cut [46] algorithm was used to adaptively select the focused objects from the focal stack of images and merge them into an all-in-focus image. In some machine vision applications where there are objects over a large depth range, the camera needs a large depth of focus. Decreasing the aperture of the camera is not possible when the ambient light is low. To address the problem, the light-field camera has been explored for human iris recognition [47] and bar-code recognition [48] as a DOF extended camera.

Parallax effect If we think of refocusing as moving the observer back and forth along the optical axis, then it is also possible to move the observer laterally by projecting the light field in a different manner, this effect is referred to as parallax motion [49].
Chapter 1. Introduction

Others

Besides the applications mentioned above, there are also some other photographic applications with a light-field camera. Ramesh et al. [50] proposed to remove the glare which arises due to multiple reflections between lens elements in strong light condition (e.g. sun) which reduces the image contrast severely. They found it is possible to remove it as high frequency noise in the 4D light field frequency domain.

1.3.2 Depth Sensing

In the last two decades, inferring the depth of objects from images has been a very active field in computer vision. Not surprisingly, the first hand-held light-field camera [18] was designed to detect depth, and was known as a single lens stereo camera. Zhou and Nayar [51] theoretically analyse the depth sensing principle with a light-field camera and they conclude that the estimation result should be more reliable than a stereo camera. Wanner et al. [49] proposed to use the structure tensor to analyse the orientation of the 2D epipolar stripes, the slope of which are proportional to the depth disparity. The final result was optimized by variational filtering. Can et al. [52] also demonstrated that the light-field camera is able to handle significant occlusions in depth estimation using a bilateral consistency metric. As shown in Fig.1.12, both Picam [19] and motorized DSLR [11] demonstrated fine depth estimation results from the captured light fields. This is investigated in details in Chapter 4.

1.3.3 Microscopy

Levoy et al. [53] first proposed light field microscopy by inserting a microlens array into the optical path of a microscope. It overcomes the main limitations of the conventional microscope such as the fact that the view is limited to the orthographic view and the depth of field is shallow. In their prototype, a relay lens was employed to match f-number of the objective lens and the microlens as shown in Fig.1.13. Their prototype is able to produce a focal stack and different perspectives of biological samples. For semi-transparent samples, they also demonstrated 3D volume reconstruction by a 3D deconvolution with a synthetic PSF. By introducing another microlens and a Digital Mirror Device(DMD), Levoy et al. [54] demonstrated a microscope with 4D light field controlled illumination. They also experimentally demonstrated moderate correction of aberrations by controlling the 4D light field illumination. Instead of using geometrical optics, Broxton et al. [55] proposed a light field transport model based on wave optics and the assumptions are that the light is emitted isotropically.
from all fluorophores in the volume and travels directly to the microscope objective without any further scattering or refraction. Using the numerical approximation of the proposed model, the volume of a fluorescent biological sample was reconstructed by an improved 3D deconvolution. In their experiment, in order to have sufficient angular sampling rate, there are roughly 27 x 27 pixels behind each microlens. They demonstrated a pollen grain volume reconstruction where the spatial resolution is improved by a factor from 2 to 4 depending on how far the reconstructed plane is offset from the focused plane along the optical axis as shown in Fig. 1.13. To mitigate the non-uniformity of lateral resolution along the optical axis, Cohen et al. [56] proposed to use wavefront coding which is achieved by placing phase masks in the optical path of the light field microscope. They demonstrated a flatter modulation transfer function (MTF) along the optical axis than the one shown in [55]. To capture the high-resolution light fields, Lu et al. [57] proposed to use a hybrid system which consists of a light field microscope and a conventional microscope with a beam-splitter. The drawback is that the system is more complicated and the light efficiency is reduced by half.
1.3.4 Optical Aberration Correction

In reality, no optical system is perfect and all exhibit different types of optical aberrations leading to reduction in image quality. An optical aberration removal solution based on image processing is deconvolution \([58, 59]\), which has been implemented in commercial software, such as *DxO Optics Pro* and *Adobe Photoshop*. From the signal processing perspective, the blurred image can be considered as a perfect image convolved with a spatially varying blur kernel or Point Spread Function (PSF) in the spatial domain. In the frequency domain, the lens quality is usually measured...
by Modulation Transfer Function (MTF) which describes the spatial frequency re-
sponse. Deblurring as an inverse operator is not able to recover signals which have 
been completely lost. For example, aberrations can cause zeros in the MTF, and these 
frequencies cannot be recovered (they may however be estimated in schemes employ-
ing prior information about the object, such as positivity or finite support). To design 
a robust deblurring algorithm, the key element is the prior model of natural images, 
which allows us to partially recover the information lost by convolution. Heide et 
al. \cite{60} demonstrated using a single home-made convex lens to achieve professional 
lens quality images using deconvolution. Their prior model included an image gra-
dient sparsity prior and a cross-colour-channel gradient correlation prior. Heide et 
al. \cite{61} further integrated deblurring and other camera processing blocks into a uni-
ified and flexible camera processing framework enabling real-time optical aberration 
removal.

Instead of image post-processing, aberrations can be removed optically using adap-
tive optics \cite{62}. The working principle is illustrated in Fig.1.14. The wavefronts are 
measured and corrected dynamically by a closed-loop control system. The advan-
tage of adaptive optics over image deconvolution is that worse SNR images can be 
restored as the zeros in the MTF are removed by correction of the aberrations, and the 
MTF is boosted. The adaptive optics system is able to correct aberrations in the optical 
path such as atmospheric distortion for telescopes or refractive index variation in 
specimens for microscopes. Simple Adaptive optics is limited to the field over which 
the PSF is constant.

Figure 1.14: The working principle of the adaptive optics.

Using light field technology to digitally remove lens aberrations was proposed by 
Hanrahan and Ng \cite{63}. The idea is to digitally resort aberrated rays to where they
Chapter 1. Introduction

should be terminated. As illustrated in Fig. 1.15, a double convex lens suffers strong spherical aberration. Instead of integrating along the vertical axis to obtain the image intensity in the 4D light field, the unaberrated image intensity should be obtained by integrating along a curve which can be determined by a ray tracing software based on a model of the specific lens system. However, this requires a light field sensor with sufficient angular sampling rate. On the other hand, the sensor spatial resolution should be beyond a certain level so that the optical aberration becomes the bottleneck that limits the spatial resolution. To meet both criteria, the resolution of the image sensor needs to be much larger than those employed in today’s highest end professional cameras. Considering current sensor technology and cost, using the deconvolution with good priors and measured point spread function (PSF) \cite{60} to reduce optical aberrations might be a better option than the light field re-raytracing approach.

1.3.5 Object Recognition

Object recognition is a well explored topic in computer vision. However, current feature descriptors such as Scale-Invariant Feature Transform (SIFT) \cite{64} or Speeded Up Robust Features (SURF) \cite{65} cannot be applied to transparent objects as the background information will be taken into account. To address this problem, Maeno et al. \cite{66} proposed a light field feature descriptor based on the fact that transparent objects have the unique characteristic of distorting the background by refraction. According to their experiment, the identification rate was improved significantly for transparent objects.
1.3.6 Light Field Probe

Wetzstein et al. [67] employed commodity hardware consisting of a camera and a specific illumination device to optically visualize the refractive change in a transparent object in a technique known as Schlieren photography. In their experiment, the key component is the light field probe which consists of a uniform illumination source, a printed transparency film with specific designed mask, and a lenslet array. By applying different types of mask on the transparency film, the illumination device is able to produce different forms of light fields as well as colour and intensity variations. As illustrated in Fig. 1.16, this idea is further extended to reconstruct the surface of the transparent object [68]. They proposed to use colour gradients and geometric constraints to find accurate ray-ray correspondences and calculate the surface normal by Snell’s law.

![Light Field Probe Diagram](image)

Figure 1.16: The light field probe consists of a LED-based illumination source, a transparency film and a lenslet sheet. Ray-ray correspondence is determined by the colour and geometric constraints.

1.3.7 Particle Image Velocimetry

Particle image velocimetry (PIV) is an imaging method for flow visualization used mainly in fluid dynamics experiments. The fluid is seeded with small tracer particles which can be illuminated by a light source. The movement of the particles is assumed to faithfully represent the local 3D velocity field. Belden et al. [69] first demonstrated a 3 by 3 camera array to resolve 3D vector fields in densely seeded simulated flows using synthetic aperture. Later, Lynch et al. [70] and Garbe et al. [71] proposed to use MLA-based light-field cameras with different optical configurations to measure three-dimension three-component (3D3C) velocity.
1.4 Other Related Technology

1.4.1 Wavefront Sensing

Wavefront sensing is a technology that measures the aberrations of a wavefront of light. It is commonly used in adaptive optics [62] systems in astronomy and recently extended to the field of microscopy [72] and ophthalmology. The Shack-Hartmann wavefront sensor (SHWFS) is the most popular wavefront sensing device. It consists of an MLA and an image sensor. The MLA is usually placed at the plane which is conjugated to the aperture plane of the telescope to measure the aberration of the wavefront.

As shown in Fig. 1.17, the wavefront distortion is estimated by the spot displacements, which are proportional to the derivative of wavefront aberrations [73]. Actually, the MLA-based light-field camera is capable of measuring the shape of the wavefront at exit pupil plane of the main lens. Both Clare and Lane [74] and Rodriguez-Ramos et al. [75] demonstrated 4D light fields to wavefront conversion. However, as pointed out by Clare and Lane [74], the design considerations for sensing a wavefront and light fields are mutually opposed. In an MLA-based light-field camera design, the spatial resolution is usually larger than the angular resolution by more than an order of magnitude to ensure good image quality near the spatial sampling plane. In contrast, for wavefront sensing, the plane of interest shifts from the focal plane to the exit pupil plane of the main lens. In other words, the angular and spatial coordinates are swapped for sensing a wavefront and 4D light fields with the MLA-based light-field camera (plenoptic 1.0 camera). This is the reason why wavefront sensing results [74, 75] using current MLA-based light-field cameras suffer low spatial resolution on the wavefront.

![Wavefront Sensing Diagram](image)

Figure 1.17: The wavefront aberration can be estimated by determining the displacement of the spots with respect to the microlens optical centre.
1.4.2 Holography

In contrast to light field imaging which is based on geometric optics, holography is based on wave optics. It is a technique that records interference patterns from monochromatic coherent light waves which are scattered from the object and a reference beam. The 3D geometric structure of the object can be reconstructed by retrieving the amplitudes and phases of the reflected light waves from the recorded interference fringes. Although holography and light field are completely different technologies, both methods can record the 3D geometry of the scene. Ziegler et al. [76] proposed a framework to transform light fields to hologram and hologram to light fields. Cossairt et al. [77] demonstrated a prototype incoherent holography camera capable of performing digital refocusing.

1.5 Lytro Camera

In this section, we give an introduction to the Lytro camera, which was launched in 2011, as the world’s first consumer light-field camera. As a consumer product, the information about the Lytro camera is not all publicly available. We include here our knowledge of the Lytro camera including the opto-electronic design, the raw format and the calibration.

1.5.1 Opto-electronic System

As illustrated in Fig.1.18 from the systems perspective the Lytro camera consists of (1) a motorized zoom lens, (2) a microlens array, (3) an image sensor, (4) a processing unit, and (5) a display.

The shape of the Lytro is a tube. More than half of the space in the tube is taken up by the zoom lens components. The focal length of the zoom lens ranges from $6.45\text{mm}$ to $51.4\text{mm}$ (zoom x1 to zoom x8) and the corresponding angle of view (AOV) is from $39^\circ$ to $5^\circ$. According to the zoom settings, the aperture stop is automatically adjusted to ensure a constant f-number of 2.

In contrast to the conventional camera, the microlens array inside the light-field camera is the unique component. As shown in Fig.1.19, the microlens array is fixed on a holder and the position is controlled by tightly attached springs. The microlenses are arranged in a hexagonal grid instead of a square grid to improve light efficiency.
The pitch of the microlenses is 13.89 µm. All the microlenses have the same focal length. Although the value of the microlens focal length is unknown, according to the $f$-number matching rule [17], the focal length of the microlens should be around 26µm. There are roughly 320 by 320 microlenses. Whether the material of the microlens array is polymer or glass is unknown. However, according to LYTRO Inc’s new released patent [79], shaping the lens surface using photo-lithographic techniques on polymer-on-glass has several advantages, such as no air gap, high placement precision. Unfortunately, this technique is not applied to the first generation Lytro camera.

Figure 1.19: The microlens array part of the Lytro camera. $d$ is the diameter of the microlens.

The image sensor is a colour sensor with Bayer filter. It is ON Semiconductor (formerly Aptina) MT9F002 which has 14 Mega pixels at 12-bit depth resolution. The size of the pixel is 1.4 µm. The output frame only uses a cropped region of the original
sensor output, which is 3,280 by 3,280 pixels. The effective pixel count is therefore 
10.7-mega-pixels. In the Lytro camera product sheet, they claim their product has 11 
mega-rays light field resolution.

1.5.2 Light Field Raw Format

The desktop software officially provided with the Lytro camera supports interactive 
refocusing and perspective shift. It also supports exporting the raw image as .LFP 
format file in its early version and .LFR format file in the latest version (4.3). Both .LFP 
and .LFR are proprietary formats. However, accessing Lytro raw files is possible. Patel 
[80] wrote a tool to extract light field raw data from .LFP file by reverse engineering. 
This allows us to access the light field raw data. As shown in Fig.1.20 there are 
four parts inside a .LFP file. Below the header file, there are two JavaScript Object 
Notation (JSON) tables describing the light field raw data. The first one describes the 
simple ID of the Lytro camera, such as the serial number. The second one describes all 
the properties of the Lytro camera such as zoom and focus settings. The information 
provided about the microlens array gives us a good reference for decoding the light 
field. The most important data for us is the light field raw data. It starts from the 
address of 0x0003C0. The total size of the raw data is the sensor resolution times the 
bit-depth per pixel, which is 3280 \times 3280 \times 1.5 = 16.1376 \text{ MB (megabytes)}. It ends at 
the address of 0xF64140. A detailed discussion of decoding light field raw data from 
.LFR can be found in [81]. A Lytro raw image is shown in Fig.1.21

1.5.3 Calibration

Calibration is a critical step in the reconstruction of 4D light field from the Lytro cam-
era 2D raw image. Determining the centre of each microlens image is not a trivial task 
as the microlens images suffer from noise, vignetting, and assembly imperfection of 
the microlens array.

For the purpose of fine calibration, there are a total of 62 reference raw images 
stored in the Lytro camera. These images will be automatically loaded into the PC 
by the desktop application software, once the camera is connected for the first time. 
These raw images are captured by Lytro with different zoom, focus, and exposure 
settings. These reference images are captured with a controlled uniform illumination 
source, probably with a diffuser in front of the Lytro camera. The total size of these 
calibration raw files is almost 1 GB (62 \times 16.14MB = 992MB ). The reason why such a
Figure 1.20: The data structure inside a .LFP. format file

A large amount of data needs to be stored in the Lytro camera is that the microlens image centres are shifted and vignetting profiles are changed with different zoom and focus settings of the main lens. Possibly, with the known optical design of the main lens and these reference images, a model can be built that would precisely predict the microlens image centres and vignetting profiles of the Lytro camera under arbitrary zoom and focus settings.

1.5.4 Reconstruction

Following proper calibration, the 4D light fields can be extracted from the Lytro 2D raw images. As illustrated in Fig. 1.22 re-arranging the 4D coordinates, the 4D light field can be stored in either microlens image array or multiple-view image array 2D format. The advantage of using multiple-view image array format is that the reconstruction quality of 4D light field can be visually examined easily, particularly for those views near the boundary. A real reconstructed multiple-view image array from the Lytro raw image is shown in 1.23. We discuss our proposed geometrical calibration and reconstruction algorithms in Chapter 3.
Figure 1.21: (a) Reconstructed central view of Lytro raw data. (b) Lytro raw data before demosacing and only available colour pixels are displayed. (c) Lytro raw data before demosacing, pixels are displayed with intensity value.
Figure 1.22: 4D Light field in 2D flatland mode. (a) Microlens image array. Each microlens image has $N$ by $N$ pixels. There are $M$ by $M$ microlenses. (b) Multiple-view image array. Each view has $M$ by $M$ pixels. There are $N$ by $N$ views in total. For example, the central view is composed by all the centre pixels (highlighted in orange) of the microlens images.
Figure 1.23: A 5 by 5 multiple-view image array reconstructed by our proposed light field reconstruction algorithm discussed in Chapter 3. The size of each view is 640 by 640 pixels.
Chapter 2

Light-Field Camera Simulation

Using computer-based raytracing to study optical aberrations has been developed over more than 40 years [73]. There are a variety of commercial software packages, such as Zemax, Code V and OSLO, which allow optical designers to minimize optical aberrations and maximize performance based on specific system constraints. In computer graphics, advances in rendering algorithms and computational power have accelerated the development of raytracing techniques significantly. Today, computer graphical software such as 3ds Max, Maya and Blender, is able to render very complex scenes with consumer-grade graphics hardware in an acceptable time.

For a computational imaging system, computational approaches will normally be involved in the optical design process. Current optical design software packages are oriented from the perspective of pure optical design. However, there are other factors that effect the performance of the whole imaging system, such as image sensor characteristics, processing algorithms. To allow computational imaging researchers to rapidly evaluate system performance from different perspectives, we have built a computational camera simulation framework. The hope of our work is to enable system developers to fully analyse and understand the system’s characteristics before prototyping. Already there are some alternative emerging commercial software packages designed for the camera system simulation, such as ISET [82] and Fivefo-
Our implementation offers several advantages,

- Our implementation supports not only conventional cameras but also computational cameras, in particular the microlens array (MLA)-based light-field camera.

- Our implementation provides flexible input and output interfaces. For example, it can load and save ray tracing results with Matlab. It can be easily extended to interface other open-source libraries. It also can be integrated into other open-source libraries as a stand-alone module.

- For the purposes of education, our implementation provides good examples of how to put the important properties of the optical system into practical use, particularity for computational imaging researchers in the fields of image processing and computer vision.

- Our implementation of the computational camera system simulation is provided as open-source software. It is free of charge.

Our framework combines camera models that are used for realistic camera rendering [84] and finite raytracing [73]. In terms of direction, raytracing can be classified into forward and backward raytracing. Forward raytracing as used in commercial packages such as Zemax is mainly for the purpose of evaluating optical performance. Light rays emanate from a light source and pass through the front lens element to the rear lens element. The simulation results are used to study the optical system aberrations, for example spot diagram and geometric point spread function. In contrast to forward raytracing, backward raytracing is widely used in computer graphics; the synthetic image is rendered by tracing rays starting from the detector plane, passing through the rear lens element to the front element, until it intersects with an object in the scene. In our work, we use both forward and backward raytracing approaches to study the computational imaging system.

Currently, wave optics effects such as polarization or diffraction, and internal reflection between lens surfaces have not been implemented. However, geometric optics is sufficient to study the optical performance for most macroscopic imaging systems with a large aperture. As a rule of thumb, a camera with \( f \)-number \( N \), operating at wavelength \( \lambda \) will have a diffraction limited spot radius of \( 1.22 \lambda N \). For example, at the wavelengths of visible light, a conventional camera with the \( f \)-number of 2, the diffraction spot radius is around 2.5 \( \mu m \) which is normally smaller than the size
of pixel and aberration spot size. The image sensor’s characteristics \cite{[85]} including spectral sensitivity and noise statistics can also be integrated into our framework.

![Diagram of simulation framework](image)

Figure 2.1: Sketch of our simulation framework which includes both forward and backward raytracing.

\section{2.1 Camera System Simulation}

Before explaining how to simulate image formation with a light-field camera, we first discuss how to simulate a conventional camera. We are interested in the conventional camera simulation because the MLA-based light-field camera is a modification of a realistic camera and thin or thick lens models limit the accuracy of the simulation result. First, we discuss how to trace rays through a single lens surface which is the most fundamental step for realistic lens raytracing. Next, we discuss how to trace rays through a lens system considering efficiency and radiometry.
2.1.1 Single Surface Raytracing

Coordinates Definition First, we define the coordinate system and the data structure of the ray. We take the \( z \) axis of a left-handed Cartesian coordinate system and the \( xy \) plane is orthogonal to the \( z \) axis. To simplify the ray tracing, we use a point \( \mathbf{r}_o = (x,y,z) \) to denote the origin of the ray and use a unit vector \( \mathbf{r}_d = (x',y',z') \) where \( x'^2 + y'^2 + z'^2 = 1 \), to represent the direction of the ray. A variable \( t \) is used to parametrize the distance that the ray travels in 3D free space,

\[
\mathbf{r}_o'(t) = \mathbf{r}_o + t \cdot \mathbf{r}_d. \tag{2.1}
\]

Now, we quantify how a ray is refracted at a given single surface. As illustrated in Fig.2.3 a ray \( \mathbf{r} \) at starting position \( C \) refracting at intersection point \( E \) with a single surface \( \Sigma \) of curvature \( \frac{1}{R} \) for which the left-hand side and right-hand side media have refraction indexes of \( n_1 \) and \( n_2 \), respectively.

First of all, we need to determine the intersection point \( E \). As shown in Fig.2.2 since the Point \( C \) and \( O \) are known to us, we can obtain vector \( \overrightarrow{CF} \) by projecting the vector \( \overrightarrow{OC} \) onto the ray \( r' \)'s direction vector \( \mathbf{r}_d \). The length of vector \( \overrightarrow{CE} \) is calculated using the theorem of Pythagoras,
Figure 2.3: A ray $r$ refracted to $r'$ by a single surface $\Sigma$.

\[ CF = CO \cdot r_d, \]
\[ |CO|^2 = |CF|^2 + |FO|^2, \]
\[ |R|^2 = |EF|^2 + |FO|^2. \]

Combining the above equations, the length of $|CE|$ is

\[ |CE| = |CF| \pm \sqrt{R^2 - |FO|^2}, \]
\[ = |CF| \pm \sqrt{R^2 + |CF|^2 - |CO|^2}. \]

The negative sign corresponds to a concave surface as defined previously. The origin of ray $r'$ is calculated as,

\[ r'_o = r_o + |CE|r_d. \] (2.2)

Next, the orientation of the ray $r'$ after refraction can be precisely calculated using Snell’s Law which is a simple consequence of Fermat’s principle [86]. The mathematical derivation of the refraction formula using similar vector representation appeared
in [87]. Here, we give a simpler derivation. As illustrated in Figure 2.4 in a unit circle, two rays \( r \) and \( r' \) can be decomposed onto an orthogonal basis \( N, T \). Since the angle between basis and rays are known, we can obtain \( r'_d \) by solving linear equations,

\[
\begin{align*}
\mathbf{r}_d &= -N\sin\theta_1 - N\cos\theta_1, \\
\mathbf{r}'_d &= -N\sin\theta_2 - N\cos\theta_2, \\
n_1\sin\theta_1 &= n_2\sin\theta_2.
\end{align*}
\] (2.3)

Eliminating the unknown unit vector \( T \), we have

\[
\frac{\mathbf{r}_d + N\cos\theta_1}{\mathbf{r}'_d + N\cos\theta_2} = \frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}.
\] (2.4)

Rewriting formula (2.4)

\[
\mathbf{r}'_d = (\mathbf{r}_d + N\cos\theta_1)\frac{n_1}{n_2} - N\cos\theta_2
\] (2.5)

where \( N, \theta_1 \) and \( \theta_2 \) are known variables. Mathematically, \( \cos\theta_1 = \mathbf{r}_d \cdot N \), \( \cos\theta_2 = \sqrt{1 - \left(\frac{n_1}{n_2}\right)^2(1 - \sin^2\theta_1)} \) and \( N = \frac{OE}{|OE|} \). When the angle between the ray and the optical axis is small, we have a first order approximation, also known as paraxial or Gaussian optics. Mathematically, we have
\[
\sin \theta \approx \theta, \\
\tan \theta \approx \theta, \\
\cos \theta \approx 1.
\]

As a consequence, Eq. 2.5 is approximated as,

\[
r'_d = (r_d + N) \frac{n_1}{n_2} - N = \frac{n_1}{n_2} r_d - (\frac{n_2 - n_1}{n_2}) N. 
\]  

(2.6)

If we use a scalar representation to substitute vector representation in Eq. 2.6, it becomes the paraxial ray tracing formula given in [73],

\[
\theta_2 = \frac{n_1}{n_2} \theta_1 - (\frac{n_2 - n_1}{n_2}) \frac{h}{R},
\]

(2.7)

where \( h \) is the height of intersection point. In our simulation framework, paraxial ray tracing simplifies the computation in finding the main properties of an optical imaging system which is discussed in the next section.

### 2.1.2 Lens System Raytracing

Prior to describing how we trace the ray through a lens system, we introduce the data structure of lens prescription used for our ray tracing.

**Lens System Description** In order to minimize different types of optical aberrations, a typical photographic lens system normally consists of from 4 to 14 lens elements depending on the design metrics such as performance and cost. A tabular description can be found in most lens patent documents. It is a very convenient and compact description of a lens system which can be easily interpreted by a computer program. A example description table of the Double-Gauss lens (US Patent 2,673,491) is shown in Tab. 2.5.

For instance, as shown in Fig. 2.5, the 5th lens surface has an aperture size of 18 mm, a radius of positive radius of curvature 12.7 mm, a separation of 5.7 mm to the next
Figure 2.5: A Double-Gauss lens, US patent 2,673,491. Each lens surface has five properties; the radius of curvature $R$, the thickness $d$, the refractive index $n_d$, the Abbe number $V_c$, and the aperture size $ap$. A negative radius of curvature indicates a concave surface when viewed from left to right. The physical stop can be distinguished either by the refractive index or the radius of curvature which are both set to zero. Units are in mm.

<table>
<thead>
<tr>
<th></th>
<th>$R$</th>
<th>$d$</th>
<th>$n_d$</th>
<th>$V_c$</th>
<th>$ap$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29.400</td>
<td>3.760</td>
<td>1.670</td>
<td>47.1</td>
<td>25.2</td>
</tr>
<tr>
<td>2</td>
<td>84.830</td>
<td>0.120</td>
<td>1.000</td>
<td>25.2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>19.275</td>
<td>4.025</td>
<td>1.670</td>
<td>47.1</td>
<td>23.0</td>
</tr>
<tr>
<td>4</td>
<td>40.770</td>
<td>3.275</td>
<td>1.699</td>
<td>30.1</td>
<td>23.0</td>
</tr>
<tr>
<td>5</td>
<td>12.750</td>
<td>5.705</td>
<td>1.000</td>
<td>18.0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.000</td>
<td>4.500</td>
<td>0.000</td>
<td>16.6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-14.495</td>
<td>1.180</td>
<td>1.603</td>
<td>38.0</td>
<td>17.0</td>
</tr>
<tr>
<td>8</td>
<td>40.770</td>
<td>6.065</td>
<td>1.658</td>
<td>57.3</td>
<td>20.0</td>
</tr>
<tr>
<td>9</td>
<td>-20.385</td>
<td>0.190</td>
<td>1.000</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>437.065</td>
<td>3.220</td>
<td>1.717</td>
<td>48.0</td>
<td>20.0</td>
</tr>
<tr>
<td>11</td>
<td>-39.730</td>
<td>0.000</td>
<td>1.000</td>
<td>20.0</td>
<td></td>
</tr>
</tbody>
</table>

The limitation of this form of description is that it is not able to represent an optical system which has a non-spherical element such as aspherical or free-form surface.

**Gaussian Property** There are several important properties of a Gaussian optical system which can be used to understand the main characteristics of the system by fast approximation, such as the focal length, exit pupil size and position, etc. As discussed in 2.1.1, we can use Eq. 2.7 to trace a ray under first order approximation to quickly determine the focal plane, principle plane, and pupil plane of the lens system. Since the lens system has rotational symmetry, paraxial ray ray tracing can be performed in
Figure 2.6: 2D Paraxial ray tracing diagram. \( u_{i-1}(u_i) \) is the incident ray’s angle before (after) passing the \( i \)th surface. \( h_i \) is the height of the intersection point at the \( i \)th surface. \( C_i \) is the curvature of the \( i \)th surface. \( n_{i-1}(u_i) \) is the refractive index at the left(right)-hand side of the \( i \)th surface.

With the first order approximation, the light ray’s transport including refraction and propagation are described by matrices. As shown in Eq. 2.7, single lens surface refraction is expressed in the following matrix notation,

\[
T_{ri} = \begin{pmatrix}
1 & 0 \\
\frac{1}{(n_i - n_{i+1})c_i} & \frac{n_i}{n_{i+1}}
\end{pmatrix}.
\] (2.8)

For rays propagating in free space, the propagation matrix \( T_{pi} \) is

\[
T_{pi} = \begin{pmatrix}
1 & d_i \\
0 & 1
\end{pmatrix}.
\] (2.9)

The main simplification of using paraxial ray tracing is that the horizontal intersection point is approximated as the intersection point between the lens surface and the optical axis. Combining Eq. 2.8 and Eq. 2.9, a ray travels with the initial condition \( x_0 = (\theta_0, h_0)^T \) through an optical system consisting of \( n \) elements with the final status \( x_n = (\theta_n, h_n)^T \) has the following mathematical representation,

\[
x_n = T_n x_0 = \prod_{i=1}^{N} (T_{ri} \cdot T_{pi}) x_0.
\] (2.10)

To determine the back focal point of the system, we trace a ray parallel to the optical axis i.e. with the initial condition \( x_0 = (0, h_0)^T \) and with the final condition \( x_n = (\theta_n, h_n)^T \). As shown in Fig. 2.6, the last segment’s distance \( d_n \) is calculated as,
Chapter 2. Light-Field Camera Simulation

\[ f' = \sum_{i=1}^{N} d_i = \sum_{i=1}^{N-1} d_i + \frac{h_n}{u_n}. \]

(2.11)

By definition [73], the back principle plane is the plane perpendicular to the optical axis at which the last ray segment meets the first ray segment. Therefore,

\[ p' = f' - \frac{h_0}{u_n}. \]

(2.12)

By analogy, the front focal point \( f \) and principle point \( p \) can be obtained in the same way but reverse order (from the back element to the front element). It is not difficult to prove that if the first and rear media are all air, which is always the case for photographic cameras, the front and back focal length are equal \( f = f_1 = f_2 \). As illustrated in Fig.2.7, knowing these important points of the lens system, we can approximately find the image point from the corresponding object point of a lens system through the lens maker’s formula [88] following the sign convention [73],

\[ \frac{1}{f} = \frac{1}{u} - \frac{1}{v}. \]

(2.13)

Figure 2.7: Thick lens approximation.

In practice, it is more convenient to use a 4 by 4 homogeneous perspective transformation matrix to find the correspondence between image and object space. In the simplest case where \( P \) and \( P' \) coincide (thin lens model), if the origin is set to \( P \) or \( P' \), the perspective transform matrix \( T_p \) for transforming the object point to image point [89] is
where \( f \) is the focal length of the system. For the thick lens model which is a generalization of the thin lens model, an extra matrix \( T_s \) is required to translate the origin between \( P \) and \( P' \) where the distance \( t = |PP'| \) is known as the thickness of the lens. The total transformation matrix \( T \) is

\[
T = T_s \cdot T_p = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & t \\
0 & 0 & 0 & 1 \\
\end{bmatrix} \cdot \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \frac{1}{f} & 1 \\
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & \frac{1}{f} t \\
0 & 0 & 1 + \frac{1}{f} t & \frac{1}{f} \\
\end{bmatrix}
\]

The perspective transformation matrix \( T' \) that transforms image points to object points is the inverse matrix of \( T \),

\[
T' = T^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & -t \\
0 & 0 & -\frac{1}{f} & 1 + \frac{1}{f} \\
\end{bmatrix}
\]

By definition [88], the entrance pupil is the image of the physical stop by the lens elements between the first one and the stop. Similarly, the exit pupil is the image of the physical stop by the lens elements between the stop and the rear one. Using paraxial ray tracing, the focal planes and principle planes are identified and then the exit pupil and entrance pupil are calculated with the perspective transformation matrix discussed above.

**Sampling Scheme**  
For forward raytracing, the origins of the rays are normally point sources in the object space. Due to the physical constraints, there is only a finite solid angle that allow rays to pass through the lens elements and arrive at the sensor plane. Improper sampling will lead to a large number of rays being blocked by the lens elements, in particular the lens stop, which reduces the simulation efficiency significantly. Taking advantage of the Gaussian properties of the lens system, an optimal
solution is to choose the entrance pupil as the sampling plane, through which the light rays can enter the lens system. By analogy, for the backward raytracing, the exit pupil is the optimal sampling plane [84].

The effect of using different sampling patterns at the entrance (exit) pupil is illustrated in Fig. 2.8. We are in favor of choosing the pattern that can produce a smooth spot diagram of which inner structure is close to the one produced by random sampling with high density. As shown in Fig. 2.8, concentric sampling produces less line artefacts than square grid sampling. This is because the shape of the entrance pupil (exit pupil) is circularly-symmetric for on-axis image points and near circularly-symmetric for off-axis image points.

**Radiometry**

For a real optical system, the image of a uniform illumination source usually shows a gradual fall-off in brightness, an effect called vignetting. One of the reasons is that the path of some light rays are blocked by different lens elements. To simulate this physical effect, we need to consider radiometry. For each pixel on the image sensor, its intensity value is proportional to the radiant flux arriving at its physical area over the exposure time. Concretely, the intensity value $I$ is the triple integral over the solid angle $\Omega$ of incoming rays of radiance $L$ over the planar area $A$ the pixel occupies and over exposure time $t$. Mathematically,

$$I = \int_{t} \int_{\Omega} \int_{A} L d\omega dt. \quad (2.14)$$

Considering a real image sensor, the spectral power distribution function $f(\lambda)$ and the spectral response function $d(\lambda)$ will be involved. Therefore, we have

$$I = \int_{t} \int_{\Omega} \int_{A} \int_{\lambda} f(\lambda) \cdot d(\lambda) \cdot L d\lambda d\omega dt. \quad (2.15)$$

As discussed in the previous section, from the perspective of the image sensor, the exit pupil is virtually from where the light rays emanate. For a realistic lens system, as illustrated in Fig. 2.9, suppose point $a$ is radiating power or radiant flux $\Phi$ which propagates uniformly to the semi-sphere. Therefore for the point $a'$ at distance $D_{aa'}$, the radiant flux $\Phi'$ becomes

---

1Here we ignore the angle sensitivity function of image pixels. 41
Figure 2.8: The effects of sampling patterns at the entrance pupil. The first row is for square grid sampling. The second row is for concentric grid sampling. The third row is for random sampling. We use around 1,000 samples for the regular sampling and 100,000 samples for the random sampling. Our spot diagrams are produced with a Double-Gauss lens.

\[ \Phi' = \frac{\Phi}{2\pi D_{aa'}^2}. \]  

(2.16)

The total amount of radiant flux \( E \) each pixel receives is calculated as
Chapter 2. Light-Field Camera Simulation

\[
E = \int_{\Omega} \int_{A} \frac{\Phi}{2\pi D_{aa}^2} d\omega' ds', \tag{2.17}
\]

\[
= \int_{\Omega} \int_{A} \frac{\Phi \cos \theta_1 \cos \theta_2}{2\pi D_{aa}^2} d\omega ds. \tag{2.18}
\]

where \(d\omega'\) and \(ds'\) are the effective area and \(d\omega' = \cos \theta_1 d\omega, ds' = \cos \theta_2 ds\) (See Fig. 2.9). If we assume the exit pupil plane is parallel to the sensor plane (\(\theta_1 = \theta_2\)) and \(D_{aa} \cos \theta = D\), Eq. 2.11 is simplified to the famous cosine-fourth law formula,

\[
E = \frac{1}{2\pi D^2} \int_{\Omega} \int_{A} \Phi \cos^4 \theta d\omega ds. \tag{2.19}
\]

Figure 2.9: Geometric relation between a pixel and the exit pupil for calculating the recorded intensity value of pixels.

Pixel vignetting as an important type of vignetting which has not been discussed. In some cases we will see later, the pixel vignetting has a significant influence e.g. in microlens based light field camera. Pixel vignetting occurs when the pixel angular sensitivity function is not uniform. Although sensors from different manufactures have different properties, it is reasonable to approximate the sensitivity function using Phong’s bidirectional reflectance distribution function (BRDF) model \[8\],

\[
\rho(u) = \cos^{\sigma}(u). \tag{2.20}
\]

where \(u\) is the angle between the incident ray and the sensor surface, \(\sigma\) represents the
reflective property of the sensor surface.

![Diagram showing sensitivity function with different reflective properties](image)

Figure 2.10: Approximated general sensitivity function with different reflective properties. Larger $\sigma$ indicates that the pixel is more angle sensitive.

See appendix C.1.1 to C.1.4 for the implementation of the C++ source code of the realistic lens simulation.

### 2.1.3 Realistic Rendering with Lens Models

To make the simulation result closer to real world experiments and to verify the algorithms with a known complex 3D scene, we choose Physically Based Ray Tracing program (PBRT) \[89\], an open source ray tracing program as the 3D rendering engine. In the main rendering loop, it consists of five modules: renderer, sampler, camera, integrator and film. The renderer module is in charge of organizing the other modules, passing parameters back and forth. The sampler module generates random samples in the pattern selected by the user. The camera module generates rays corresponding to the samples on the film (sensor). The integrator module computes the radiance along the ray arriving at the film. The film module produces the final image by calculating each pixel value contributed by associated random samples’ radiance on the film plane. With this structure it is only necessary to modify the camera module to a realistic lens or to add computational components such as coded aperture, microlens array.
2.2 Simulation Applications

In this section, we demonstrate three typical applications using our simulation framework. The first one is a classical optical design problem of finding the best focus point in a real lens system. In the second case, we use our simulation framework to visualize how the light fields propagate inside the real lens system and how they are related to image formation. In the last case, we evaluate the refocusing algorithm using backward raytracing and realistic image rendering.

2.2.1 Double-Gauss Lens Focusing

For a conventional photographic camera, there is no perfect focused point due to the presence of optical aberrations. Therefore, as shown in Fig.2.11(a), the circle of least confusion is treated as the best focus point. In this application, we demonstrate that our simulation can be used to find the focus point of a Double-Gauss lens by the means of minimizing the circle of confusion (CoC). The results are verified with Zemax, which is the standard optical design software used in industry.

![Double-Gauss Lens Focusing Diagram](image)

Figure 2.11: (a) A close-up of the circle of confusion for the on-axis object point imaged by the Double-Gauss lens. (b) The image distance is the optimal focus when the CoC has the minimum size.

In this example, we put the object point on the optical axis and 5000 mm from the first element of the Double-Gauss lens. The f-number is set to 4. Only the light at one wavelength (587.6 nm) is considered.
First, we calculate the first-order properties of the Double-Gauss lens as discussed in the previous section. Next, we use the image point calculated by paraxial approximation as the initial estimation. We shift the position of our initial estimation along the optical axis and use the RMS (Root Mean Square) radius as the metric (known as merit function in Zenax) to search for the circle of least confusion, as illustrated in Fig. 2.11(b). The results from our simulation and from Zemax OpticStudio 15.5 are compared in Tab. 2.12.

<table>
<thead>
<tr>
<th>Item</th>
<th>Zenax</th>
<th>Our</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entrance pupil (radius)</td>
<td>6.400</td>
<td>6.400</td>
</tr>
<tr>
<td>Entrance pupil (position)</td>
<td>19.947</td>
<td>-19.947</td>
</tr>
<tr>
<td>Exit pupil (radius)</td>
<td>6.814</td>
<td>6.814</td>
</tr>
<tr>
<td>Exit pupil (position)</td>
<td>-14.308</td>
<td>-14.309</td>
</tr>
<tr>
<td>Effective focal length</td>
<td>51.200</td>
<td>51.202</td>
</tr>
<tr>
<td>Airy disk (radius)</td>
<td>0.00289</td>
<td>0.00287</td>
</tr>
<tr>
<td>The circle of least confusion (position)</td>
<td>-69.343</td>
<td>-69.344</td>
</tr>
<tr>
<td>The circle of least confusion (radius)</td>
<td>0.00049</td>
<td>0.00029</td>
</tr>
</tbody>
</table>

Figure 2.12: Results Comparison Table. The units are in mm.

Our results are almost identical to Zemax with the exception of the radius of the circle of least confusion, maybe because the numerical error has a larger impact when the value is small. In this example, the spot radius will in fact be dominated by diffraction (radius of diffraction spot is 2.9 µm) rather than geometric optics.

### 2.2.2 Light-Field Camera Image Formation

In MLA-based light-field cameras, the angle information of the rays is preserved at the expense of loosing the spatial resolution. For the detailed discussion, see Chapter 1, light field acquisition devices section. It is of interest to understand the different working principles between the MLA-based light-field camera and the conventional camera from the perspective of light field transport and projection.

Our simulation setup is illustrated in Fig. 2.13: three local light fields are sampled at plane 1, 2 and 3. The point source is 5000 mm from the first element of the primary lens, which is a Double-Gauss with a f-number of 4. The microlens is treated as a thin lens. The microlens has a matched f-number to the primary lens. The sensor is placed behind the microlens array, with a fixed separation of one focal length. We observe the sampled local light field in three positions: (b) just before the microlens refraction, (c) after refraction by the microlens array, and (d) at the sensor plane. The
specification of the optical system and the optical setup are summarized in Tab. 2.2.

![Diagram of light-field camera simulation setup](image)

Figure 2.13: Light-field camera simulation setup. The microlens arrays is inserted at three different planes 1, 2 and 3.

<table>
<thead>
<tr>
<th>Object</th>
<th>Main Lens</th>
<th>MLA</th>
<th>Sampled 2D Light Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-axis</td>
<td>Double-Gauss Lens</td>
<td>Thin lens model</td>
<td>1: ( z = -67.343 )</td>
</tr>
<tr>
<td>( z = 5000 )</td>
<td>( F = 50.3 )</td>
<td>( F = 0.8 )</td>
<td>2: ( z = -69.343 )</td>
</tr>
<tr>
<td>( f )-number= ( \frac{f}{4} )</td>
<td>( f )-number= ( \frac{f}{4} )</td>
<td>3: ( z = -71.343 )</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Optical configurations. Units are in \( mm \).

As shown in Fig. 2.14(a1-a3), the point source is represented by a vertical line in the 2D light field representation. After passing through the Double-Gauss lens, the 2D light fields are sheared both horizontally and vertically due to refraction and propagation. Without the presence of the microlens array, three local light fields are sampled at 1, 2, 3 planes. Since the image can be seen as the vertical projection of the 2D light field, as shown in Fig. 2.14(b2), the radius of the geometric spot is less than 1\( \mu m \) at 2 plane. As the geometric spot is small enough, 2 plane can be treated as the focal plane of the primary lens. In an aberration-free lens system, for an image point, the ideal shape of the 2D light field is also a vertical line at the focal plane. However, for the Double-Gauss lens, the spherical aberration distorts the vertical line to a curved line. As shown in Fig. 2.14(b1),(b3), the shapes of the 2D light field at plane 1, 3 are straight lines with different tilt angles. They have similar size to the geometric spot, both are due to large defocus.

Fig. 2.14(c1-c3) show the shape of the sampled light fields which are just refracted by the microlenses at plane 1,2 and 3. In contrast to the conventional camera, the local light fields are not projected onto the sensor plane but refracted by the microlenses. As shown in Fig. 2.14(c1,c3), the local light fields are vertically sheared in a periodic manner, whereas the local light field at plane 2 remains almost the same as most of the local light fields pass through the optical centre of the microlens. After passing
through the microlens array, light rays continue to propagate to the sensor plane, the corresponding shapes of the 2D local light field are shown in Fig.2.14(d1-d3). For the defocused planes 1 and 3, the final light field 2D images have a similar spot radius but also contain patterns among the microlens images in which the angular information is embedded. For the focal plane 2, the spot radius is increased by a factor of 10 compared to the case in which the microlens array is absent. However, the angular information is well preserved inside the microlens image. This is indeed the trade-off between spatial resolution and angular resolution for the MLA-based light-field camera.

See appendix C.1.5 for the implementation of the C++ source code of the microlens array raytracing.

### 2.2.3 Refocusing Performance Evaluation

In comparison with the conventional camera, we use our simulation framework to evaluate the refocusing performance of an MLA-based light-field camera. For a conventional camera, a smaller $f$-number would lead to higher light efficiency but also a shallower depth of field (DoF). Due to the digital refocusing capability of the MLA-based light-field camera, with the same optical settings of the main lens the MLA-based light-field camera has a larger DoF.

In this example, we use the Double-Gauss lens (with a large aperture size) as the main lens for both the conventional camera and the light-field camera. The ISO 12233 test chart is the object in the scene. Initially, we place the chart 2000 $mm$ away and adjust the MLA plane (sensor plane) to the focal plane of the light-field (conventional) camera. Next, we move the chart to 6 different positions and render the image. The simulation configurations are shown in Tab.2.3.

<table>
<thead>
<tr>
<th>Object</th>
<th>Main Lens</th>
<th>MLA</th>
<th>Sensor</th>
<th>Object Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 12233 Chart</td>
<td>Double-Gauss Lens $F=50.3$</td>
<td>Thin lens model $f=0.025$</td>
<td>4983 by 4983</td>
<td>1400,1600</td>
</tr>
<tr>
<td></td>
<td>$f$-number=$f/2$</td>
<td>$f$-number=$f/2$</td>
<td>Pixel size 0.014</td>
<td>1800,2000</td>
</tr>
<tr>
<td></td>
<td>Diameter 0.0154</td>
<td>Diameter 0.0154</td>
<td></td>
<td>2200,2400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2600</td>
</tr>
</tbody>
</table>

Table 2.3: Simulation configurations. Units are in $mm$.

The results of the conventional camera is shown in Fig.2.16. Except for the focused image, the images look blurry due to the defocus. In contrast, the optimal digitally
Figure 2.14: Light field transport and image formation. The first column shows the 2D local light fields at the object plane. The second to fourth columns show the 2D local light fields before the microlens refraction, after microlens refraction, and before the projection onto the sensor plane, respectively. Three rows from top to bottom correspond to plane 1, plane 2, and plane 3, respectively.
refocused images look sharper. In the following, after showing the details of the refocusing algorithm [26], we analyse the sharpness of the results using Spatial Frequency Response (SFR) [90].

Recall that the refocusing operation is just a re-projection operation of the 2D light field after propagating it to a virtual sensor plane. With the 4D light field representation (Chapter 1), the new spatial location \((x', y')\) of each ray \(l(x, y, u, v)\) is calculated as,

\[
x' = x + (1 - \alpha) \cdot u,
\]

\[
y' = y + (1 - \alpha) \cdot v,
\]

where \(\alpha\) is the refocusing parameter that governs how far the synthetic plane is away from the original plane. The pixel intensity value \(I_\alpha(x', y')\) of digitally refocused image is obtained by,

\[
I_\alpha(x', y') = \sum_u \sum_v l(x', y', u, v).
\]

We determine the optimal refocusing parameter \(\alpha\) using the Laplacian operator [91] as the sharpness metric. The optimal refocusing parameters and equivalent distance of the focal planes are shown in Table 2.15.

<table>
<thead>
<tr>
<th>Distance of the Object</th>
<th>Distance of the Focal Plane</th>
<th>Refocusing Parameter (\alpha)</th>
<th>Distance of the Virtual Focal Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1400</td>
<td>-71.735</td>
<td>1.013</td>
</tr>
<tr>
<td>2</td>
<td>1600</td>
<td>-70.491</td>
<td>1.008</td>
</tr>
<tr>
<td>3</td>
<td>1800</td>
<td>-70.303</td>
<td>1.003</td>
</tr>
<tr>
<td>4</td>
<td>2000</td>
<td>-70.150</td>
<td>1.000</td>
</tr>
<tr>
<td>5</td>
<td>2200</td>
<td>-70.030</td>
<td>0.997</td>
</tr>
<tr>
<td>6</td>
<td>2400</td>
<td>-69.930</td>
<td>0.995</td>
</tr>
<tr>
<td>7</td>
<td>2600</td>
<td>-69.843</td>
<td>0.992</td>
</tr>
</tbody>
</table>

Figure 2.15: Refocusing parameters. Units are in mm.

In order to assess the quality of the image, we use slanted-edge analysis [90] to calculate the spatial-frequency response (SFR), of which the open source implementation can be found from [92]. The spatial Nyquist frequency for conventional cameras and MLA-based light-field cameras are determined by the pixel size and the microlens size, respectively. For this case, the Nyquist frequencies are 350 line pairs/mm (conventional camera) and 32 line pairs/mm (MLA-based light field camera), above which
Chapter 2. Light-Field Camera Simulation

Figure 2.16: The first row and second row are the conventional camera image sequence and the optimal refocused image sequence from the MLA-based light-field camera, respectively. The objects are placed from 1600 mm to 2400 mm away from the main lens. The red boxes indicate where we measure the SFR.

The signal will cause aliasing in the image and reduce the quality. We choose SFR20 (spatial frequency where contrast is 20%) as the threshold to determine the maximum resolvable spatial frequency.

<table>
<thead>
<tr>
<th>Object Distance (mm)</th>
<th>1400</th>
<th>1600</th>
<th>1800</th>
<th>2000</th>
<th>2200</th>
<th>2400</th>
<th>2600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional Camera (lp/mm)</td>
<td>4.2</td>
<td>6.4</td>
<td>12.9</td>
<td>223</td>
<td>16.5</td>
<td>9.1</td>
<td>7.1</td>
</tr>
<tr>
<td>Light-Field Camera (lp/mm)</td>
<td>22.0</td>
<td>25.8</td>
<td>25.8</td>
<td>34.5</td>
<td>25.8</td>
<td>23.9</td>
<td>18.7</td>
</tr>
<tr>
<td>Improvement</td>
<td>5.2</td>
<td>4.0</td>
<td>2.0</td>
<td>0.15</td>
<td>1.6</td>
<td>2.6</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Figure 2.17: SFR table (Spatial frequency at 20% modulation).

In the case of focus, the resolvable frequency of the conventional camera is 223 lp/mm. Because of the spatio-angular trade-off, the resolvable frequency for the MLA-based light-field camera is reduced to 34.5 lp/mm. In the case of defocus, the resolvable frequencies of the conventional camera drop significantly, which are all less than 17 lp/mm. With the optimal digital refocusing, in general the resolvable frequencies for the MLA-based light-field camera are improved to higher than 18 lp/mm. Using our simulation framework, we demonstrate that digital refocusing extends the DoF but also at the expense of losing spatial resolution when the image is focused.

See appendix C.1.6 for the implementation of the C++ source code of the digital refocusing.
Figure 2.18: Top figure shows the SFR results for the conventional camera. Bottom figure shows the SFR results for the MLA-based light-field camera. Different object distances are labeled with different colours, for which the value can be found in Tab.2.15.
Chapter 3

Light-Field Camera Calibration

In this chapter, our work is focused on the light field reconstruction and radiometric calibration for the MLA-based light-field camera. The critical step of 4D light field reconstruction is to accurately determine the centre of the microlens images. In the first section, we introduce our approaches on how to use the global grid to extract the microlens image centres with the reference image and without the reference image. The quality of the reconstructed multiple-view image array is quantitatively accessed experimentally and compared with state-of-the-art algorithms. The importance of radiometric calibration for the MLA-based light-field camera is explained at the beginning of the second section. To model the vignetting of the MLA-based light-field camera, we propose a joint global polynomial fitting and local Gaussian fitting. We also demonstrate vignetting correction based on our modeling prediction.

Contribution  We introduce a novel technique to reconstruct the 4D light field from the MLA-based light-field camera without the need for the reference image using a simple and efficient blind global grid fitting approach. Also, according to our knowledge, our work is the first attempt to investigate the vignetting modeling for the MLA-based light field camera. Although our vignetting modeling is not able to remove the vignetting as well as the direct approach, it demonstrates the possibility of successful
vignetting removal with the modeling.

3.1 Light Field Reconstruction

The geometric calibration of traditional cameras is a well-explored topic in the field of computer vision \[93\]. Due to the presence of the microlens array, the geometric calibration for an MLA-based camera inevitably becomes not only more complicated but also requires extra steps. As discussed in the previous section, in decoding the 4D light field from a 2D raw image, the centre of each microlens image is regarded as the origin of the angular samples inside the microlens image. However, the microlens array is not perfectly aligned with the sensor due to manufacturing and assembly imperfections. Accurately detecting the centre of each microlens image is the fundamental requirement for restoring high-quality light fields. We first discuss the relation between the microlens image and optical settings. Next, we explore how to determine the microlens image centre by a robust local centroid estimator with line and global grid fittings. We also explore the possibility of directly extracting the 4D light field from the 2D light field raw image without the need for a reference image. This is achieved by a blind global grid fitting and a refinement scheme.

3.1.1 Related Work

The recent growing interest in light field imaging has resulted in several papers addressing calibration and reconstruction of the light field from an MLA-based light-field camera. Dansereau et al. \[94\] proposed a decoding, calibration and rectification pipeline, in which a 15-parameter model was introduced to correct geometric distortion. Cho et al. \[95\] introduced a learning based interpolation algorithm to refine the barycentric interpolation result to restore high-quality light field images. Yunsu et al. \[96\] proposed a line feature-based geometric calibration method for an MLA-based light-field camera. All these approaches require a uniformly illuminated image as a calibration reference image. One exception is Fiss’s \[97\] recent work, which proposed to use dark vignetting points as a metric to find the spatial translation of the MLA with respect to the centre of the image sensor.

Actually, detecting the centre of microlens images has been well studied in wavefront sensing. A comprehensive survey can be found in \[98\]. The difference is that in wavefront sensing the measured centre is usually a small spot and noise is the main
factor that degrades the estimation accuracy.

3.1.2 Principles

To reconstruct 4D light fields from 2D light field raw images, the key step is to accurately determine the centre of the microlens image rather than the geometric centre of the microlens. This is because the central part of the exit pupil is imaged to the central part of the microlens image. As illustrated from Fig. 3.1, the embedded \((u, v)\) coordinates are extracted from where the microlens image centres are regarded as the origins.

![Microlens Centre and Microlens Image Centre](image)

Figure 3.1: (a) Pinhole lens model. (b) The additional coordinates \((u, v)\) is embedded in the microlens images. There is a spatially varying offset between the centre of microlens image and the centre of microlens, depending on the optics settings.

In fact, the microlens image centre changes for different zoom and focus settings. As illustrated in Fig. 3.1, the relation between the physical microlens centre and microlens image centre is

\[
h' = (1 + \frac{d_{ml}}{d})h.
\]  

(3.1)

where \(d\) is the separation between the exit pupil plane and the microlens array plane, \(d_{ml}\) is the separation between microlens array and image sensor, \(h', h\) are the heights of microlens centre and microlens image centre with respect to the optical axis. Ideally, given the physical location of each microlens centre, optical settings \(d\) and \(d_{ml}\), we can determine the corresponding microlens image centres intermediately by Eq. 3.1. As shown in Fig. 3.2, a shorter focal length of the main lens leads to a larger centre offset. The centre offset is also spatially variant; it increases as the microlens centre is further away from the optical axis.
Chapter 3. Light-Field Camera Calibration

Figure 3.2: The microlens image centre offsets with respect to the focal length of the main lens. Separation between sensor and microlens array $d_{ml} = 0.0025\, mm$, Pixel size $p = 0.0014\, mm$. Height refers to the distance of the microlens centre from the optical axis.

In practice, in order to determine the microlens centres, we need to remove the main lens and shine a parallel beam perpendicular to the microlens array (Fig. 3.3(a)). Alternatively, we can place a point source at the front focal point of the main lens (Fig. 3.3(b)). These approaches require high precision control of the light source, position and orientation. For example, if the parallel beam is not perpendicular to the microlens array, there will be a constant offset between the microlens image centre and the microlens centre.

Figure 3.3: Determine the centres of microlenses. (a) Use a parallel beam. (b) Place a point source at the front focal point of the main lens.

However, it is possible to extract the centre of the microlens image without using the above mentioned illumination control. Specifically, we can use a uniform illumi-
nation source and place a diffuser in front the MLA-based light-field camera. There will be a uniform spot image formed under each microlens, and this light field raw image is referred to as the reference image or the modulation image. We can reduce the diameter of the aperture stop to limit the size of the uniform spot to several bright pixels. The centre can then be determined using a centroid algorithm. Nevertheless, this is not a solution for the Lytro camera, as the aperture stop is controlled by the electronic circuit and is not accessible by common users. Placing a pinhole mask in front of the Lytro could also reduce the uniform spot image to a few pixels. But it is not equivalent to stopping down the aperture, as it also shifts the exit pupil plane along the optical axis. In the next section, we will discuss how to accurately extract the microlens image centres from the uniformly illuminated reference image.

### 3.1.3 Centre Estimation with Reference Images

In this section, we introduce our approach to determine the centre of the microlens images. Due to the accuracy limitation of the local centroid estimator, we employ a line fitting refinement scheme to improve the estimation accuracy to the sub-pixel level. We also show that it is possible to use a global grid to localize the centres of the microlens images without considering the individual spatial variation.

**Reference Images** In Fig. 3.4 we show two uniformly illuminated reference images with different focal lengths of the main lens. These reference images are stored in the Lytro camera for the purpose of calibration. The shape of the microlens image disk varies across the raw image and also varies with different optical settings (see Fig. 3.4). In addition, the shape of the disks located near the corner of the raw image are highly distorted. The colour filter array (CFA) also reduces the uniformity of the intensity due to different spectral responses.

**Local Centroiding Estimator** Dansereau et al. [94] proposed to convolve the reference image with a 2D Gaussian kernel and use the position of the pixel with the maximum intensity as the centre. Cho et al. [95] used a parabolic surface fitting to the 2D intensity profile of the microlens image to determine the microlens image centre. Similar to these approaches, we first apply an erosion operation to the reference image and then convolve with a box filter. Next, we treat the pixel which has the highest intensity value of each spot as the centre. Sub-pixel centre estimation \((x'_c, y'_c)\) is obtained by applying intensity weighted centroiding [98] at the coarse estimation.
Chapter 3. Light-Field Camera Calibration

Figure 3.4: First row is the Lytro raw image with a focal length 51.3 mm of the main lens. Second row is the Lytro raw image with a focal length 7.8 mm of the main lens.

where \( I_{ij} \) is the pixel value in the processed reference image. Compared to Cho et al.'s [95] approach, our approach does not require large matrix multiplications. Therefore, our approach is faster by an order of magnitude with our computational platform.

In the following, two reference images (see appendix A.1 for more detailed description of the raw images) with focal lengths of 51.3 mm and 7.8 mm zoom lens are tested with line fitting and global fitting.

Line Fitting We assume the position error tolerance for the microlens is less than 0.25 \( \mu m \) (equivalent to 0.18 pixels for the Lytro camera), which is a reasonable assumption according to the specification of SUSS MicroOptics microlens array [99]. We first look at the line fitting errors to investigate the effects of the individual microlens spatial variation. In this procedure, we fit straight lines to the horizontal and vertical centroid position estimates. A single line fitting example is shown in Fig. 3.5

The single line fitting example shows that there are about five pixels difference in the vertical direction from the left side to the right side of the microlens image. The
line fitting results for both centroid estimators are almost identical. For Cho’s and the intensity weighted centroid estimators, the mean residual error are 0.099 and 0.054 pixels, respectively.

Fig. 3.6 shows the line fitting results using the intensity weighted centroid estimator for the whole reference image. For both cases, the mean and standard deviation of the residuals are less than 0.12 pixels. This shows that random variations in the lenslet positions are negligible. In fact, line fitting refinement not only rejects the outliers estimated by the centroid estimator, but also improves the estimation accuracy to the sub-pixel level. In the next section, we investigate that whether a global grid can be used to localize the micro lens image centres.

**Global Grid** We use the projective (homography) transformation to describe the relation between the ideal micro lens image centre \((x_c, y_c)\) and the real micro lens image centre \((x'_c, y'_c)\). Mathematically,
Figure 3.6: Line fitting results for the *Lytro* reference images captured with a focal length of 51.3 mm and 7.8 mm, respectively. The shaded regions indicate the standard deviations.

The transform matrix $H_{3 \times 3}$ is defined by

$$
H_{3 \times 3} = \begin{pmatrix}
    a_{00} & a_{01} & a_{02} \\
    a_{10} & a_{11} & a_{12} \\
    a_{20} & a_{21} & 1
\end{pmatrix}.
$$

The projective transformation matrix $H_{3 \times 3}$ can be obtained by solving the optimization problem minimizing the total error residual between the measured point set and transformed ideal point set as illustrated in Fig. 3.7. The residual error $e$ is calculated as

$$
e = \sqrt{\left(\frac{a_{00}x_c + a_{01}y_c + a_{02}}{a_{20}x_c + a_{21}y_c + 1} - x'_c\right)^2 + \left(\frac{a_{10}x_c + a_{11}y_c + a_{12}}{a_{20}x_c + a_{21}y_c + 1} - y'_c\right)^2}.
$$

As shown in Fig. 3.8, the mean and standard deviation of the residual error by global grid fitting are $\mu = 0.10, \sigma = 0.04$ and $\mu = 0.09, \sigma = 0.04$ (units are in pixels). Therefore, the planar global grid is accurate enough to localize the microlens image centres.
Chapter 3. Light-Field Camera Calibration

3.1.4 Centre Estimation without Reference Images

In the previous section, we discussed how to build a global grid from the reference image. However, reference images are not always available. Another advantage of establishing the global grid without the reference image is that the memory storage requirement is significantly reduced (See Chapter 1 Lytro camera).

Our goal is to infer the transformation matrix $H_{3 \times 3}$ without the reference image. Ideally, four pairs of points are enough to find the homography matrix which contains 8 unknowns. In practice, we would like to include more pairs to improve the estimation accuracy. In Fig.3.9(a), we show how we find the correspondences with

Figure 3.7: The projective transformation matrix is estimated by minimizing the total residual error of the correspondences from the ideal grid and practical grid.

Figure 3.8: The residual errors (in pixels) using the planar global grid for the reference images captured with a zoom lens with focal lengths of 51.3 mm and 7.8 mm, respectively.
a step-by-step neighbour search for the reference image. However, for the content image as illustrated in Fig. 3.9(b), it is not feasible when there are low-intensity pixels. Therefore, we proposed a blind global fitting approach. First, the initial guess of $H$ is determined by a frequency domain analysis and is then refined by our proposed optimization framework; blind global grid fitting.

**Frequency domain analysis** We perform the 2D Fourier transform on both the reference raw image and the content raw image (Office). The results are shown in Fig. 3.10. The regular geometric structure of the hexagonal microlens array results in six high peaks distributed uniformly around the zero frequency. Although the spectrum of the content image is contaminated by the content changes inside each microlens image, the six high peak frequency components are preserved and can be used to determine the size of the hexagonal grid.

In the spatial domain, as shown in Fig. 3.11, the repetitive pattern is illustrated in one direction. According to Fourier transform theory, the radius $R$ (unit is in pixels) of the hexagon is calculated as

$$R = \frac{2d'}{\sqrt{3}} = \frac{4\sqrt{3}L}{\sum_{i=0}^{5}|f_i|}. \tag{3.5}$$

where $L$ is the total number of cycles across the raw image which is equal to the width of the raw image. $f_i$ is $i$th peak frequency component, unit is in cycles/pixel.
Local Centroid Estimator  We propose a unique local centroid estimator which enables us to measure the centre of microlens images using Lytro content raw image. As discussed in 4.1.3, these local centroid estimators \[94,95\] are not able to determine the centre with the non-uniform spot. In contrast, we measure the individual microlens image centres by examining the dark pixels among the microlens images. Concretely, for either a square or hexagonal MLA, there are dark gaps between the microlens images. For example, as shown in Fig.\[3.12\] the position of the darkest spots of a hexagonal grid with respect to the centre of a microlens are \[p_0 = (R, \frac{R}{\sqrt{3}}), \quad p_1 = (0, \frac{2R}{\sqrt{3}}), \quad p_2 = (-R, \frac{R}{\sqrt{3}}), \quad p_3 = (-R, -\frac{R}{\sqrt{3}}), \quad p_4 = (0, -\frac{2R}{\sqrt{3}}), \quad p_5 = (R, -\frac{R}{\sqrt{3}})\], where \(R\) is the radius of the hexagonal grid. For an arbitrary pixel at position \(x = (x, y)^T\) of the light field raw image \(I\), the summation of the six surrounding pixels denoted by \(P(x)\) is used to
detect the centre of the microlens image. To achieve sub-pixel accuracy, we up-sample
the image by a factor of 4 with cubic interpolation. Additionally, to reduce the noise,
a Gaussian filter is applied before the up-sampling. The local centroiding estimator is
defined as finding the minimum solution for a score map $P(x)$,

$$P(x) = \sum_{i=0}^{5}(G_{\sigma} * I_{4})(x + 4p_i). \quad (3.6)$$

If the light field raw image is uniformly illuminated, $P(x)$ reaches a local minimum
only when $x$ is at the centre of a microlens image. Even for the natural light field raw
image, $P(x)$ still reaches a local minimum when $x$ is at the centre of a microlens im-
age. However, if there are some low-intensity pixels, then multiple minimum points
may exist, one of which is the true centre of the microlens image, as illustrated in
Fig.3.13.

Evidently, our local estimator is not able to individually find every microlens im-
age centre from a natural light field raw image. Instead of using the local minimum
to identify the individual microlens image centre, we propose a global optimization
scheme to estimate the global grid transformation matrix $H$.

**Global Grid Parameter Optimization** Our approach first generates an up-sampled
centroiding score map $P$ based on our local estimators. Then we can use the summa-
Chapter 3. Light-Field Camera Calibration

Figure 3.13: Top row, microlens images (with a magnification factor of 8) with different shapes. Bottom row, score maps produced by our local estimator. The centre of microlens image is located in the dark blue regions.

...tion of the scores of the grid as a metric to measure how well the grid is fitted to the centres of the microlens array. As illustrated in Fig. 3.14, only a best fit grid model will produce the global minimum.

...Figure 3.14: Global grid fitting. The fitting score reaches the minimum only when the grid parameters are all optimum as highlighted in red.

Thus we formulate it as a global optimization problem. The cost function $F$ is defined as

65
Chapter 3. Light-Field Camera Calibration

\[ \mathcal{F}(H) = \sum_{j=1}^{M} \sum_{i=1}^{N} P(\mathcal{T}(H, x_{ij})). \]  
(3.7)

where \( \{x_{ij} | i = 1, \ldots, N, j = 1, \ldots, M \} \) is the hexagonal grid with a unit radius,

\[ x_{ij} = \begin{cases} 
(i,j,1) & \text{for } j \text{ is odd}, \\
(i,j + \frac{1}{2},1) & \text{for } j \text{ is even}. 
\end{cases} \]  
(3.8)

\( \mathcal{T} \) is the homography transformation operation and \( H \) is the 3 by 3 homography matrix. The cost function \( \mathcal{F} \) reaches a global minimum only when the grid parameters are accurately estimated.

Since this problem has no analytical expression, we evaluated several numerical optimization methods. For example, the Nelder-Mead algorithm \([100]\) has fast convergence rates but occasionally gets stuck at a local minimum. The simulated annealing algorithm \([101]\) as a probabilistic method guarantees the solution is a global minimum but the rate of convergence is very slow. Also tuning the parameters such as the cooling factor can be difficult to optimize.

Instead, we choose a two-step brute-force searching scheme. However, there are eight unknowns in homography matrix \( H \). It would consume significant time optimise these parameters. To reduce the dimensionality, we only estimate four parameters. They are horizontal translation, vertical translation, rotation and scaling. The homography matrix is estimated in the refinement scheme. Thus the optimization problem is simplified to

\[ \mathcal{F}(S, K, T) = \sum_{j=1}^{M} \sum_{i=1}^{N} P((S \cdot K \cdot T \cdot A \cdot x_{ij})). \]  
(3.9)

where matrix \( A \) contains the information of the real size of the hexagonal grid and it is determined in the frequency domain analysis section,

\[ A = \begin{pmatrix} 
2R & 0 & 0 \\
0 & \sqrt{3}R & 0 \\
0 & 0 & 1 
\end{pmatrix}, \]  
(3.10)

Matrix \( T \) is the translation matrix,
Chapter 3. Light-Field Camera Calibration

\[ T = \begin{pmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{pmatrix}, \]  
\[ (3.11) \]

Matrix \( K \) is the rotation matrix governed by the rotation angle \( \theta \),

\[ K = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \]  
\[ (3.12) \]

Matrix \( S \) is the scaling matrix,

\[ S = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix}. \]  
\[ (3.13) \]

Figure 3.15: A example of the two-step 2D brute-force search. Left, translation parameter search. Right, scaling and rotation parameter search.

To speed up the search, we set reasonable boundary constraints for each parameter. The spatial translations \( T_x \) and \( T_y \) are in the range of \([-R, R]\). We also assume the rotation angle is within \( \pm 0.1 \) degree. To search the parameters in 4 dimensions is still time consuming. We divide it into two steps: we first search \( T_x \) and \( T_y \) then \( s, \alpha \). A real world example is shown in Fig. 3.15.

At this stage, we have a coarse estimation of the global grid with 4 parameters. However, to get the homography matrix \( H \) just as we did with the reference image, we propose a valid region selection scheme. There are some microlens images with
low image intensity in the content raw image and our proposed dark gap centroid estimator is not able to find these centres accurately. We use the ratio between the average score of the surrounding neighbor pixels and the local pixel score to identify if the estimation is reliable or not. For the pixel at $x_0$, the confidence $r(x_0)$ is defined as

$$r(x_0) = \frac{\sum_i P(x_i)}{N \cdot P(x_0)}.$$  

(3.14)

Our algorithm is summarized in Alg.1.

**Input**: Centroiding score map $P$.

**Output**: Homography Matrix $H$.

**Processing**:

Step 0. Parameter initialization $A, T, S, R$.

Step 1. Brute-force Search to find optimum $Tx$ and $Ty$.

for $j \leftarrow -R$ to $R$ do

for $i \leftarrow -R$ to $R$ do

$Tx_i = Tx_0 + \delta \cdot i$

$Ty_i = Ty_0 + \delta \cdot j$

Update $F$

if $F < F_{min}$ then $F_{min} = F, Tx = Tx_i, Ty = Ty_j$;

end

end

Step 2. Brute-force Search to find optimum $\alpha$ and $\theta$. Similar to Step 1.

Step 3. Build the global grid.

Step 4. Remove the unreliable grids (Optional).

for $j \leftarrow 0$ to $M$ do

for $i \leftarrow 0$ to $N$ do

if $r(x_{ij}) < 1.25$ then Remove $x_{ij}$;

end

end

Step 5. Homography Estimation.

**Algorithm 1**: Centre Estimation without Reference Image.

We applied our algorithms to six natural scenes as shown in Fig.3.16. The optical setting for these scenes can be found in Appendix A. The residual errors between the global grids built by the reference image and without reference image are shown in Tab.3.1 and Fig.3.17.

In general, our algorithm successfully determines the centre of the microlens image without the need for a reference image and the RMSE is kept within 0.25 pixels, even
Use all the region | Use only valid region
---|---
ISO chart1 | 68.9% | 0.048 | 0.050
Parrot | 66.9% | 0.153 | 0.141
Office | 42.5% | 0.124 | 0.349
Toy map | 64.5% | 0.185 | 0.226
Flower4 | 49.5% | 0.268 | 0.278
Cat | 23.5% | 0.223 | 0.353

Table 3.1: Error table for building global grid without the reference image. The root-mean-square error (RMSE) are in pixel unit.

for the case that there are large regions where image intensity is low (Flower4 and Cat). As expected, the most significant errors are found at the boundaries of the raw image. This is because the uncertainty is larger at boundaries for our dark-gap based centroid estimator due to vignetting. The experiment results also show that our valid region selection scheme does not improve the result or it even increases the error slightly.
### 3.1.5 Multiple-View Array Reconstruction

Our 4D light field reconstruction pipeline for the 2D *Lytro* raw image is illustrated in Fig. 3.18. The global grid is used to identify the centres of the microlens image array either from the 2D content raw image itself or the associated 2D reference raw image. Before re-sampling the 4D light field from an irregular 4D grid to a regular 4D grid, there are two pre-processing steps; removing the vignetting and converting the Bayer pattern raw image to the colour raw image with three channels. In fact, de-vignetting and demosacing play important roles in restoring a high quality 4D light field. We give a more in-depth discussion to the de-vignetting in the next session. In the following, we explain how we re-sample the 4D light field from the hexagonal microlens array image given the global grid.

#### 4D Light Field Re-sampling

In 1.5.4, we briefly explained how to extract the multiple-view image array. A real-world example processed using our reconstruction algorithm is shown in Fig. 1.23. In practice, the reconstructed 4D light field is in an irregular arrangement mainly due to two reasons. First, the microlens array is not perfectly aligned with the sensor pixels. Second, the microlens array is arranged in a hexagonal grid. Therefore, we re-sample the 4D light field in 2 steps. Firstly, we re-sample each microlens image to a square image with 9 by 9 pixels using a linear interpolation. Secondly, we exploit a simple gradient-based approach to re-sample the hexagonal grid to the square grid. This is illustrated in Fig. 3.19.

---

**Figure 3.17:** Colour coded RMSE map for the global grid. The RMSE is in pixel unit.
Figure 3.18: Our 4D light field reconstruction pipeline. Following the de-vignetting and demosaicing, the 4D light field is reconstructed with the global grid. The 4D light field is further re-sampled to generate the rectangular multiple-view array image.

Mathematically, the vertical and horizontal gradients $|G_v|$ and $|G_h|$ are defined by

$$|G_{ij}^v| = |I_{ij}^{left} - I_{ij}^{right}|, \quad (3.15)$$

$$|G_{ij}^h| = |I_{ij}^{up} - I_{ij}^{down}|. \quad (3.16)$$

where $i$ and $j$ are the coordinates inside the microlens image. The microlens image pixel $I_{ij}$ is interpolated along the direction that has less absolute gradient value. After re-sampling the hexagonal grid to the rectangular grid, we re-arrange the 4D coordinates to generate the multiple-view array image, as illustrated in Fig. 1.22.

Figure 3.19: Microlens image interpolation from its four adjacent microlens images. In contrast to linear interpolation, the gradient-based approach ensures a sharper edge inside the microlens image.
Table 3.2: The Image Quality Table based on SSEQ. Less score means a better quality, and the score range is between 0 to 100. R and NR stand for our proposed algorithm with the reference image and without the reference image, respectively.

See appendix C.2.1 for the implementation of the C++ source code of the blind global grid fitting.

3.1.6 Experimental Results

To quantitatively and objectively evaluate our multiple-view reconstruction algorithm, we use SSEQ [102] (Spatial-Spectral Entropy-based Quality) to measured the quality of reconstructed multiple-view images. Liu et al. [102] demonstrated that SSEQ is statistically superior to reference image quality assessment algorithms such as structural similarity (SSIM) and non-reference image quality assessment algorithms. The software implementation can be found from [103]. We compare our SSEQ results with state-of-the-art algorithms, Cho et al. [95] and Dansereau et al. [94]. As shown in Tab.3.2, our approach performs better than [94] overall and slightly worse than [95] in one case. In particular, our proposed light field reconstruction without the reference image algorithm produces consistently good results. Three examples are shown in Fig.3.20 and Fig.3.21. Our results are almost identical visually to Dansereau et al. [94] and Cho et al. [95].

3.2 Radiometric Calibration

Vignetting as described in Chapter 2, refers to the gradual fall-off of light intensity from the centre of the image captured by an imaging system [88]. It is a common imaging phenomenon and is usually corrected in the camera processing pipeline.
Figure 3.20: Top row: left is Dansereau et al.’s [94] result, right is our result. Bottom row: left is Cho et al.’s [95] result, right is our result. The image is the central view cropped from the reconstructed multiple-view light field image.

Figure 3.21: The chart example. Left column is Cho et al.’s [95] result. Middle is Dansereau et al.’s [94] result. Right is our result.
along with other processing steps such as demosaicing, distortion correction and noise reduction. The sources of vignetting can be classified into natural vignetting, optical vignetting, mechanical vignetting and pixel vignetting [88]. For conventional cameras, focal lengths, large apertures and large format sensors usually introduce more severe vignetting effects. An MLA-based light-field camera exhibits severe vignetting effects due to two factors. First, the pixels underneath each microlens can be seen as a large format sensor. As a result of natural vignetting, also known as the cosine-fourth law, the edge pixels receive much less light than the central ones. Secondly, a microlens-based light field camera has a constant, small $f$-number, usually as small as $f/2$. For such an optical configuration in a conventional camera, vignetting effects are always significant.

Vignetting in the MLA-based Light Field Camera

For the conventional cameras, vignetting normally only leads to a tiny intensity difference between adjacent pixels. In contrast, for the MLA-based light-field cameras, the vignetting leads to a significant intensity discrepancy inside the microlens image. We carry out a preliminary experiment to examine the light intensity fall-off across the 2D Lytro reference raw image. We define the uniformity factor $\eta$ as the ratio between average and maximum intensity value in a region of size $M$ by $N$,

$$\eta = \frac{\frac{1}{MN} \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} I_{ij}}{\max\{I_{ij}\}}. \quad (3.17)$$

We calculate the ratio at central and corner regions over a range of focal lengths from 5mm to 50mm and the results are shown in Fig. 3.22. As shown in Fig. 3.22, even for the central region, the uniformity factor $\eta$ is less than 80%. In addition, the uniformity drops almost by a factor of two from the central regions to the corner regions. More importantly, the uniformities of three colour channels are different with the same optical settings. One reason is the geometric relation is changed as discussed in Section 3.1.2. Another reason is chromatic aberration; the size of the exit pupil is slightly different at different wavelengths. This indicates that vignetting correction should be performed channel by channel.

---

1 The $f$ number or focal ratio is defined as the focal length divided by the aperture diameter.
Chapter 3. Light-Field Camera Calibration

3.2.1 Related Works

Vignetting correction is a well-explored topic in optics and computer vision. It is also known as flat-field correction (FFC) in astronomy. The most simple and efficient approach to FFC \[104\] can be written as

\[
I'(x, y) = \frac{I(x, y) - I_B(x, y)}{I_U(x, y) - I_B(x, y)} M. \tag{3.18}
\]

where \(I_B\) is a dark frame, \(I_U\) is uniform illumination image, \(I\) is the uncalibrated image, \(I'\) is the flat field corrected image and \(M\) is a normalizing constant.

Evidently, the above approach is simple and robust for any optical imaging system. However, this approach requires a large amount of non-volatile memory to store the reference images for different optical configurations, i.e. focal distance, exposure time. Furthermore, the reference images are normally generated with a complicated calibration process including averaging a sequence of images with the same illumination, optical settings, and temperature condition. Instead, accurate prediction of light intensity fall-off and modeling the vignetting effect can both significantly reduce the memory requirement and the calibration complications. Specifically, a quadratic polynomial function \[105\] and a hyperbolic function \[106\] have been proposed to approximate the light fall-off profile across images.

According to our knowledge, our work is the first attempt to model the vignetting effect for an MLA-based light-field camera. In \[94\], an MLA-based light-field camera...
decoding and calibration pipeline is proposed, but the vignetting is corrected by a traditional FFC approach. In contrast, our work explores the possibility of modeling and removing vignetting using a joint global and local parametric fitting.

### 3.2.2 Vignetting Modeling

In this section, we propose a joint 2D global polynomial fitting and local Gaussian quadratic fittings to model the vignetting of an MLA-based light field raw image. In principle, a more complicated 4D fitting model should be considered to accurately model the vignetting of the sampled 4D light field. In fact, our approach can be seen as a 4D fitting model which is separated into a 2D polynomial function and a 2D Gaussian quadratic function.

**Local Parametric Kernel Estimation**

Inspired by Kee et al. [58], whose work involved using a spatially smooth 2D kernel to estimate the image blur kernel to restore images, we choose a two-dimensional Gaussian function \( G(\Sigma, a) \) to fit the light intensity profile of each individual microlens image with a window size of \( S \) by \( T \) pixels. The orientation is controlled by a single parameter \( \rho \). All the parameters can be obtained by minimizing the energy function,

\[
\{\hat{\Sigma}, \hat{a}\} = \arg\min_{\Sigma, a} \sum_{m=0}^{S} \sum_{n=0}^{T} | | I_{s,t} - G(\Sigma, a) | |^2.
\]  

(3.19)

where the 2D Gaussian quadratic function is expressed as,

\[
G(\Sigma, a) = \frac{\exp(-x^T \Sigma^{-1} x)}{a |\Sigma|^{1/2}}.
\]  

(3.20)

The covariance matrix \( \Sigma \) is

\[
\Sigma = \begin{bmatrix}
\sigma_1^2 & \rho \sigma_1 \sigma_2 \\
\rho \sigma_1 \sigma_2 & \sigma_2^2
\end{bmatrix},
\]  

(3.21)

where \( x \) is the spatial coordinate vector in the microlens image. The Levenberg-Marquardt algorithm [107] is used to solve this non-linear optimization problem. Fig. 3.23 shows the fitting results for our simulated microlens image array. Our model gives residual errors that are consistently at an acceptable level (less than 10% error).
Figure 3.23: (a) The ground truth microlens array image generated by simulation. (b) Microlens image fitting results at different locations. The first row is the ground truth microlens image, second row is the parametric fitting result, the third row is residual errors. For the purpose of visualization, the errors are magnified by a factor of 10.

Global Parametric Fitting

The orientation of each microlens image varies smoothly in spatial coordinates. Polynomial functions are employed to model the global variations of the local parameters $a$, $\sigma_1$, $\sigma_2$ and $\rho$. To balance the approximation accuracy and computational complexity, we use a fourth order polynomial, in which 15 coefficients are estimated by a least square fitting. For example, for the polynomial coefficients $c_{ij}$ of the local parameter $\rho$ is obtained by solving the following equation,

$$
\{\hat{c}_{ij}\} = \arg\min_{c_{ij}} (\sum_{u,v=0}^{M,N} \||\rho_{uv} - \sum_{j=0}^{i} \sum_{i=0}^{j} c_{ij} x^i y^j ||^2).
$$

(3.22)

where $\rho_{uv}$ is estimated by our local parametric fitting and the size of the microlens array is $M$ by $N$. In addition, we discard the 10% estimated local parameters from the poorest fit microlens images to obtain the global vignetting fitting coefficients for the local parameters $a$, $\sigma_1$, $\sigma_2$ and $\rho$.

In Fig.3.24, we show an example of our fitting result which is arranged in a multiple-view image array format. The vignetting profile for the central view is very similar to the conventional camera vignetting profile, whereas the view at the boundary exhibits strong orientation due to the distortion of the exit pupil.

See appendix C.2.2 for the implementation of the C++ source code of the vignetting correction.
3.2.3 Experimental Result

In order to obtain a light field raw image with uniform illumination, we place a diffuser in front of the Lytro camera. A sequence of uniform illumination images and a dark frame with the same zoom and exposure settings are captured. The average image of the sequence is used for our modeling. To quantitatively analyse the performance of our proposed algorithm, we capture another uniformly illuminated image in a different light condition. Ideally, the reconstructed multiple-view array image should have a constant intensity value across all views in each colour channel. We use the Peak Signal to Noise Ratio (PSNR) to measure the uniformity of the reconstructed multiple-view image. The ground truth image $K$ is approximated as a uniform image of which the constant intensity value is the average intensity of the reconstructed multiple-view array image $I$. The PSNR is calculated as
\[ PSNR = 10 \log_{10} \left( \frac{M}{\sum \sum [I(i,j) - K(i,j)]^2} \right) \] (3.23)

where \( M \) is the maximum possible pixel value, e.g. \( M = 255 \) when the pixel has 8 bits depth. As shown in Tab. 3.3, we compare our result with FFC approach and without de-vignetting. For the comparison we use 7 by 7 views out of the 9 by 9 reconstructed multiple-view images. Our approach and FFC approach improve the uniformity around 3 \( dB \) and around 6 \( dB \), respectively. FFC approach outperforms our approach, this is because our fitting result is not able to accurately model the vignetting profile. However, we believe that the model accuracy can be significantly improved if the number of pixels under the microlens is increased.

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>FFC</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>23.7 ( dB )</td>
<td>28.8 ( dB )</td>
<td>25.3 ( dB )</td>
</tr>
<tr>
<td>Green</td>
<td>21.6 ( dB )</td>
<td>27.0 ( dB )</td>
<td>24.3 ( dB )</td>
</tr>
<tr>
<td>Blue</td>
<td>23.3 ( dB )</td>
<td>30.2 ( dB )</td>
<td>27.4 ( dB )</td>
</tr>
</tbody>
</table>

Table 3.3: The PSNR for the original, FFC corrected and our proposed multiple-view array image. The test raw image is uniformly illuminated.
Chapter 4

Depth From Light Field

Depth-sensing has been an active topic over decades both in academia and industry. Recently, extracting depth from light field has emerged as a new depth-sensing approach, which is of great interest in the computational imaging research field. Specifically, the portable MLA-based light-field camera such as Lytro and Ratrix, enable us to perceive the depth of the scene with a single-lens-sensor system in a snapshot. At the beginning of this chapter, we first overview available depth-sensing technologies including interferometry, Time-of-Flight (TOF), shape from shading, and triangulation. Next, we give a theoretical analysis for the depth resolvability of the MLA-based light-field camera. Then, we present our depth from light field algorithm in detail. In the experimental section, our results are compared with both the state-of-the-art algorithms and the commercial software.

Contributions Firstly, we give our theoretical analysis for the depth resolvability of the MLA-based light-field camera. Secondly, we present a simple and efficient implementation of depth from light field algorithm, and in particular an effective refinement scheme for the Lytro camera.
4.1 Overview of Depth Sensing Technology

Having depth sensing capability allows intelligent devices to perceive the world with an additional dimension. This enables a number of appealing applications such as human pose estimation, 3D object recognition, and indoor odometry. Thanks to advances in sensor technology and computational power, the depth sensing function has been integrated into consumer electronic devices. The Microsoft Kinect with an embedded depth sensing feature was the Guinness world record holder for the year 2011 as the fastest-selling consumer electronics device [108]. The depth sensing function is also emerging in mobile devices such as tablet [109] [110] [111] and glasses [112].

Generally, apart from depth from light field, there are four basic depth sensing technologies: triangulation, interferometry, time of flight, and depth from shading. In the following, we give a brief introduction to these fundamental depth sensing technologies.

4.1.1 Interferometry

Interferometry, which is based on coherent light interference, is able to measure a small fractional displacement between the reference and the sample. A typical optical setup for interferometry is illustrated in Fig.4.1. A pinhole device and a collimating lens are placed in front of the light source to produce plane wavefronts. The plane wavefronts are divided by the beam splitter. The interference pattern is formed by the wavefronts reflected by the sample and the mirror (reference target).

For monochromatic waves, the two beam interference equation is

\[ I = I_1 + I_2 + 2\sqrt{I_1 I_2} \cos(\Delta \Phi). \]  

(4.1)

where \( I_1, I_2\) are the irradiances of the reference and test beams respectively, \( \Delta \Phi \) is the phase difference between test and reference beams which is proportional to the distance disparity \( \Delta x = x_1 - x_2\). Interferometry is able to measure tiny displacements, as small as \( 10^{-9} \text{m} \). Therefore, interferometry has been widely used in precision metrology e.g. to measure the flatness of a surface. However, interferometry is sensitive to mechanical and thermal stabilization.
4.1.2 Time-of-Flight

A Time-of-flight (ToF) camera, as a subclass of LiDAR (light radar), measures the time interval between when light is emitted from the source and when it is reflected back to the camera. ToF camera modulates the light source and demodulates the reflected signal to infer the phase correlation between these two signals. The working principle of ToF camera, which is based on continuous-wave modulation, is sketched in Fig. 4.2. The illumination signal is modulated with a signal \( g(\omega t) \). After the emitted signal \( g(\omega t) \) reaches the object, the reflected signal \( ag(\omega t + \phi) \) travels back to the ToF camera with an amplitude attenuation \( a \) and a phase delay \( \phi \). The key question now becomes how to accurately estimate the phase delay \( \phi \) the value of which is proportional to the time of travel. The estimation is achieved by applying a correlation operation between the received signal \( ag(\omega t + \phi) \) and the reference signal \( g(\omega t + \tau) \). As a result, the correlation function \( H_\omega(\tau, \phi) \) is

\[
H_\omega(\tau, \phi) = a \int g(\omega t + \tau)g(\omega t + \phi)dt. \tag{4.2}
\]
The phase delay $\phi$ can be determined by electronically tuning the configurable parameter $\tau$ and observing the change of the correlation function $H_{\omega}(\tau, \phi)$.

Figure 4.2: ToF camera working principle. The time of travel of a signal between ToF camera and the object is measured from the phase correlation between a phase shifted version of emitted signal $g(\omega t + \tau)$ and the reflected signal $ag(\omega t + \phi)$.

The ToF camera is able to measure the depth at high accuracy regardless of the ambient light condition and the texture condition of the object. However, the correlation between the reference signal and the reflected signal has to be measured by the electronic circuit inside each individual sensor element. As a result, a typical ToF sensor element would have a much more complicated internal structure than that of a common CMOS image sensor. Therefore, the spatial resolution of a TOF sensor, which is currently limited by the size of the pixel, is as low as 300 by 300 pixels [113].

4.1.3 Shape-from-Shading

As illustrated in Fig 4.3, Shape-from-Shading refers to a methodology that uses a controlled illumination to acquire the reflected light intensity and estimate the shape of the object by matching a known reflectance model assuming the reconstructed surface is smooth [114]. Strictly speaking, stand-alone shape-from-shading can only recover the shape of objects. The advantage of shape-from-shading is fine detail over the continuous smooth surface can be reconstructed even if the test sample is made of both Lambertian and specular materials, such as metal-plastic parts.
4.1.4 Triangulation

Triangulation is probably the most widely used depth sensing method due to its ease of realization. The depth is determined by the geometry of the scene from two or more views. There are variants of triangulation methods such as the stereoscopic camera which mimics the human vision system. However, the fundamental principle remains the same. As illustrated in Fig.4.4, the 3D location of point $X$ can be determined from two views which are aligned vertically in this example. The disparity $p$ between two correspondence points $X'$, $X''$ and the real world depth $d$ has the following relation:

$$p = X_L - X_R = b \frac{d'}{d}, \quad (4.3)$$

where $b$ (known as the baseline) is the separation between two projection centres $O_L$ and $O_R$, $d'$ is the image distance. From Eq.4.3, we see that given the properties of the camera setup (baseline $b$ and image distance $d'$) and the disparity value $p$ of the correspondence points, the distance $d$ can be immediately obtained.

Finding the correspondences is not a trivial problem, because the image can be noisy, some parts of the image can have low contrast, seamless textures are hard to distinguish, no correspondence pair exists in occlusion regions, etc. As a consequence, most of the disparity maps generated by direct correspondence matching are inevitably noisy and unreliable. To improve the correspondence matching reliability, active illumination with specific patterns is widely used for a precise depth measurement. For example, the ground truth of the most recognized stereo evaluation dataset is obtained using structured light [115]. From the image processing point of view, a
common remedy is to produce a "regularized" disparity map using a global optimization scheme, such as Markov Random Field (MRF) [116] or variational method [117].

4.1.5 Depth From Light Field

As mentioned in the previous section, correspondence matching is the most critical step for triangulation or depth from stereo. Nevertheless, it is possible to have correspondences without matching. Nearly three decades ago, Bolles et al. [118] proposed to steadily move a camera horizontally and take photos with a fixed spatial interval to estimate the structure of the scene, namely structure from Epipolar Plane Image (EPI). The same idea has been re-implemented with modern devices and more advanced algorithms by Kim et al. [11]. This is actually equivalent to capturing the light field, more precisely, a 3D light field. Their idea is illustrated in Fig.4.5. In this multiple-view setup, the multiple image points of the same object point are sampled. If the object in the scene has a Lambertian surface and the sample grid is dense enough, multiple image points of the same object point would form a slant line when the 1D images (epiploar lines) are stacked as an image (EPI). A real world example is shown in Fig4.6. Evidently, there is no need to search for the correspondence points. Now, the disparity calculation is equivalent to estimating the slope of the slant line. This can also be seen as an extension of depth from stereo and sometimes is referred to as depth from light field.

However, it is worth pointing out that there is a fundamental difference between depth from EPI and depth from light field. Recall the light field transport discussion
Figure 4.5: For a multiple-view setup, the absolute depth is proportional to the slope in the epipolar slice. A larger slope means the object is closer to the camera and a zero slope means the object is at infinity. When the multiple-view images are converted to the 4D light field representation, the relative depth with respect to the reference plane (object plane) is proportional to the slope in the 2D light field slices. Specifically, the slope of the reference point which lies on the reference plane has a zero value. In contrast to point A, which is further away to the plane of focus and has a positive slope, point B, which is closer to the camera than the plane of focus, has a negative slope in the 2D light field slice.

in Chapter 1; the shape of the 2D light field slice is sheared by an angle which is proportional to the distance that it travels in free space from its original location. As shown in Fig.4.7, if light rays \( L_1(x_1, y_1, u_1, v_1) \) and \( L_2(x_2, y_2, u_2, v_2) \) converge at point \( P \), we have the following geometric relation,

\[
\begin{align*}
\frac{u_1 + \frac{x_1 - u_1}{d}z'}{u_2 + \frac{x_2 - u_2}{d}z'} &= \frac{\Delta x}{\Delta u} = \frac{z' - d}{z'}, \\
\frac{v_1 + \frac{y_1 - v_1}{d}z'}{v_2 + \frac{y_2 - v_2}{d}z'} &= \frac{\Delta y}{\Delta v} = \frac{z' - d}{z'}.
\end{align*}
\]

(4.4)

where \( d \) is the distance between the two planes that characterize the light field, \( z' \) is the depth of the object point \( P \) with respect to plane \( UV \), \( \Delta u = u_2 - u_1 \) and \( \Delta x = x_2 - x_1 \) are the disparity of angular and spatial coordinates respectively. For Eq.4.4, the depth \( z' \) can be estimated either by vertical slope \( \frac{\Delta v}{\Delta x} \) or the horizontal slope \( \frac{\Delta u}{\Delta y} \). The slope value (unit in pixels/view) is equal to the maximum pixel disparity (unit in pixels) of an object point among views divided by the number of views.
Figure 4.6: A real world example. (a). The multiple-view image array is reconstructed from the 4D light field captured by Lytro camera. (b) The central view of (a) with horizontal and vertical 2D light field slices. In 2D light field slices, the correspondence points forms stripes with different slopes.

Figure 4.7: The working principle of depth from light field. The depth of a point source can be estimated by calculating either the vertical or horizontal slope from the light field 2D slices.

One may immediately notice that Eq. 4.4 has a slightly different form compared to the triangulation formula shown in Eq. 4.3. The reason is that multiple-view images captured by a camera array or camera motion needs an additional alignment to reconstruct the captured light field, which operation is illustrated from Fig. 4.5 (b) to (c).
Comparison to depth from stereo  

First of all, these two approaches rely on the same principle: photo constancy. However there are still a few subtle differences. In depth from stereo, when the disparity between correspondences is zero, the real depth is at infinity. On the contrary, in depth from light field, when the disparity (slope) is zero, the real depth is at the plane of focus. The reason is illustrated in Fig.4.5, these two approaches use different references. Comparing the accuracy of these two approaches is a complex issue since each approach may use different configurations such as optical settings, geometric settings, sensor resolutions. In general, depth from light field doesn’t require correspondence search in 2D which means it is naturally a more efficient and robust approach.

4.2 Theoretical Analysis

As the MLA-based light-field camera has already become an off-the-shelf device, there is growing interest in recovering depth from the captured light field data. The first prototype of an MLA-based light-field camera was made in 1992 for the purpose of depth estimation by Adelson and Wang [1]. The captured image was used to reconstruct sub-aperture images and the depth was estimated by the standard multi-view algorithm. In this section, we first show how the depth information is encoded in the microlens array raw image. Next, we give an insight into the factors that influence the depth resolvability.

4.2.1 Depth Cue from Raw Images

As discussed in Chapter 1, the MLA-based light-field camera preserves the angular information of the light rays by placing a microlens array behind the main lens. As a result, the geometric information of the scene is encoded in the microlens array raw image. To demonstrate the connection between the pattern of the microlens array images and the depth of the scene, we carried out a simple experiment. As shown in Fig.4.8, the Lytro camera is mounted on a motorized stage. A vertical white-black stripe pattern is displayed on the LCD screen. We fixed the camera between 4th and 5th position and deliberately adjusted the focal lengths setting to ensure the LCD screen is focused at this point. We slide the camera from the first to the tenth position step by step to capture a sequence of light field images. As shown in Fig.4.8(b), when the Lytro camera moves further away from the original plane of focus, the range of the microlens image pattern extends. Clearly, this is the cue that we can use to extract
depth from the microlens array raw image.

Figure 4.8: (a) Vertical white-black stripes are displayed on the LCD screen. The distance between the LCD screen and the Lytro camera is varied by sliding the Lytro camera from the 1st position to 10th position. The separation between each position is 30 mm. (b) The microlens image array patterns varies from 1st position to 10th position, which conveys depth information.

However, it is impractical to directly extract the depth information from the microlens array raw image. The reason is that the number of pixels in a microlens image is as few as 9 by 9. In addition, the Bayer pattern further reduces the effective spatial resolution. Another real-world example is shown in Fig.4.9, the direct analysis of the microlens image pattern becomes extremely difficult and inaccurate. Therefore, we need to first reconstruct the 2D microlens array raw image to a 4D light field data structure using the process discussed in Chapter 3. Next, we extract the depth from the 2D light field slice which is discussed in the previous section.

### 4.2.2 Depth Resolvability

For an MLA-based light-field camera, the encapsulated 4D light field inside the camera is jointly characterized by the exit pupil plane of the main lens and the microlens array plane. Therefore, to estimate the real world depth, we first extract the depth from the light field captured inside the camera and then convert it to the real world depth. Therefore, combining Eq.4.4 and the lens-maker-formula, the real-world depth \( z \) is
Figure 4.9: A real-world example shows the depth information is embedded in the Lytro camera raw image. However, the depth is difficult to extract due to the limited resolution of a microlens image. (b)-(e) A close-up of (a) at different regions. (c) has a larger pattern region than (e) which indicates a larger distance with respect to the plane of focus. For the lack of texture region like (b), there is no depth information encoded in the raw image.

\[
\begin{align*}
z &= \left(\frac{1}{F} + \frac{1}{z'}\right)^{-1} = \frac{Fd}{d + F(1 - k)},
\end{align*}
\]

where \(F\) is the focal length of the main lens, \(d\) is the distance between exit pupil plane and the MLA plane, \(k\) is the slope of a the stripe formed by point correspondences. According to the sign convention [73], \(d\) is negative. Notice that we deliberately choose \(k = \frac{\Delta x}{\Delta u}\) rather than \(k = \frac{\Delta u}{\Delta x}\) to avoid its value growing to infinity when \(z' = d\). In a practical setup, another point of concern is that most of the MLA-based light-field cameras use different spatial sampling rates for \(XY\) and \(UV\) planes. As a result, the slope measured from the 2D light field slices is scaled by a factor \(s\). Lets suppose the diameter of the exit pupil of the main lens is \(A\), the width of pixel is \(p\) and there are \(N\) by \(N\) pixels under each microlens, then the value of scale factor \(s\) is equal to \(\frac{A}{N^2p}\).

Intuitively, the light rays converging before the microlens array plane have negative slope values whereas the light rays converging after the microlens array plane have positive slope values. According to Eq.4.5, three special cases of real world depth with corresponding slope values are listed in Table 4.1. The range of slope \(k\) extracted from 2D light field slices is within the range of \([\frac{F + d}{F(1-k)} \cdot s, 1 \cdot s]\). Lets take a con-
crete example to show the relation between slope $k$ and real-world depth $d$. Four different configuration profiles of an MLA-based light-field camera are listed in Tab. 4.2. Based on Eq 4.5, the relation between the depth $z$ and the slope $k$ is plotted in Fig. 4.10 under different focal length and plane of focus settings.

$$\begin{align*}
\text{Slope} & \quad \text{Inside-camera depth} & \quad \text{Real world depth} \\
k = 1 \cdot s & \quad |z'| \gg |d| & \quad z = F \\
k = 0 & \quad |z'| = |d| & \quad z = \frac{F_d}{d+l} \\
k = \frac{F+d}{F} \cdot s & \quad |z'| = F & \quad z = +\infty
\end{align*}$$

Table 4.1: Real world depth with three special scenarios

<table>
<thead>
<tr>
<th>Slope</th>
<th>Inside-camera depth</th>
<th>Real world depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 1 \cdot s$</td>
<td>$</td>
<td>z'</td>
</tr>
<tr>
<td>$k = 0$</td>
<td>$</td>
<td>z'</td>
</tr>
<tr>
<td>$k = \frac{F+d}{F} \cdot s$</td>
<td>$</td>
<td>z'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>config. 1</th>
<th>config. 2</th>
<th>config. 3</th>
<th>config. 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>20</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>$A$</td>
<td>10</td>
<td>10</td>
<td>12.5</td>
</tr>
<tr>
<td>$N$</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$p$</td>
<td>0.025</td>
<td>0.025</td>
<td>0.025</td>
</tr>
<tr>
<td>$z$</td>
<td>2000</td>
<td>1000</td>
<td>2000</td>
</tr>
</tbody>
</table>

Table 4.2: Four configuration profiles of an MLA-based light-field camera. Length unit is in mm.

Figure 4.10: The theoretical relation between the real-world depth $z$ and the slope $k$. The sampling interval of $k$ is 0.02.
As shown in Fig. 4.10, the curve of the slope $k$ with respect to depth $z$ is plotted with four configurations. For configurations 1 and 3, and configurations 2 and 4, the plane of focus is at distance around 2000 mm and 1000 mm, respectively. Generally, the depth resolvability is much lower when $k$ is negative. Specifically, we can see that a longer focal length $F$ and a longer separation $d$ (configuration 4) result in a better spread of values of $z$ versus $k$. However, the trade off of using a longer focal length is that the field of view (FOV) is reduced.

**Error Analysis** From Eq. 4.5 we can see the total depth error $\varepsilon_z$ is determined by the error of focal length $\varepsilon_F$, the error of separation between exit pupil plane and the microlens array $\varepsilon_d$, and the error of slope estimation $\varepsilon_k$. Using a first-order Taylor series approximation, the error model for $\varepsilon_z$ is a linear combination of the error components $\varepsilon_F, \varepsilon_d, \varepsilon_k$, where the sensitivity coefficients $c_F, c_d, c_k$ are the partial derivatives of $z$ with respect to $F, d$ and $k$. The absolute errors $\varepsilon_z$ and relative depth errors $\varepsilon_Z$ are calculated as,

\[
\varepsilon_z = |c_F|\varepsilon_F + |c_d|\varepsilon_d + |c_k|\varepsilon_k,
\]

\[
= z^2 \left( \frac{1}{d^2} \varepsilon_d + \frac{1}{F^2} \varepsilon_F + \frac{1}{|d|} \varepsilon_k \right),
\]

\[
\approx \frac{z^2}{|d|} \varepsilon_k = \frac{-F^2d}{(d + F(1-k))^2} \varepsilon_k, \quad (4.6)
\]

\[
\frac{\varepsilon_z}{z} \approx \frac{z}{|d|} \varepsilon_k = \frac{-F}{d + F(1-k)} \varepsilon_k. \quad (4.7)
\]

Taking into account that the two error sources $\varepsilon_F$ and $\varepsilon_d$ are generally small and can be reduced by careful calibration, $\varepsilon_k$ can be treated as the major error source. From Eq. 4.6 and Eq. 4.7 we can see that both absolute depth error $\varepsilon_z$ and relative depth error $\varepsilon_Z$ is a function of focal length $F$, two-plane distance $d$ and slope estimation error $\varepsilon_k$. If we assume a constant slope estimation error $\varepsilon_k$, the relative depth error $\varepsilon_Z$ with regards to the value of $k$ is plotted in Fig. 4.11. Except the focal length $F$ and two-plane distance $d$ are different, other parameters are kept as the same as described in Tab. 4.2.

From Fig. 4.11, we can see the relative error $\varepsilon_Z$ is generally small when slope $k$ is positive and it decreases when the slope value increases. On the other hand, the relative error $\varepsilon_Z$ grows significantly when slope $k$ is negative. With the same plane
of focus, a longer focal length reduces the relative error $\frac{\epsilon_\ell}{Z}$ when the slope value is negative, but it also narrows down the field of view. The analysis of relative error $\frac{\epsilon_\ell}{Z}$ also gives us the sense of what range of slope value $k$ is reliable. For example, when the slope estimation error $\epsilon_k$ is 0.01 and the plane of focus is 4000 mm away, the negative slope value becomes extremely unreliable for the focal length over the range from 5 mm to 50 mm. In depth from stereo, the baseline has a large impact on depth estimation accuracy. Longer baselines will be used for long distance depth sensing. However, the MLA-based light-camera has a tiny baseline which is the diameter of the exit pupil divided by the horizontal or vertical number of views [96]. This is the reason why the depth resolvability is low when the real-world depth increases. Unfortunately, the baseline in the MLA-based light field camera is fixed.

The slope estimation error $\epsilon_k$ is directly connected to the vertical sampling (the number of pixels under the microlens) and horizontal sampling (the size of the microlens) rates. If we have a fixed sensor resolution, these two sampling rates cannot be increased at the same time. The depth resolvability for MLA-based light-field camera

Figure 4.11: The relative error curves as a function of the measured slopes under different optical configurations. For the upper and lower rows, the planes of focus for the main lens are 1000 mm and 4000 mm, respectively. The unit of $k$ is in pixels/view.
is a complex issue as the depth resolvability is also spatially variant. Image points of the main lens positioned on the same plane orthogonal to the optical axis may have different depth resolvability. This is illustrated in Fig. 4.12.

![Fig. 4.12: Ray diagrams of microlens images. For (a), (b), and (c), the patterns of the microlens image are nearly identical. In contrast, for (d), (e), and (f), the patterns of the microlens image reveal the depth information. In both cases, the object points have the same depth. The ray diagrams are produced using our proposed simulation framework discussed in Chapter 2.](image)

There are a few other factors that degrade the slope estimation accuracy, such as optical aberrations. For example, field curvature \[88\] bends the flat focal plane to a curved surface. This effect makes the estimation of off-axis image points by the main lens inaccurate. Astigmatism \[88\], arising from the difference of optical power in tangential and sagittal planes, makes the depth estimation from vertical and horizontal 2D light field slices unequal.

### 4.3 Algorithm Design and Implementation

In this section, we describe the implementation of our depth from light field algorithm. As depicted in Fig. 4.13, we divide it into 3 steps, slope estimation, fusion and refinement. First, we evaluate two slope estimators; brute-force search and structure tensor. Based on our simulation results, we analyse the performance for each slope estimator and explain why the brute-force search is preferred. Next, we present the way that we fuse the two disparity maps into a single high-quality disparity map. Finally, in order to improve the final quality of the disparity map extracted from the
**Lytro** raw data, we propose a global refinement scheme based on Markov random field (MRF).

![Image of workflow](image)

Figure 4.13: The work-flow of our depth from light field algorithm. We rearrange the 4D light field data to horizontal and vertical 2D light field slices as the input. The output is a dense disparity map. There are three steps inside our algorithm: slope estimation, fusion and refinement.

### 4.3.1 Slope Estimator

Generally, there are two different approaches for measuring slopes from 2D light field slices or EPIs; brute-force search [11][119] and structure tensor [49]. As the name indicates, the idea of brute-force search is that possible results are exhaustively evaluated and the best candidate is selected. In contrast to the brute-force search, the structure tensor approach calculates the image’s internal structure based on image gradient components manipulations. In the following, we give an in-depth discussion and evaluation of these two approaches.

**Brute-force search** Correspondence matching by brute-force search has been used extensively in depth from stereo and depth from multiple-view. In a similar fashion, determining the slope by brute-force search has been applied in depth from light field [11][119]. Due to the photo constancy assumption, rays that come from the same location of the object should have similar intensity values. Considering the horizontal 2D light field slice, we define the ray sample set $R_{y}^{(x,k)}$ and its mean $\mu_{k_{xy}}$ and variance $\sigma_{k_{xy}}^2$ as
Figure 4.14: A single slope estimation. (a) We can only obtain the minimum variance of specific sampled rays (white dots) along the true slope. The yellow dot is the reference ray. (b) Plot of variance with respect to slope.

\[ R_{yv}(x,k) = \{ l(x + uk, y, u, v) | u = -N, \ldots, N \}, \]  

(4.8)

\[ \mu_{kyv} = \frac{1}{2N + 1} \sum_{u=-N}^{N} l(x + uk, y, u, v), \]  

(4.9)

\[ \sigma_{kyv}^2 = \frac{1}{2N} \sum_{u=-N}^{N} |l(x + uk, y, u, v) - l(x, y, 0, v)|^2, \]  

(4.10)

where rays are in two-plane representation \( l(x, y, u, v) \), \( k \) is the slope, \( yv \) indicates the rays are sampled at the 2D horizontal light field slice \( L_{yv} \) (fixed coordinates of \( y \) and \( v \)), and we have totally \( 2N + 1 \) vertical views. As illustrated in Fig. 4.14, the idea of brute-force search is to evaluate all possible uniformly discretized slope values and select the slope that produces minimum variance. Ideally, the variance of the ray samples that are sampled along the true slope should be zero. In practice, while vignetting, discretization, occlusion, sensor noise, etc increase the variance, according to our experiments the true slope still corresponds the minimum variance. Mathematically,

\[ \hat{k}_{xyv} = \arg\min_{k \in [k_{min}, k_{max}]} \{ \sigma_{kyv}^2 \}, \]  

(4.11)

where we constrain the possible slope range between \( k_{min} \) and \( k_{max} \). In practice, noise is a major factor that degrades the accuracy of brute-force search. There are several possible solutions to reduce the noise influence. Tao et al. [119] proposed to average the variances in a small patch as the final variance estimation. Kim [11] proposed to use an Epanechnikov kernel as the error function instead of the quadratic form for the
purpose of reducing the influence of the outliers. In addition, the intensity value of the reference ray is iteratively updated in a mean-shift manner to mitigate the noise.

**Structure Tensor** The structure tensor has been acknowledged as a powerful tool to analyse the local orientation of neighborhood pixels exhibiting a single orientation. For a continuous 2D signal, the ratio between two gradient components is sufficient to describe the orientation. However, for a discretized 2D signal, e.g. digital image, the structure tensor as a blurred version of the covariance matrix of image derivatives, has proved to be a robust estimator. Wanner et al. first applied it to analyse the structure of 2D light field slices. If \( I_x \) and \( I_y \) are the horizontal and vertical partial derivatives of the Image \( I \), the 2D structure tensor \( J \) is expressed as

\[
J = \begin{bmatrix}
J_{xx} & J_{xy} \\
J_{xy} & J_{yy}
\end{bmatrix} = \begin{bmatrix}
G_\sigma \ast (I_x I_x) & G_\sigma \ast (I_x I_y) \\
G_\sigma \ast (I_x I_y) & G_\sigma \ast (I_y I_y)
\end{bmatrix}, \tag{4.12}
\]

where \( G_\sigma \) represents a Gaussian blurring with kernel size \( \sigma \).

![Figure 4.15: Work-flow of the structure tensor operator. Three smoothing stages (1, 3, 5) are involved to ensure robustness of the estimator.](image)

In Fig.4.15 we show the detailed scheme to compute the structure tensor proposed in and implemented in an open source library. The input is a 2D image \( I \). The smoothing configurable parameters: outer scale \( \tau \) and the inner scale \( \sigma \), determine the blur kernel size at input and output stages. The output is the structure tensor matrix \( J \) which is a symmetric matrix. There are five building boxes inside the structure tensor operator including outer scale blurring, computing gradients, 1D low pass filtering, multiplication, and inner scale blurring. The cascaded blurring operations placed at different stages ensure the orientation is estimated from the neighborhoods rather than a single pixel and therefore, less sensitive to noise. The slope \( k \) now can
be obtained by [114],

\[ k = \tan \left( \frac{1}{2} \arctan \left( \frac{2f_{xy}}{f_{xx} - f_{yy}} \right) \right). \] (4.13)

Another important property of the structure tensor is that we can use coherency \( c \) ranging from 0 to 1 to measure confidence of the estimation,

\[ c = \sqrt{\frac{(f_{yy} - f_{xx})^2 + 4f_{xy}^2}{f_{xx} + f_{yy}}}. \] (4.14)

**Evaluation** To evaluate the accuracy of these two slope estimators, we synthetically generate an image test set. Each image contains a single slant edge structure. The slope range is from -2 to 2. There are two grey levels \( A_1 \) and \( A_2 \) in each image. The contrast \( \eta \) is defined as the ratio between \( A_1 \) and \( A_2 \). In addition, we add independent and identically distributed (i.i.d.) Gaussian noise with standard deviation \( \sigma_n \) to the image test set. If we use the absolute difference between the two grey levels as the signal amplitude \( A = |A_1 - A_2| / 2 \), the signal to noise ratio (SNR) is calculated as

\[ \text{SNR} = 20 \log_{10} \left( \frac{A}{\sigma_n} \right). \] (4.15)

Figure 4.16: Synthetic data set for evaluating the accuracy of slope estimation by different algorithms. The range of the slope is from -2 to 2. Signal to noise ratio is from 15dB to 30dB. For the above images, the lower grey level is 96 and the contrast is 2. The unit of \( k \) is pixels/views.

In our evaluation experiments, the structure tensor and brute-force search approaches are applied to the synthetic data set with different configurations including
noise level and the number of views. While the contrast also affects the estimation results, we use a constant contrast value ($\eta = 2$). For the brute-force search, we use 64 discretized values (labels) to detect the best slope candidate. For the structure tensor, we use inner scale $\sigma = 2$ and outer scale $\tau = 4$. The root mean square error (RMSE) of slope estimation results are presented in Fig.4.17. For each test, we run 128 iterations with different seeds for generating the random noise.

For both brute-force search and structure tensor approach, the RMSE grows when the SNR drops, especially at higher slope value regions. In general, brute-force search approach slightly outperforms structure tensor approach, particularly for a lower number of views. The reason is because the structure tensor approach requires sufficient number of pixels to analyse the structure of the image, whereas brute-force search does not have such a requirement. For both approaches, increasing the number of views does help improve the estimation accuracy. However, this effect becomes less obvious when the number of views is larger than 9. For example, at $SNR = 20dB$ with brute-force approach, the RMSE reduced by 0.007 $\text{pixels/views}$ when the number of views is changed from 5 to 7, while the RMSE reduced by 0.002 $\text{pixels/views}$ when the number of views is changed from 11 to 13.

The goal of our depth from light field implementation is to allow fast depth map generation. Although the structure tensor estimator is fast and accurate, we are in favor of using the brute-force search estimator. The first reason is that it does not require to tweak the outer scale and inner scale parameters to accommodate the noise as the structure tensor approach. The second reason is that it can be easily implemented in parallel computing devices, e.g. (Graphical Processing Unit) GPU or Field Programmable Gate Array (FPGA).

In terms of estimation accuracy, we would like to use as many labels as possible in brute-force search. But on the other hand, we would like to use as few labels as possible to reduce computational burden. To determine the proper number of labels, we test the performance of the brute-force estimator with different number of labels: 4, 8, 16, 32, 64, 128. In theory, due to the discretization, the average estimation error $e$ and the number of labels $M$ have the following relation,

$$e = \frac{1}{4M}.$$  \hspace{1cm} (4.16)

In Fig.4.18, our numerical simulation results show that when the SNR is higher than $25dB$, the estimation accuracy is close to the theoretical bound. However, we
Figure 4.17: The RMSE of brute-force search and structure tensor in slope with varying SNR and number of views. The first and second rows are the results from brute-force search approach. The third and fourth rows are the results from structure tensor approach. The colour bar shows the RMSE levels in pixels/view unit.
find that 32 labels achieves the best balance between computational cost and accuracy. And at the low noise case, the number of 64 labels is a better option to achieve higher accuracy but also at the expense of doubling the computational time. When the number of labels beyond 64, the RMSE is roughly constant.

### 4.3.2 Disparity Map Fusion

At this stage, we have two disparity maps obtained from vertical and horizontal 2D light field slices by brute-force slope estimator. We propose a simple fusion algorithm to produce an accurate disparity map by fusing these two disparity maps. Since vertical (horizontal) structures of the scene is not preserved in the horizontal (vertical) light field slices, a good fusion algorithm could potentially improve the quality of the disparity map significantly. A real-world example is shown in Fig. 4.19, strong vertical edges can be observed from the horizontal disparity map, whereas horizontal edges can be observed from the vertical disparity map. Therefore, we propose to build a confidence map associated to each disparity map.

Concretely, we use a 1D horizontal gradient with the true slope in 2D light field slices to measure the confidence. The horizontal confidence $c_{h_k}$ and vertical confidence $c_{v_k}$ are calculated as
Figure 4.19: Disparity map fusion with the scene of flower2. (a) Horizontal disparity map. (b) Vertical disparity map. (c) Central view. (d) Fused disparity map. The unreliable pixels are masked to enhance visualization.

\[
ch_{k_{xy}} = |\mu_{k_{+1,y}} - \mu_{k_{xy}}|,
\]

(4.17)

\[
v_{k_{xy}} = |\mu_{k_{xy+1,u}} - \mu_{k_{xy,1}}|.
\]

(4.18)

We choose the final slope value with the higher confidence,

\[
\hat{k}_{xy} = \begin{cases} 
\hat{k}_{xyu} & v_{k_{xyu}} > ch_{k_{xyu}}, \\
\hat{k}_{xyv} & v_{k_{xyv}} < ch_{k_{xyv}}.
\end{cases}
\]

(4.19)

We also record the associated confidence value \(c_{xy}\), which will be used in the next stage.

\[
c_{xy} = \begin{cases} 
c_{xyu} & v_{k_{xyu}} > ch_{k_{xyu}}, \\
c_{xyv} & v_{k_{xyv}} < ch_{k_{xyv}}.
\end{cases}
\]

(4.20)
An example of our fusion algorithm is shown in Fig. 4.20. The data comes from a single row of the image (disparity map). We use the ground truth to calculate the mean absolute error (MAE) of slope estimation. The MAE of horizontal and vertical estimation are 0.039 pixels/view and 0.038 pixels/view, respectively. The MAE of simply averaging these two results is 0.067 pixels/view. By adaptively selecting the results from horizontal and vertical results, the MAE of the final result is reduced to 0.027 pixels/view. However, the possible minimum MAE by combining horizontal and vertical results is 0.019 pixels/view. Although our algorithm is not able to achieve the best possible result, it is efficient and more importantly, it consistently produces a more accurate result than a single direction estimation. See experimental results section for the detailed results.

![Graph](image1.png)

Figure 4.20: The MAE of slope estimation results from horizontal 2D light field slices (top), vertical 2D light field slices (middle), our fusion algorithm (bottom). This example takes the 285th row of the HCI Buddha disparity maps. The MAE is in pixels/view unit.

### 4.3.3 Disparity Map Refinement

As discussed previously, due to insufficient illumination and texture-less objects, most of the disparity maps extracted from *Lytro* raw images are quite noisy. An example is shown in Fig. 4.21(a). To effectively replace the unreliable disparity values with a more confident estimation, we develop a refinement scheme. This idea is not novel, and it
has been extensively used in stereo matching refinement, known as the multi-labeling problem. It also has been used in depth from light field \cite{119, 122}. Our contribution is that we propose a simple and efficient refinement scheme. More importantly, our approach ensures a sharper disparity map after refinement than the state-of-the-art algorithms \cite{119, 123}.

As shown in Fig.4.21, the confidence map obtained from the previous step is able to help us to identify those pixels that are most likely to be unreliable. Higher confidence threshold leads to a cleaner image but fewer valid pixels. As shown in Fig.4.21(d), setting the threshold to 4 would lead to the percentage of total valid pixels less than 25%, for instance. The optimal threshold depends on the quality of the light field raw image. We normally use the threshold that selects 20% to 30% of total number of pixels as valid pixels. This pre-filtering operation is implemented into the graph construction.

We use a MRF to propagate the disparity value of reliable pixels to unreliable regions. Concretely, the possible slope values are discretized to a finite number of values (e.g. 64 labels ranging from 0 to 63). For each pixel, there is a cost associated with the label which is defined in Eq.4.21. Our pre-filtering strategy is that for those unreliable pixels we use a constant zero value cost over all labels in the cost-volume.
Figure 4.22: A unidirectional graph constructed by MRF with our binary weighting edge function. Grey nodes are in low confidence and green nodes are in high confidence. Missing edges indicate the edge weight is zero.

As a typical inference problem (MRF), we treat each pixel as a node (variable) and an edge represents the interaction between each node in the graph. Assuming we have a set of labels $\mathcal{L} = \{1, \cdots, m\}$, the node index is $x = (i, j)$, $l_x \in \mathcal{L}$. The total cost assigning a specific label to each node, also called the data term, is expressed as

$$E_{data} = \sum_x g(f(l_x)),$$  \hspace{1cm} (4.21)

where function $g(k)$ is defined in $4.18$, function $f(l)$ maps the label value to the slope value. Keep in mind our goal of refinement is to propagate the value of high confidence pixels to its neighbors which have low confidence. In MRF, a regularization term is normally used to integrate the prior knowledge into the model. In our case, the prior knowledge is that the disparity map should be smooth overall. Mathematically, the smoothness term is defined as

$$E_{smoothness} = \sum_x \sum_{x' \in \mathcal{N}(x')} W_{x,x'} h(l_x - l_{x'}),$$  \hspace{1cm} (4.22)

where $\mathcal{N}(x')$ is the set of nodes adjacent to $x$. $W_{x,x'}$ is a weighting factor that determines the relationship between $x$ and $x'$. We use $L1$-norm as the function $h$. Combining Eq.4.21 and Eq.4.22 the total cost or energy function $E$ is

$$E = E_{data} + \lambda E_{smoothness},$$  \hspace{1cm} (4.23)

where $\lambda$ is a factor balance the data term and smoothness term. Based on our experiments, we use 0.01 for most of the cases. To construct a simple graph, we use a binary
edge weight function. As illustrated in Fig. 4.22, 0 (unconnected nodes) for those high confident pixels. Meanwhile, we set 1 (connected nodes) for those pixels that their edges are connected to low confidence pixels.

To solve this optimization problem, we adapt the graph cuts solver using the alpha-expansion and swap algorithms [124], the implementation of which is provided as an open source library [125]. We use the original disparity map as the initial condition. Fig. 4.23 shows the total costs or energy functions of 7 selected scenes are reduced significantly after MRF optimization. The image size of the original disparity map is 640 by 640. Although the computational time is dependent on the content of the graph we build, the computational time has small variations over our datasets (See Tab. 4.3). We demonstrate that our refinement scheme produces high quality and sharp depth maps. These refinement results are shown in the experimental section.

See appendix C.3.1 to C.3.4 for the implementation of the C++ source code of our proposed depth from light field.

![Bar chart showing the total cost of the MRF graph before and after MRF Optimization](image)

Figure 4.23: Comparison of the total cost of the graph before and after MRF Optimization.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Toy Map</th>
<th>Flower1</th>
<th>Flower2</th>
<th>Guitar</th>
<th>Squirrel</th>
<th>Office</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Time (ms)</td>
<td>9645</td>
<td>8355</td>
<td>5587</td>
<td>8545</td>
<td>9399</td>
<td>7045</td>
<td>8869</td>
</tr>
</tbody>
</table>

Table 4.3: The parameter configuration and computational time based on our computational platform.
4.4 Experimental Results and Discussion

4.4.1 Introduction

In this section, we give a brief introduction to the datasets used for our experiments and the specification of our computational device. There are two sources of light field data used in our experiments; the light field dataset from Heidelberg Collaboratory for Image Processing (HCI) \[126\] and raw files from the Lytro camera either captured by ourselves or obtained from the Internet \[127-129\].

**HCI Dataset** Although there are a few light field datasets publicly available, the HCI light field dataset is the only one that provides a densely sampled light fields with ground truth. Two different approaches were used for generating the HCI light field dataset, namely two categories Blender and Gantry. In Blender, 7 scenes were rendered by the computer graphics software Blender \[130\]. In Gantry, the light fields of 6 scenes were physically captured by moving a DSLR with a stepper-motor. A structured light scanner was used to measure the 3D polygon meshes of the object in the scene. The ground truth depth information was obtained by transforming the polygon meshes based on the relative pose estimation between the 3D structured light scanner and the light field camera. In Gantry, a binary mask is associated with each dataset, indicating the regions where scene geometry are not known due to limitation of the measurements. Each light field dataset contains 9 by 9 views (angular samples). The triangulation equation was used to do the conversion between the real-world depth and the slope (disparity),

\[
p = b\frac{d'}{d} - \delta,
\]

where \( p \) is the slope (disparity), \( d' \) is image distance, \( d \) is the real-world depth, and \( \delta \) is the shift between two neighboring images. The difference between Eq.4.3 and Eq.24 is the additional term \( \delta \) which is used to convert the light field representation to multiple-view representation as we discussed in Section 4.1.5. Readers are referred to \[131\], for the detailed implementation of their light field datasets.

**Lytro Dataset** To evaluate the performance of our algorithms, we include 8 light field raw images captured from the hand-held light-field camera Lytro. These examples are captured with 4 different Lytro cameras by different users. For these Lytro
raw images, the detailed optical and sensor settings can be found in appendix A. The detailed description of Lytro camera and how to obtain the raw data from the Lytro camera can be found in Section 1.5. How to extract the 4D light field from the Lytro 2D raw image is described in Chapter 3. A total number of 9 by 9 views can be extracted from the Lytro 2D raw image. Unfortunately, there is no ground truth data for our examples. Instead, we compare our results with the state of the art algorithms [119,123] and the results from the latest version (5.0.1) commercial software provided by Lytro.

**Computational Resources** Our computational device is a moderate PC. Specifically, it is equipped with an Intel® Core™ i5 3.3 GHz CPU and 8 GB RAM. Operation system is 64 bit Ubuntu 14.04 and the complier is GCC 4.9.2.

### 4.4.2 HCI Light Field Dataset Results and Discussion

Our proposed algorithm is first evaluated on the HCI light field dataset. Excluding one dataset in which the object is transparent, a total of 12 datasets are evaluated. The overall quality of the HCI light field dataset is good. Therefore, there is no need to refine the results after pixel disparity estimation. We use the mean square error (MSE) as the error metric. We show detailed quantitative disparity estimation and visual results on the HCI light field dataset in Tab.4.4, Tab.4.5, Fig.4.25. Besides our approaches, we also include multiple-view stereo and structure tensor approach [132] for comparison.

The HCI dataset absolute depth error maps are shown in Fig.4.25 and the corresponding optical configurations are shown in Tab.4.6. We classify the major error sources into three categories,

- Most of the large errors can be found at the boundary of the object. The reason is our simple slope estimator is not able to accurately detect the pixels from regions where there are two overlapping slopes due to occlusions. This is also a well-known problem in stereo vision.

- Another major source of error can be found in regions which are texture-less or the reflection property of the surface is not Lambertian. Active illumination would be helpful to reduce this kind of error.

- The quantization error from our slope estimator. However, these errors are normally very small.
Comparison to Multiple-View Stereo  In contrast to our approach, multiple-view stereo uses all the views to estimate the disparity (slope). Mathematically, the stereo matching cost is

$$
\sum_{v=-N}^{N} \sum_{u=-N}^{N} |l(x + uk, y + vk, u, v) - l(x, y, 0, 0)|^2. \quad (4.25)
$$

The sampling patterns for our approach and multiple-view stereo are compared in Fig. 4.24. The main difference is that multiple-view searches the correspondence in 2D, whereas our approach searches the correspondence in 1D. For the real-world datasets, multiple-view stereo outperforms our approach in some datasets such as Couple and Cube. The reason is that multiple-view stereo, which uses all views, naturally reduces the influence of noise. However, multiple-view stereo is computationally expensive. Suppose we have N by N views, the computational time for multiple-view stereo is $O(N^2)$ whereas it is $O(2N)$ for our approach. On the other hand, our approach only uses the cross-hair of the multiple-view array. For the case of 7 by 7 views, 70% light field data is redundant for disparity estimation.

Comparison to Structure Tensor  The real-world datasets (the category of Gantry), are noisier or have less texture than the synthetic datasets (the category of Blender). The structure tensor approach, which involves three blurring operations, also reduces the influence of noise. Nevertheless, we find that our results can be further improved by employing a denoising filtering either with the light field 2D slices or the original multiple-view images depending on the source and the level of the noise. However, to ensure our disparity estimators are robust and simple, we prefer not to involve any specific denoising operation at the intermediate stage. For those low quality and noisy data sets, such as the one from the Lytro camera, we use Markov Random Field to refine the result.
<table>
<thead>
<tr>
<th>Scene</th>
<th>Our(h)</th>
<th>Our(v)</th>
<th>Our(f)</th>
<th>Multiview</th>
<th>EPIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buddha</td>
<td>0.579</td>
<td>0.544</td>
<td>0.391</td>
<td>0.512</td>
<td>0.57</td>
</tr>
<tr>
<td>Buddha2</td>
<td>2.099</td>
<td>2.044</td>
<td>1.345</td>
<td>1.092</td>
<td>0.87</td>
</tr>
<tr>
<td>Horse</td>
<td>2.497</td>
<td>3.732</td>
<td>1.723</td>
<td>1.656</td>
<td>2.12</td>
</tr>
<tr>
<td>Medieval</td>
<td>1.768</td>
<td>1.948</td>
<td>0.901</td>
<td>0.845</td>
<td>1.15</td>
</tr>
<tr>
<td>MonasRoom</td>
<td>1.213</td>
<td>0.760</td>
<td>0.572</td>
<td>0.650</td>
<td>0.90</td>
</tr>
<tr>
<td>Papillon</td>
<td>1.901</td>
<td>2.259</td>
<td>1.592</td>
<td>2.105</td>
<td>2.26</td>
</tr>
<tr>
<td>StillLife</td>
<td>4.004</td>
<td>2.669</td>
<td>2.277</td>
<td>3.500</td>
<td>3.06</td>
</tr>
<tr>
<td>Couple</td>
<td>0.557</td>
<td>0.319</td>
<td>0.364</td>
<td>0.316</td>
<td>0.18</td>
</tr>
<tr>
<td>Cube</td>
<td>0.750</td>
<td>0.639</td>
<td>0.542</td>
<td>0.531</td>
<td>0.85</td>
</tr>
<tr>
<td>Maria</td>
<td>0.115</td>
<td>0.302</td>
<td>0.107</td>
<td>0.156</td>
<td>0.10</td>
</tr>
<tr>
<td>Pyramide</td>
<td>0.758</td>
<td>0.633</td>
<td>0.449</td>
<td>0.467</td>
<td>0.38</td>
</tr>
<tr>
<td>Statue</td>
<td>1.283</td>
<td>1.312</td>
<td>0.333</td>
<td>0.310</td>
<td>0.33</td>
</tr>
<tr>
<td>Average</td>
<td>1.460</td>
<td>1.430</td>
<td>0.883</td>
<td>1.012</td>
<td>1.07</td>
</tr>
</tbody>
</table>

Table 4.4: MSE Error table for HCI light field datasets. All values should be multiplied by 0.001. From the second to fifth column, the results are estimated by brute-force search in horizontal 2D light field slices, brute-force search in vertical 2D light field slices, fusing horizontal and vertical results, multiple-view approach and structure tensor approach, respectively. Units are in pixels/view.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Our(h)</th>
<th>Our(v)</th>
<th>Our(f)</th>
<th>Multiview</th>
<th>EPIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buddha</td>
<td>3.18%</td>
<td>2.86%</td>
<td>1.98%</td>
<td>2.48%</td>
<td>3.82%</td>
</tr>
<tr>
<td>Buddha2</td>
<td>23.7%</td>
<td>24.4%</td>
<td>14.9%</td>
<td>11.2%</td>
<td>11.2%</td>
</tr>
<tr>
<td>Horse</td>
<td>15.4%</td>
<td>15.5%</td>
<td>11.4%</td>
<td>10.4%</td>
<td>19.6%</td>
</tr>
<tr>
<td>Medieval</td>
<td>18.4%</td>
<td>21.1%</td>
<td>9.93%</td>
<td>9.08%</td>
<td>13.8%</td>
</tr>
<tr>
<td>MonasRoom</td>
<td>8.42%</td>
<td>5.50%</td>
<td>4.62%</td>
<td>5.39%</td>
<td>6.65%</td>
</tr>
<tr>
<td>Papillon</td>
<td>19.3%</td>
<td>20.8%</td>
<td>14.9%</td>
<td>13.0%</td>
<td>26.6%</td>
</tr>
<tr>
<td>StillLife</td>
<td>9.03%</td>
<td>6.91%</td>
<td>5.85%</td>
<td>6.07%</td>
<td>16.1%</td>
</tr>
<tr>
<td>Couple</td>
<td>12.3%</td>
<td>2.92%</td>
<td>2.42%</td>
<td>2.77%</td>
<td>8.17%</td>
</tr>
<tr>
<td>Cube</td>
<td>4.24%</td>
<td>5.27%</td>
<td>3.56%</td>
<td>3.24%</td>
<td>4.22%</td>
</tr>
<tr>
<td>Maria</td>
<td>0.85%</td>
<td>5.92%</td>
<td>0.52%</td>
<td>0.50%</td>
<td>2.98%</td>
</tr>
<tr>
<td>Pyramide</td>
<td>13.5%</td>
<td>7.46%</td>
<td>3.10%</td>
<td>3.23%</td>
<td>7.25%</td>
</tr>
<tr>
<td>Statue</td>
<td>14.9%</td>
<td>36.6%</td>
<td>7.05%</td>
<td>6.32%</td>
<td>13.4%</td>
</tr>
</tbody>
</table>

Table 4.5: Percentage Error table for HCI light field datasets. The figures are the percentage of pixels that are larger than 0.1% deviation from ground truth. From the second to fifth column, the results are estimated by brute-force search in horizontal 2D light field slices, brute-force search in vertical 2D light field slices, fusing horizontal and vertical results, multiple-view approach and structure tensor approach, respectively.
Figure 4.24: Colour coded ray sample patterns in a normalized coordinates. Different rays are sampled from 4D light field with two-plane parameterization (UV and XY plane). The first row is full view sampling (multiple-view stereo). The second row is vertical views sampling. The third row is horizontal views sampling. The second to the fourth column shows the XY plane sampling pattern at different slope value $k$ (depth).

<table>
<thead>
<tr>
<th>Scene</th>
<th>Focal Length</th>
<th>Baseline</th>
<th>Shfit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buddha</td>
<td>9.3 pixels</td>
<td>48 mm</td>
<td>24 pixels</td>
</tr>
<tr>
<td>Horse</td>
<td>9.3 pixels</td>
<td>51 mm</td>
<td>24 pixels</td>
</tr>
<tr>
<td>MonasRoom</td>
<td>9.3 pixels</td>
<td>38 mm</td>
<td>18 pixels</td>
</tr>
<tr>
<td>Maria</td>
<td>4152 pixels</td>
<td>2 mm</td>
<td>10 pixels</td>
</tr>
<tr>
<td>Cube</td>
<td>2595 pixels</td>
<td>2 mm</td>
<td>10 pixels</td>
</tr>
<tr>
<td>Statue</td>
<td>2595 pixels</td>
<td>2 mm</td>
<td>10 pixels</td>
</tr>
</tbody>
</table>

Table 4.6: Optical configurations for the scenes shown in Fig 4.25
Figure 4.25: Visualization of our results on the HCI dataset. The ground truth, our result and error map are colour coded. The left scale bar shows the absolute depth range and the right scale bar shows the absolute depth error range. All units are in mm.
4.4.3  *Lytro* Light Field Dataset Results and Discussion

We also evaluate our algorithm on the *Lytro* camera raw files with both indoor and outdoor scenes. Extracting depth from *Lytro* is challenging mainly due to the fact that quality of the reconstructed 4D light field is limited by the total resolution of the image sensor. However, with our fusion and refinement schemes, the quality of the disparity map can be improved. According to our depth error analysis in Section 4.2, ideally, the reliable slope range can be determined for given optical settings. However, the exact optical settings for *Lytro* camera is unknown to us. Therefore, we constrain the range of slope $k$ from -2 to 2 pixels/view regardless of the zoom and focus settings of the captured *Lytro* raw image. Actually, we deliberately choose very similar optical settings with the *Lytro* camera in the relative error example demonstrated in Section 4.2. It turns out that -2 pixels/view is a safe and reliable lower slope boundary for a wide range of focal lengths. For the positive slope boundary, the maximum reliable slope value can be as large as 5 pixels/view, depending on how close the object is to the camera. However, a narrow slope range is preferred to reduce the computational burden. Empirically, we choose 1 pixels/view as the upper bound and it results in a good estimation for our datasets.

To quantitatively analyse our results is difficult as the ground truth is not available. Creating a MLA-based light field dataset for quantitatively evaluating the depth from light field algorithms is our future work (See discussion in section 5.2 Future work). Instead, we compare our results with the state-of-the-art algorithms and the commercial software provided by *Lytro*. As shown in Fig.4.26 and Fig.4.27 our results are comparable to their results. In most of the cases, our results looks sharper and more realistic. The edges of the objects are well preserved rather than smoothed, thanks to our effective refinement scheme. Since our implementation is simple, the computational time is far less than Tao et al. [119] and Lin et al. [123]. It requires less than 10 seconds for a disparity map with 640 by 640 resolution and roughly 1 second for a disparity map with 320 by 320 resolution. In [119, 123], they reported more than a couple of minutes processing time to generate a disparity map of size 320 by 320.
Figure 4.26: Comparison on indoor Lytro dataset. First column is the central view. Second column shows the results from Lin et al. [123] or Lytro commercial software. The third column shows our results. The colour bar shows the slope range of which units are in pixels/view.
Figure 4.27: Comparison on outdoor Lytro dataset. First column is the central view. Second column shows the results from Lin et al. [123] or Lytro commercial software. The third column shows our results. The colour bar shows the slope range of which units are in pixels/view.
In this thesis, we present a study of 4D light field reconstruction, calibration, and depth estimation from plenoptic camera raw images. In Chapter 2, a computational camera simulation framework is developed for the purpose of evaluating an MLA-based light-field camera design from both optical design and image processing perspectives. In Chapter 3, a geometric and radiometric calibration pipeline for 4D light field reconstruction is introduced. In Chapter 4, we discuss the depth resolvability of an MLA-based light-field camera and propose an effective depth estimation algorithm.

5.1 Summary of Thesis Work

Light-Field Camera Simulation  We develop a framework for the simulation of computational cameras. It provides an open simulation framework with minimum restrictions for computational imaging researchers, which allows us to jointly consider the design from both optics and image processing perspectives. Although there are similar commercially available software packages, our work provides the detailed implementation which reveals how to use the abstract optic concepts to simulate an optical system. In the application section, we demonstrate three examples using our simulation framework. The first example shows that our implementation can produce almost identical results to the standard optics design software, Zemax in finding the on-axis optimal focus (the least circle of confusion) of a Double-Gaussian lens. In the
second example, we visualize the 2D light field transport including light field propagation, refraction, and projection, inside an MLA-based light-field camera using our simulation framework. In the last example, we analyse the refocusing performance of an MLA-based light-field camera. Our results show that using digital refocusing the spatial resolvability is improved by over a factor of 2 on average, whereas the spatial resolvability is dropped by a factor of 6.7 in the focused case, due to the spatio-angular trade-off.

**Light-Field Camera Calibration** We propose a light field calibration and reconstruction pipeline. For the 4D light field reconstruction, we demonstrate that it is feasible to use global grid to localize the centres of the microlenses as the mean residual error between the measured grids and global grids is less than 0.3 pixels. We use homography transformation to model the relationship between the ideal and measured grids. To reconstruct the 4D light field without using a reference image, we propose a blind global grid fitting approach which is based on a novel centroid estimator, detecting the dark gaps between microlens images. Our results are comparable to state-of-the-art light field reconstruction algorithms, which all require a reference image to decode the light field. Our quantitative analysis shows the image quality, the SSEQ score (See Table 3.2) of our approach is comparable to the scores of Cho et al. \[95\] and Dan et al. \[94\] approaches which all require a calibration image. For the 4D light field radiometric calibration, we use local and global fitting functions to jointly model the image irradiance variations across the plenoptic raw images. Although our results are not as good as the direct flat-field correction approach, we demonstrate the possibility of removing the vignetting using modeling.

**Depth From Light Field** We present our theoretical analysis for the depth resolvability of the MLA-based light-field camera. We show that the depth resolvability drops dramatically when the slope in the 2D light field slice is less than -2 pixels/view. This is because the equivalent baseline of an MLA-based light-field camera is constrained by the size of the aperture stop of the main lens. We also present a simple and efficient implementation of depth from light field algorithm, and in particular an effective refinement scheme for the noisy raw data from the Lytro camera. The experimental results show that our algorithm outperforms the state-of-the-art methods and commercial software. In the HCI dataset, the average MSE over 12 scenes of our approach is 0.883 pixels/view which is smaller than multiple-view (1.012 pixels/view) and structure tensor (1.07 pixels/view) approaches. In the Lytro dataset, our pro-
posed approach consistently produces sharper and more realistic results compared to the state-of-the-art approaches \cite{123} and Lytro commercial software.

\section*{5.2 Future Work}

Within the scope of the topics covered in this thesis, there are some points which could be investigated as further work.

- Our light-field camera simulation framework can be extended to a more generic framework that supports different kinds of computational cameras such as lattice-focal camera and multi-aperture camera. The hope is to allow computational imaging researchers to quickly evaluate their ideas without spending too much time and buying expensive instruments.

- In our light field reconstruction, the global parameters are derived from an affine transformation. Globally, this yields plausible results. However, locally, particularly at the boundary of the raw image, the results are not as good as the centre region due to the vignetting effect. Additional local processing would further improve the quality of the reconstructed light field. A more advanced solution might be to use an ultra-definition LCD screen as a dynamically controlled reference to calibrate the MLA-based light-field camera pixel by pixel.

- A good quality light field provides the foundation to extract depth information. However, the optical aberrations, which every optical system exhibits, not only degrade the image quality but also degrade the quality of depth estimation. For example, the chromatic aberrations result in depth variations among colour channels, the field curvature aberrations result in depth variations across the central view image. An optical aberration correction process is necessary for recovering a higher quality depth map from light field.

Currently, there is no ground truth dataset for quantitatively evaluating the depth from light field algorithms \cite{119,122,123} for MLA-based light-field cameras. It is of great interest to create such a dataset similar to Middlebury \cite{133} or KITTI \cite{134} for evaluating depth from stereo algorithms. The depth of the scene can be acquired using structure light (a conventional camera plus a 2D projector) approach \cite{135}. The key question will be how to accurately estimate relative pose between the MLA-based light field camera and the conventional camera.
In the broad sense, one key question should be always be kept in mind; what substantial benefit can an MLA-based light-field camera offer to the end user? The concept of capturing the 4D light field with an MLA-based light-field camera has been well acknowledged by the research community, regarding it as a milestone in the evolutionary process of photography. Some experts even predict that every camera will have the capability to record the light field in the future. However, the MLA-based light-field camera has not become a popular product in the consumer market yet. From the user’s side, the reason is simple; there is no substantial benefit to using an MLA-based light-field camera instead of a conventional camera. For example, digital refocusing can also be realized by a stereo camera setup without losing spatial resolution. The real secret is that the 4D light field contains too much redundant data when it spatially multiplexes angular and spatial information. For example, less than 20% (only the cross-hair views) of the light field data are used for extracting the depth. It is still a task for researchers to develop novel applications that efficiently use the 4D light field data. On the other hand, an MLA-based light-field camera can be modified to multiplex other information into the 4D data space such as spectrum or polarization. This is another promising direction to efficiently use the 4D light field data and offer unique functionalities compared to other computational imagers.
Appendices
## Appendix A : List of Raw Images

In Tab.1 we list all the Lytro raw images that are used in the thesis.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Exposure</th>
<th>F</th>
<th>Camera ID</th>
<th>Page</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform1</td>
<td>0.004</td>
<td>51.3</td>
<td>A202200390</td>
<td>58-60,62,78</td>
<td>Us</td>
</tr>
<tr>
<td>Uniform2</td>
<td>0.004</td>
<td>7.80</td>
<td>A202200390</td>
<td>58-60,</td>
<td>Us</td>
</tr>
<tr>
<td>Office</td>
<td>0.016</td>
<td>6.45</td>
<td>A202200390</td>
<td>29,64,69-70,72,115</td>
<td>Us</td>
</tr>
<tr>
<td>Toy map</td>
<td>0.004</td>
<td>6.58</td>
<td>A202200390</td>
<td>69-70,73,87,104,115</td>
<td>Us</td>
</tr>
<tr>
<td>Flower4</td>
<td>0.005</td>
<td>7.39</td>
<td>A202200390</td>
<td>69-70</td>
<td>Us</td>
</tr>
<tr>
<td>Cat</td>
<td>0.040</td>
<td>16.14</td>
<td>A202200390</td>
<td>69-70,90</td>
<td>Us</td>
</tr>
<tr>
<td>Campus</td>
<td>0.004</td>
<td>12.55</td>
<td>A202200390</td>
<td>72</td>
<td>Us</td>
</tr>
<tr>
<td>ISO chart</td>
<td>0.004</td>
<td>12.95</td>
<td>A202380752</td>
<td>69-70,72-73,104</td>
<td>Us</td>
</tr>
<tr>
<td>Flower1</td>
<td>0.007</td>
<td>7.27</td>
<td>A102020186</td>
<td>116</td>
<td>Tao et al. [119]</td>
</tr>
<tr>
<td>Flower3</td>
<td>0.008</td>
<td>6.99</td>
<td>A102020186</td>
<td>102,116</td>
<td>Tao et al. [119]</td>
</tr>
<tr>
<td>Guitar</td>
<td>0.017</td>
<td>6.46</td>
<td>A102020186</td>
<td>115</td>
<td>Tao et al. [119]</td>
</tr>
<tr>
<td>Ironwire</td>
<td>0.011</td>
<td>6.48</td>
<td>A202260222</td>
<td>116</td>
<td>Lin et al. [123]</td>
</tr>
<tr>
<td>Chicken</td>
<td>0.017</td>
<td>6.44</td>
<td>A202260222</td>
<td>116</td>
<td>Lin et al. [123]</td>
</tr>
<tr>
<td>Parrot</td>
<td>0.004</td>
<td>51.31</td>
<td>A102160215</td>
<td>69-70,72-73</td>
<td>Dan et al. [94]</td>
</tr>
<tr>
<td>Jumper</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>72-73</td>
<td>Cho et al. [95]</td>
</tr>
<tr>
<td>Squirrel</td>
<td>0.004</td>
<td>6.61</td>
<td>FGC3310013</td>
<td>116</td>
<td>[78]</td>
</tr>
</tbody>
</table>

Table 1: Lytro raw image list. The unit of exposure time is in seconds. F stands for the focal length of the zoom lens, which unit is in mm.
Appendix B : Glossary

COC  Circle of Confusion
DMD  Digital Mirror Device
DOF  Depth of Field
EPI  Epipolar Image
FFC  Flat Field Correction
FOV  Field-of-View
FPGA Field Programmable Gate Array
GPU  Graphical Processing Unit
HCI  Heidelberg Collaboratory for Image Processing
LCOS Liquid Crystal on Silicon
MAE  Mean Absolute Error
MSE  Mean Square Error
MLA  Microlens Array
MTF  Modular Transfer Function
PSF  Point Spread Function
PSNR Peak Signal to Noise Ratio
SNR  Signal to Noise Ratio
SSEQ Spatial-Spectral Entropy-based Quality
SSIM Structural Similarity
TOF  Time-of-Flight
RMSE Root Mean Square Error
Appendix C : Source Code

.1 Light-Field Camera Simulation

.1.1 Single Surface Raytracing

```cpp
bool Ray_Tracer::single_surface_refraction(Ray * outRay,
                                          const Ray * inRay,
                                          const double input_miu,
                                          lensElement l) {

    Point centre (0,0,(l.zpos – l.rad));
    Vector C = centre – inRay->o;
    double C02 = C.x*C.x+C.y*C.y+C.z*C.z;
    double CF = C.x*inRay->d.x+C.y*inRay->d.y+C.z*inRay->d.z;//Dot product
    double EF2 = (l.rad*l.rad) + CF*CF – C02;

    if(EF2 < 0.f)
        return false;
    double EF = sqrt(EF2);
    double t = (l.rad > 0.f) ? (CF – EF) : (CF + EF);
    Point Plens = inRay->o + t * inRay->d;

    if(((Plens.x*Plens.x + Plens.y*Plens.y) > (l.aper*l.aper/4))
        return false;

    Vector I, N, T;
    N = (l.rad > 0.f) ? Normalize(Plens – centre):
       Normalize(centre – Plens);
    I = inRay->d;
```
\section*{1.2 Forward Raytracing}

```cpp
double c1, c2;
c1 = -(I.x*N.x+I.y*N.y+I.z*N.z);
c2 = 1.f - (input_miu*input_miu)* (1.f - c1 * c1);
if(c2 < 0.f)
  return false;

c2 = sqrt(c2);
T = input_miu * I + (input_miu * c1 - c2) * N;
outRay->o = Plens;
outRay->d = Normalize(T);
outRay->group= inRay->group;
return true;
}
```

```cpp
bool Ray_Tracer::forward_tracing(Ray inRay, vector<Ray>* rr, double dist){
  vector<lensElement>::iterator Li, Lp;
  int index = 0;
  Ray outRay;
  double miu = -1;

  for (Li = lens_sys.lenses.begin(); Li < lens_sys.lenses.end(); Li++) {
    if (Li->isActive & & !Li->isStop) {
      Lp = Li - 1;
      while (Lp->isStop) Lp = Lp - 1;
      if (Lp < lens_sys.lenses.begin() ) {
        miu = 1.0f/Li->ndr;
      }
    }
    else
      miu = Lp->ndr / Li->ndr;
    bool b_penetrated = single_surface_refraction(&outRay, &inRay, miu,
       +Li);
    if ( !b_penetrated) {
      invalidateRay(&outRay);  
      return false;
    }
    if ( Li == lens_sys.lenses.begin() ){
      rr->push_back(inRay);
    }
    rr->push_back(outRay);
  }
```
26 index ++;
27 inRay = outRay;
28 if (Li == (lens_sys.lenses.end()−1)) |
29 outRay.o=outRay.o+((dist−outRay.o.z)/outRay.d.z)*outRay.d;
30 rr->push_back(outRay);
31 |
32 |
33 else if (Li->isActive && Li->isStop ) |
34 bool b_penetrated = check_stop(&outRay, &inRay,*Li);
35 if (!b_penetrated) |
36 invalidateRay(&outRay);
37 return false;
38 |
39 |
40 } push_back(outRay);
41 index ++;
42 inRay = outRay;
43 |
44 |
45 return true;
46 }

.1.3 Finding Gaussian Property

1 void lensSystem::calF1F2P1P2(int start,
2 int end,
3 float& ff1,
4 float& ff2,
5 float& pp1,
6 float& pp2) {  
7 int size=lensSurfaces.size();
8 float u[size],h[size−1],n[size],d[size−1],c[size−1];
9 for(int i = 0; i <= size−1; i++) {
10 d[i] = lensSurfaces[size−i−1].thick;
11 c[i] =(lensSurfaces[size−i−1].isStop)? 0: 1/lensSurfaces[size−i−1].←
12 radius;
13 n[i+1]=(lensSurfaces[size−i−1].isStop)? 1: lensSurfaces[size−i−1].←
14 index;
15 }
16 u[start]=0.f;
17 h[start]=1;
18 n[start]=1.f;


```c
for (int i=start; i<=end; i++)
    float K=(n[i+1]-n[i])*(c[i]);
    u[i+1]= (n[i+1]<0.01)? u[i]:(-h[i]* K + n[i]*u[i] ) / n[i+1];
    if (i<end) h[i+1]= h[i]+d[i] * u[i+1];
}

ff2 = h[end]/u[end+1]+lensSurfaces[size-end].posZ ;
pp2 = ff2 − h[start]/u[end+1];
u[end+1]=0.f;
h[end] =1;
n[end+1]=1.f;

for (int i=end; i>=start; i--){
    float K=(n[i]-n[i+1])*(-c[i]);
    u[i]= (n[i]<0.01)? u[i+1]:(-h[i]* K + n[i+1]*u[i+1] ) / n[i];
    if(i>start) h[i-1]= h[i]+d[i-1] * u[i];
}

ff1 =−h[start]/(u[start])+lensSurfaces[size−start].posZ;
pp1 = ff1 + h[end]/(u[start]);
```

### 1.4 Concentric Sampling

```c
Point todisk(Point s, double l){
    double phi,r;
    double a=2*s.x−l;
    double b=2*s.y−l;
    if(a*a>b*b){
        r=a;
        phi=(CV_PI/4.0f)*(b/a);
    } else{
        r=b;
        phi=(CV_PI/2.0f)−(CV_PI/4.0f)*(a/b);
    }
    Point u;
    u.x=l*r*cos(phi);
    u.y=l*r*sin(phi);
    u.z=s.z;
```
u.w=1;
    return u;
}

bool gen_ray_pt_4d(Point pt,
    double radius,
    double z_axis,
    int num,
    vector<Ray>& rs,
    int id){
    double inc = 1/(double) num; //num should be odd
    radius = radius;
    for (int j=0; j<num;j++)
        double y=(double)j*inc;
        for (int i=0; i<num;i++)
            double x=(double)i*inc;
            Point ex= todisk(Point(x,y,z_axis), radius);
            Vector r = ex – pt;
            Ray l;
            l.o = pt;
            l.d = Normalize(r);
            l.group=id;
            l.w= 1;
            l.ex_pos = ex;
            rs.push_back(l);
    }
    return true;
}

1.5 Microlens-Array Raytracing

void microlens_array::microlens_refraction(vector<Ray> *r_chain,
    float iz){
    this->pos=iz;
    r_chain->pop_back();
    Ray r=r_chain->back();
    r.o = r.o + ((iz-r.o.z)/r.d.z)*r.d;
    int sign_x,sign_y;
    if (r.o.x >0)
        sign_x=1;
    else sign_x=-1;
    if (r.o.y >0)
        sign_y=1;
    else sign_y=-1;
else sign_y = -1;
int idx = int((r.o.x + sign_x * this->r) + 0.5 / this->r);
int idy = int((r.o.y + sign_y * this->r) + 0.5 / this->r);
float shift_exit_pupil_x = idx * 2 * this->r;
float shift_exit_pupil_y = idy * 2 * this->r;
if ((abs(idx) <= this->w) && (abs(idy) <= this->h)) {
    float abs_x = r.o.x - (shift_exit_pupil_x);
    float abs_y = r.o.y - (shift_exit_pupil_y);
    if ((abs_x * abs_x + abs_y * abs_y) < (this->r) * (this->r)) {
        r.d.x = -(r.d.x) / (r.d.z);
        r.d.y = -(r.d.y) / (r.d.z);
        r.d.z = -1;
        r.d = Normalize(r.d);
        r_chain->push_back(r);
        r.o = r.o + ((-this->z) / r.d.z) * r.d;
        r_chain->push_back(r);
    }
}

.1.6 Digital Refocusing

void LightField_PE::Image_Rendering(gen_msg *lf_msg, float *im2d_ptr) {
  float scale = (2 + lf_format->lens_radius / lf_format->lf_angular_n) / 
               lf_format->dist_fplane;
  int half_xy = (lf_format->lf_spatial_n - 1) / 2;
  int half_uv = (lf_format->lf_angular_n - 1) / 2;
  int img_w = lf_format->lf_spatial_n;
  int view_n = lf_format->lf_angular_n;
  float alpha_inv = 1 / alpha;
  #pragma omp parallel for
  for (int j = 0; j < img_w; j++) {
    for (int i = 0; i < img_w; i++) {
      for (int v = 0; v < view_n; v++) {
        float vv = (v - half_uv) * scale;
        float y = j + (vv * (alpha - 1));
        int idy = int(y) * 3 * img_w;
        if ((y < (img_w - 2)) && (y >= 0)) {
          for (int u = 0; u < view_n; u++) {
            ...
          }
        }
      }
    }
  }

\[ u = (u - \text{half}_u) \times \text{scale}; \]
\[ x = i + (u_{\alpha} - 1); \]
\[ \text{idx} = \text{id} + \text{int}(x) \times 3; \]
\[ \text{if } ((x < (\text{img}_w - 2)) && (x > 0)) \{
\]
\[ \text{float } sy = y - \text{int}(y); \]
\[ \text{float } sz = x - \text{int}(x); \]
\[ \text{float } sxy = sz \times sy; \]
\[ \text{float } s1xy = (1 - sz) \times sy; \]
\[ \text{float } sx1y = (sz) \times (1 - sy); \]
\[ \text{float } s1x1y = (1 - sz) \times (1 - sy); \]
\[ \text{for } (\text{int } k = 0; k < 3; k++) \{
\]
\[ \text{float } \text{raw}_\text{data} = \text{lf}_1[j][i][v][u][k]; \]
\[ \text{im2d}_\text{ptr}[\text{idx} + k] += s1x1y \times \text{raw}_\text{data}; \]
\[ \text{im2d}_\text{ptr}[\text{idx} + k + 1] += sx1y \times \text{raw}_\text{data}; \]
\[ \text{im2d}_\text{ptr}[\text{idx} + k + 3 \times \text{img}_w] += s1xy \times \text{raw}_\text{data}; \]
\[ \text{im2d}_\text{ptr}[\text{idx} + k + 3 \times \text{img}_w + 1] += sxy \times \text{raw}_\text{data}; \]
\[ \} \]
\[ \} \]
\[ \} \]
\[ \} \]
\[ \}

.2 Light-Field Camera Calibration

.2.1 Blind Global Grid Fitting

bool gen_score_map(Mat img, Mat& score, PARA* para, float D) {
    double idy[6] = {1.0f/3.0f, -1.0f/3.0f, 1.0f/6.0f, 1.0f/6.0f, -1.0f/6.0f};
    double idx[6] = {0, 0, 0.5, -0.5, 0.5, -0.5};
    score = Mat(img.size(), CV_32FC1);
    for(int j = 0; j < img.rows; j++) {
        for(int i = 0; i < img.cols; i++) {
            if (((j > 4*D) && (j < (img.rows - 4*D))) && (i < (img.cols - 4*D))) {
                double avg = 0;
                double v;
                double vmax = 0;
            }
        }
    }
}
double vmin = 10000000;
for (int t=0;t<6;t++)
    Point pt = Point(round(i+4*D*idx[t]),round(j+4*D*sqrt(3)*idy[t]));
    v=(double)img.at<float>(pt);
    avg=avg+v;
    if (v>vmax)
        vmax=v;
    if (v<vmin)
        vmin=v;
}
    score.at<float>(j,i)=(avg-vmax-vmin)/4;
}
}
return true;
}
bool brute_force_search_2d (int* itn,
    double* step,
    double* tran_vec,
    vector<Point2d> pos,
    const Mat& img){
double tran_vec_temp[6];
double tran_vec_optimum[6]={0,0,0,0,0,0};
save_para(tran_vec,tran_vec_temp);
double current=100000;
Mat score_map = Mat::zeros(2*itn[2]+1,2*itn[5]+1,CV_32F);
for(int g=-itn[2];g<(itn[2]+1);g++)
    for(int h=-itn[5];h<(itn[5]+1);h++)
        double next=0;
        vector<Point2d> pos_2d;
        pos_2d.resize(pos.size());
        for(size_t kk=0;kk<pos.size();kk++)
            pos_2d[kk].x = pos[kk].x*tran_vec_temp[0]
                +pos[kk].y*tran_vec_temp[1]+tran_vec_temp[2];
            pos_2d[kk].y = pos[kk].x*tran_vec_temp[3]
            Point p=Point(round(pos_2d[kk].x),round(pos_2d[kk].y));
            next=next+img.at<float>(p);
        }
        next=next/pos.size();
        score_map.at<float>(g+itn[2],h+itn[5])=next;
        if (next<current)
            current=next;
save_para(tran_vec_temp, tran_vec_optimum);

vector<Point2d>().swap(pos_2d);
}

bool brute_force_search_theta (float & theta, 
    float & scale, 
    double * tran_vec, 
    vector<Point2d> pos, 
    vector<Point2d>& pos2, 
    Mat img, 
    Mat img2){

double current=100000;
Mat score_map = Mat::zeros(2*100+1,2*100+1,CV_32F);
for(int m=-100;m<=100;m++){
    for(int n=-100;n<=100;n++){
        double next=0;
        pos_2d.resize(pos.size());
        float thetan=n/10000.0f;
        float scalen=1+m/10000.0f;
        for(size_t kk=0;kk<pos.size();kk++){
            pos_2d[kk].x = -3280*4/2 + (pos[kk].x*tran_vec[0]+tran_vec[2]);
            pos_2d[kk].x = pos_2d[kk].x * cos(thetan) - pos_2d[kk].y * sin(thetan);
            pos_2d[kk].y = pos_2d[kk].x * sin(thetan) + pos_2d[kk].y * cos(thetan);
            pos_2d[kk].x = 3280*4/2 + scalen*pos_2d[kk].x ;
            pos_2d[kk].y = 3280*4/2 - scalen*pos_2d[kk].y ;
            Point p=Point(round(pos_2d[kk].x),round(pos_2d[kk].y));
            next=next+img.at<float>(p);
        }
        next=next/pos.size();
        score_map.at<float>((m+100,n+100)=next;
        if (next<current){
            current=next;
            theta = thetan;
            scale = scalen;
        }
    }
}
vector<Point2d>().swap(pos_2d);
pos2.resize(pos.size());
for(size_t kk=0;kk<pos.size();kk++){
pos2[kk].x = -3280*4/2 + (pos[kk].x*tran_vec[0]+tran_vec[2]);
pos2[kk].x = pos2[kk].x * cos(theta) - pos2[kk].y * sin(theta);
pos2[kk].y = pos2[kk].x * sin(theta) + pos2[kk].y * cos(theta);
pos2[kk].x = 3280*4/2 + scale*pos2[kk].x;
pos2[kk].y = 3280*4/2 - scale*pos2[kk].y;
Point p=Point(round(pos2[kk].x),round(pos2[kk].y));
}
return true;
}

bool grid_optimization(Mat& img, Mat img2, Mat& H, vector<Point2d>& pos_vec,
vector<Point2d>& pos_vec2, double* tran_mat_vec, PARA* para){
  int itn[6] = {100, 100, 200, 100, 100, 200};
  double step[6] = {0.002, 0.002, 0.1, 0.002, 0.002, 0.1};
  cout<<"Optimization Step 1..."<<endl;
gen_pos(pos_vec, Rect(para->ML_W*3/8,
    para->ML_H*3/8,
    para->ML_W/4,
    para->ML_H/4),
    para, 4, 2);
  tran_mat_vec[2]=4.0f*tran_mat_vec[2];
  tran_mat_vec[5]=4.0f*tran_mat_vec[5];
brute_force_search_2d(itn,step,tran_mat_vec,pos_vec,img);
  vector<Point2d>().swap(pos_vec);
  cout<<"Optimization Step 2..."<<endl;
gen_pos(pos_vec, Rect(0,0,para->ML_W,para->ML_H), para, 4, 4);
  float theta=0;
  float scale=0;
brute_force_search_theta(theta, scale, tran_mat_vec, pos_vec, pos_vec2,
    img, img2);
  vector<Point2d>().swap(pos_vec);
  vector<Point2d>().swap(pos_vec2);
gen_pos(pos_vec, Rect(0,0,para->ML_W,para->ML_H), para, 4, 1);
gen_pos2(pos_vec, pos_vec2, tran_mat_vec, theta, scale, para);
2.2 Vignetting Correction

```cpp
bool vig_correction_modeling_raw(Mat raw16,
    Mat& img_c,
    vector<Point2f>& pos,
    PARA* para)
{
    Mat raw_mat;
    raw16.convertTo(raw_mat, CV_32FC1);

    for (size_t j=0; j<raw_mat.rows; j++)
        for (size_t i=0; i<raw_mat.cols; i++)
            raw_mat.at<float>(j, i) = raw_mat.at<float>(j, i) - 168;

    mat2hdf5 ("./debug/vig/raw.h5","data", H5T_NATIVE_FLOAT, float(), raw_mat);

    Array_Vig* array_vig = new Array_Vig(para->ML_W*para->ML_H);
    (*array_vig).resize(para->ML_H);
    for (int j = 0; j < para->ML_H; j++)
        (*array_vig)[j].resize(para->ML_W);
}
```
local_vig_estimator(raw_mat, pos, array_vig, para); // local fitting

double* coie_data = new double[45];
global_vig_estimator(array_vig, pos, coie_data, para);

Mat m_mat (para->RAW_H, para->RAW_W, CV_32FC1); // 4D multiview colour ← image.

Mat s1(1, 10, CV_64F);
Mat s2(1, 10, CV_64F);
Mat rr(1, 15, CV_64F);
Mat mm(1, 10, CV_64F);
Mat t1(10, 1, CV_64F);
Mat t2(15, 1, CV_64F);

for (int i = 0; i < 15; i++){
    rr.at<double>(0, i) = coie_data[i];
}

for (int i = 0; i < 10; i++){
    s1.at<double>(0, i) = coie_data[15 + i];
    s2.at<double>(0, i) = coie_data[25 + i];
    mm.at<double>(0, i) = coie_data[35 + i];
}

for (int k = 0; k < pos.size(); k++){
    Point2d centre = pos[k];
    Point ccc = Point(round(pos[k].x), round(pos[k].y));
    double x = (double)((centre.x) - (para->RAW_W - 1) / 2) / ((para->RAW_W - 1) / 2);
    double y = (double)((centre.y) - (para->RAW_H - 1) / 2) / ((para->RAW_H - 1) / 2);
    t1.at<double>(9, 0) = x * x * x * x * x * x * x;
    t1.at<double>(8, 0) = x * x * x * x * y * y;
    t1.at<double>(7, 0) = x * x * y * y * y * y;
    t1.at<double>(6, 0) = y * y * y * y * y * y;
    t1.at<double>(5, 0) = x * x * x * x;
    t1.at<double>(4, 0) = x * x * y * y;
    t1.at<double>(3, 0) = y * y * y * y;
    t1.at<double>(2, 0) = x * x;
    t1.at<double>(1, 0) = y * y;
    t1.at<double>(0, 0) = 1;

    t2.at<double>(14, 0) = x * x * x * x * x;
    t2.at<double>(13, 0) = x * x * x * y;
    t2.at<double>(12, 0) = x * x * y * y;
    t2.at<double>(11, 0) = x * y * y * y;
    t2.at<double>(10, 0) = y * y * y * y;
\[
\begin{align*}
t_2.\text{at<double>}(9, 0) &= x * x * x; \\
t_2.\text{at<double>}(8, 0) &= x * x * y; \\
t_2.\text{at<double>}(7, 0) &= x * y * y; \\
t_2.\text{at<double>}(6, 0) &= y * y * y; \\
t_2.\text{at<double>}(5, 0) &= x * x; \\
t_2.\text{at<double>}(4, 0) &= x * y; \\
t_2.\text{at<double>}(3, 0) &= y * y; \\
t_2.\text{at<double>}(2, 0) &= x; \\
t_2.\text{at<double>}(1, 0) &= y; \\
t_2.\text{at<double>}(0, 0) &= 1; \\
\end{align*}
\]

Mat a = (s1 * t1);
Mat b = (s2 * t1);
Mat c = (r * t2);
Mat d = (m * t1);
float ss1 = a.at<double>(0, 0);
float ss2 = b.at<double>(0, 0);
float rrr = c.at<double>(0, 0);
float mmm = d.at<double>(0, 0);

for (int jj = -4; jj < 5; jj++)
  for (int ii = -4; ii < 5; ii++)
    double offset_x = ii + round(pos[k].x) - pos[k].x;
    double offset_y = jj + round(pos[k].y) - pos[k].y;
    double aa = ss1 * ss2 * sqrt(1 - rrr * rrr);
    double bb = ss2 * ss2 * (offset_x) * (offset_x)
               - 2 * ss1 * ss2 * rrr * (offset_x) * (offset_y)
               + ss1 * ss1 * (offset_y) * (offset_y);
    m_mat.at<float>(ccc.y + jj, ccc.x + ii)
      = m_mat.at<float>(ccc.y + jj, ccc.x + ii) * 4095 + 168;

for (int jj = -4; jj < 5; jj++)
  for (int ii = -4; ii < 5; ii++)
    double offset_x = ii + round(pos[k].x) - pos[k].x;
    double offset_y = jj + round(pos[k].y) - pos[k].y;
    double aa = ss1 * ss2 * sqrt(1 - rrr * rrr);
    double bb = ss2 * ss2 * (offset_x) * (offset_x)
               - 2 * ss1 * ss2 * rrr * (offset_x) * (offset_y)
               + ss1 * ss1 * (offset_y) * (offset_y);
    m_mat.at<float>(ccc.y + jj, ccc.x + ii)
      = m_mat.at<float>(ccc.y + jj, ccc.x + ii) * 4095 + 168;

img_c = m_mat.clone();
img_c.convertTo(img_c, CV_16U);
return true;
bool disparity_cost( const Mat& img, float *depth,
    float *depthc, LF* lf_ptr)
{
    float data_new[3];
    Vec3f *data_ptr = (Vec3f*)(img.data);
    for (int i=3;i<(img.cols-3); i++){
        Vec3f data = img.at<Vec3f>((lf_ptr->U-1)/2,i);//Vec3b
        float* depth_addr = depth + i*lf_ptr->nlabels;
        float* depthc_addr = depthc + i*lf_ptr->nlabels;

        for (int k=0; k<lf_ptr->nlabels; k++)
            float err0=0;
        float avg =0;
        for (int t=-3; t<=3; t++)
            if (abs(t)>0)
                float xnew = (i + d[k] * float(t));
            float idy =((lf_ptr->U-1)/2+t);
            int idx = int(idy)*img.cols+int(xnew);
            float a,b;
            a = 1-(xnew- int(xnew));
            b = 1 -a;
            data_new[0] = data_ptr[idx][0]*a + data_ptr[idx+1][0]*b ;
            data_new[1] = data_ptr[idx][1]*a + data_ptr[idx+1][1]*b ;
            data_new[2] = data_ptr[idx][2]*a + data_ptr[idx+1][2]*b ;
            float tmp[3];
            tmp[0] = data[0] -data_new[0];
            avg = avg + data_new[0] + data_new[1] + data_new[2];
            err0 += tmp[0]*tmp[0]+tmp[1]*tmp[1]+tmp[2]*tmp[2];
    }
    *depth_addr++ = err0;
    *depthc_addr++ = avg;
}
return true;
}

.3.2 Build The Cost Volume

void cost_volume( vector<Mat>& epi_h,
    vector<Mat>& epi_v,
    float* depth_x, float* depth_y,
    float* depthc_x, float* depthc_y,
    float* base_x, float* base_y, float* basec_x, float* basec_y)
{
float* depth_cx, float* depth_cy,
LF* lf_ptr) {

d = new float[lf_ptr->nlabels+1];
float dmin=lf_ptr->d_min;
float dmax=lf_ptr->d_max;
for (int k=0; k<=lf_ptr->nlabels; k++)
  d[k] = dmin+float(k)*(dmax-dmin)/float(lf_ptr->nlabels);
}
#pragma omp parallel for
for (int j = 0; j < lf_ptr->H; j++) {
  float *depth_array = (float*) calloc(lf_ptr->nlabels*lf_ptr->W, sizeof(float));
  float *con_array = (float*) calloc(lf_ptr->nlabels*lf_ptr->W, sizeof(float));

disparity_cost( epi_h[j], depth_array, con_array, lf_ptr);
memcpy(depth_x + lf_ptr->nlabels+j+lf_ptr->W, depth_array,
  lf_ptr->W+lf_ptr->nlabels * sizeof(float));
memcpy(depth_cx+ lf_ptr->nlabels+j+lf_ptr->W, con_array,
  lf_ptr->W+lf_ptr->nlabels * sizeof(float));
delete[] depth_array;
delete[] con_array;
}
#pragma omp parallel for
for (int i = 0; i < lf_ptr->W; i++) {
  float *depth_array = (float*) calloc(lf_ptr->nlabels*lf_ptr->W, sizeof(float));
  float *con_array = (float*) calloc(lf_ptr->nlabels*lf_ptr->W, sizeof(float));

disparity_cost( epi_v[i], depth_array, con_array, lf_ptr);
for (int j = 0; j < lf_ptr->H; j++){
  int idx = j+lf_ptr->W+i;
  memcpy(depth_y + lf_ptr->nlabels*(idx),
    depth_array+ j+lf_ptr->nlabels, lf_ptr->nlabels * sizeof(float));
  memcpy(depth_cy + lf_ptr->nlabels*(idx),
    con_array + j+lf_ptr->nlabels, lf_ptr->nlabels * sizeof(float));
}
delete[] depth_array;
delete[] con_array;
}
### 3.3 Build MRF

```c
bool lf2depth_mrf(float *depth_x,
                  float *depth_y,
                  float *confidence_x,
                  float *confidence_y,
                  uchar *depth_best_xy,
                  LF* lf_ptr)
{
    int width = lf_ptr->W;
    int height = lf_ptr->H;
    int num_pixels = width*height;
    int num_labels = lf_ptr->nlabels;
    float *cost = new float[num_pixels*num_labels];
    float* cost_filter = new float[num_pixels*num_labels];
    volume_merge(depth_x, depth_y, cost, lf_ptr);
    cost_filter = new float[num_pixels*num_labels];
    for (int l = 0; l < num_labels; l++)
        for (int j = 0; j < height; j++)
            for (int i = 0; i < width; i++)
                cost2[l*num_pixels+j*width+i] = cost[num_labels*(j*width+i)+l];
    for (int l = 0; l < num_labels; l++)
        for (int j = 0; j < height; j++)
            cost_filter[l*num_pixels+j*width+i] = cost[num_labels*(j*width+i)+l];
    cost_filtering(cost2, cost_filter, lf_ptr);
    int *smooth = new int[num_labels*num_labels];
    for (int l1 = 0; l1 < num_labels; l1++)
        for (int l2 = 0; l2 < num_labels; l2++)
            smooth[l1 + l2*num_labels] = abs(l1 - l2);
    try{
        GCoptimizationGeneralGraph *gc = new GCoptimizationGeneralGraph(num_pixels, num_labels);
        if (lf_ptr->lambda==0) lf_ptr->lambda=1;
        for (int j = 0; j < height; j++)
            for (int i = 0; i < width; i++)
                int idx = j*width+i;
                for (int k = 0; k < num_labels; k++)
                    if ( (j>4)&(i>4)&(j<(height-4))&(i<(width-4))
                        &&lf_ptr->disparity_mask.at<float>(j,i))
                        &&((confidence_x[idx]<lf_ptr->threshold)
                        &&(confidence_y[idx]<lf_ptr->threshold))
                        gc->setDataCost(j*width+i, k, 0);
    }
```
else
    gc->setDataCost(j*width+i, k, (lf_ptr->lambda *cost_filter[k*(height*width)+j*width+i]));
}
else
    gc->setDataCost(j*width+i, k, 0);
}

for (int y = 0; y < height; y++)
    for (int x = 1; x < width; x++){
        int p1 = x-1+y*width;
        int p2 = x+ y*width;
        int weight = confidence_x[p2]>lf_ptr->threshold ? 0 : 1;
        gc->setNeighbors(p1,p2,lf_ptr->disparity_mask.at<float>(y,x)←weight);
    }

for (int y = 1; y < height; y++)
    for (int x = 0; x < width; x++){
        int p1 = x+(y-1)*width;
        int p2 = x+y*width;
        int weight = confidence_y[p2]>lf_ptr->threshold ? 0 : 1;
        gc->setNeighbors(p1,p2,lf_ptr->disparity_mask.at<float>(y,x)←weight);
    }

gc->setSmoothCost(smooth);
gc->setVerbosity(1);
gc->expansion(1);
for (int j = 0; j<height; j++)
    for (int i = 0; i<width; i++)
        lf_ptr->depth.at<float>(j,i) = gc->whatLabel(j*width+i);
delete gc;
}
catch (GCException e){
e.e.Report();
}
delete [] cost;
delete [] cost2;
delete [] cost_filter;
delete [] smooth;

.3.4 Depth from Light Field

bool compute_slope_xy( float* data1, float* data2,
float* conf1, float* conf2,
float* cf1, float* cf2,
uchar* data_best,
LF* lf_ptr)
{
    int height = lf_ptr->H;
    int width = lf_ptr->W;
    int labels = lf_ptr->nlabels;
    int cnt=0;
    for (int j=0; j< height; j++)
        for (int i=0; i< width; i++)
            if (((j>0)&(i>0)&&(j<(height-1))&&(i<(width-1)))
                float score[2] = {0, 0};
                int idx[2] = {0, 0};
                depth_optimal_pixel(&data1[cnt*labels], labels, idx[0], score[0]);
                depth_optimal_pixel(&data2[cnt*labels], labels, idx[1], score[1]);
                int offsetx0 = cnt*labels + idx[0];
                int offsety0 = cnt*labels + idx[1];
                int offsetx1 = (j-1)*width+i)*labels + idx[1];
                int offsety1 = (j+1)*width+i)*labels + idx[0];
                int offsetx2 = (j+1)*width+i)*labels + idx[0];
                int offsety2 = (j+1)*width+i)*labels + idx[0];
                float cx1 = fabs(conf1[offsetx0] - conf1[offsetx1]);
                float cy1 = fabs(conf2[offsety0] - conf2[offsety1]);
                float cx2 = fabs(conf1[offsetx2] - conf1[offsetx1]);
                float cy2 = fabs(conf2[offsety2] - conf2[offsety1]);
                data_best[cnt] = cx2>cy2? idx[0]:idx[1]; //
                if (score[0]>0.4)
                    cf1[cnt] = cx1/18.0f;
                else
                    cf1[cnt] = 0;
                if (score[1]>0.4)
                    cf2[cnt] = cy2/18.0f;
                else
                    cf2[cnt] = 0;
            }
        else{
            data_best[cnt] = 0;
            cf1[cnt] = 0;
            cf2[cnt] = 0;
        }
    }
}

140
bool l2depth(LF* lf_ptr)
{
    int width = lf_ptr->W;
    int height = lf_ptr->H;
    int num_pixels = width*height;
    int num_labels = lf_ptr->nlabels;
    if (lf_ptr->type==0)
    {    
        lf_ptr->d_min=lf_ptr->dt_min;
        lf_ptr->d_max=lf_ptr->dt_max;
    }
    if (fabs(lf_ptr->d_min-lf_ptr->d_max)>2)
    
        lf_ptr->nlabels=64;
    else
    
    
        lf_ptr->nlabels=64;
        num_labels = lf_ptr->nlabels;
    float *depth_x = new float[num_pixels*num_labels];
    float *depth_y = new float[num_pixels*num_labels];
    float *depth_cx = new float[num_pixels*num_labels];
    float *depth_cy = new float[num_pixels*num_labels];
    float *confidence_x = (float*) calloc (num_pixels, sizeof(float));
    float *confidence_y = (float*) calloc (num_pixels, sizeof(float));
    uchar *depth_best_x = (uchar*) calloc (num_pixels, sizeof(uchar));
    uchar *depth_best_y = (uchar*) calloc (num_pixels, sizeof(uchar));
    uchar *depth_best_xy = new uchar[num_pixels];
    float *cost = new float[num_pixels*num_labels];
    float* cost_filter;
    float* cost2 = new float[num_pixels*num_labels];
    int64 t0, t1;
    t0 = cv::getTickCount();
    cost_volume(lf_ptr->epi_h, lf_ptr->epi_v, depth_x, depth_y,
        depth_cx, depth_cy, lf_ptr);
    compute_slope_xy(depth_x, depth_y, depth_cx, depth_cy,
        confidence_x, confidence_y, depth_best_xy, lf_ptr);
    t1 = cv::getTickCount();
vector<Mat>().swap(lf_ptr->epi_h);
vector<Mat>().swap(lf_ptr->epi_v);

lf2depth_mrf(depth_x, depth_y, confidence_x,
confidence_y, depth_best_xy, lf_ptr);
depth_filtering(lf_ptr);
color_map(lf_ptr->depth(Rect(20, 20, width - 40, height - 40)),
lf_ptr->depth_filename.c_str(), 0);
color_map(lf_ptr->depth_f(Rect(20, 20, width - 40, height - 40)),
lf_ptr->depth_filter_filename.c_str(), 0);

delete[] depth_x;
delete[] depth_y;
delete[] depth_cx;
delete[] depth_cy;
delete[] confidence_x;
delete[] confidence_y;
delete[] depth_best_x;
delete[] depth_best_y;
delete[] depth_best_xy;
delete[] d;
return true;
References


REFERENCES


[42] [www.androidcentral.com/app/google-camera](http://www.androidcentral.com/app/google-camera), 2016.


REFERENCES


[126] hci.iwr.uni-heidelberg.de/HCI/Research/LightField, 2016.


