
Virtual Computational Camera

Changyin Zhou

CHANGYIN@CS.COLUMBIA.EDU

Columbia University, 1214 Amsterdam Ave, New York, NY 10027, USA

Keywords: computer camera, optics, light field

Abstract

In computational camera research, people design unconventional optics to sample the light field in a way that is radically different from the traditional camera and preserve new forms of visual information for various applications. When one has an idea of optics design, it is very often that an evaluation based on simulation has to be conducted first before building a real system. However, how to quickly simulate an unconventional camera is often a challenging problem even with professional optical design software (e.g. Zemax). First, these software is designed for optics researches and usually do not support light field, which is a central concept in computational camera research. Secondly, researchers have to spend plenty of time in designing detailed physical profiles of optical elements. In this project, we develop a framework of virtual computational camera for light field processing. With this framework, one can easily simulate the light field processing of a virtual computational camera.

In the proposed framework, we formulate three categories of optics, including linear transform optics (lenses and prisms), dot-production optics (photomasks), and convolution optics (diffusers and some phase-plates). We propose an efficient computation strategy to render light fields at a decent resolution.

1. Introduction

Computational cameras, which use new optics to map rays in the light field to pixels in an unconventional

fashion, have attracted increasing attentions in recent years(Nayar, 2006). Due to the unconventional optics, computational cameras can optically encode new and useful forms of visual information in the captured image for the afterward interpretation. For example, researches show that a simple photomask can be used in a camera for different purposes. (Levin et al., 2007) put an optimized mask at the aperture plane for the purpose of depth from defocus. (Zhou & Nayar, 2009) optimized different aperture patterns for better performance of defocus deblurring. (Veeraraghavan et al., 2007) placed a planar photomask in the optical path between the lens and sensor for light field acquisition. Besides, many other optics, such as prisms, phase-plates, lens arrays, diffusers, and so on, have also been widely used for a variety of applications (Lee et al., 1999)(Ng et al., 2005)(Dowski & Cathey, 1995)(García-Guerrero et al., 2007).

People may notice that few of these computational designs are easy to implement. Actually it is not always possible for one to open an lens and input something inside precisely. Before doing these, one may prefer to conduct a simulation to verify the design and optimize the settings. One possible way is to use some professional optical design software (e.g. Zemax(Geary, 2002)). With these professional software, one can design the detailed profile of each optics and simulate how rays will be processed. If the profiles are given precisely, a high precision output which perfectly fits to the real result is expected. However, these optical design software are not well designed for computational camera researches.

First, these software do not support light field(Levoy & Hanrahan, 1996), a central concept in most computational researches. Some of them may be able to ray trace a planar lambertian scene and simulate a limit resolution image, but to my best knowledge, none of them is able to simulate the imaging process of a general light field. Secondly, using these software forces researchers to spend plenty of time in designing

Preliminary work. Under review by Prof. Peter Belhumeur and Jinwei Gu as a final project report of Computational Photography, Spring 2009.

detailed physics profiles of optical components. Although doing this may help to clarify the feasibility issue at the very beginning, it might not be the right time.

Also, it should be noted that for the 4D nature of light field and the complicated optics properties, in many cases, it is difficult or time consuming to write a simulator from scratch. For example, in (Veeraraghavan et al., 2007), the authors proposed to place a mask in the optical path. Intuitively, it is simple. However, most people may find it extremely difficult to simulate such a system. This is because human usually is not good at thinking in a 4D space and because the simulation algorithm is very tricky for the massive size of 4D light field.

Our motivation is to build a framework of virtual computational camera, with which researchers can quickly prototype their ideas and simulate how the systems will work with any given light field. This simulation is good for a concept verification or understanding before one really designs detailed physics profiles of optics or implements a real system.

1.1. Design Purpose

To better serve researches in computational camera, the virtual computational camera will have the following properties:

- Light field based: all the processing should be performed on light field.
- Object oriented: users can simply define optics elements and indicate where to place, and do not have to care too much about the detailed processing issues.
- Virtual: All the components are defined mathematically. Users do not have to worry about the implementation issues at this stage.
- Reasonable Resolution and precision: We require the system to afford precise simulations at a reasonable resolution.

2. Related Work

Ray tracing (Glassner, 1989)(Levoy, 1990) is a popular method widely used in graphics rendering and also in most optical design software. This method traces each light through the pixel and computes how the ray will be changed at every intersection to any object. Ray tracing is usually computational expensive, and therefore does not support light field processing well.

Light field rendering using matrix optics (Ahrenberg & Magnor, 2006)(Georgeiv & Intwala, 2006) is a more effective rendering method for imaging. By making use of the matrix optics (Halbach, 1964)(Gerrard & Burch, 1994), this method propagates the optics transform matrix instead of transforming the light field at each intersection, and therefore greatly reduces the computation cost. However, all the previous discussions are strictly constrained to a system that only has matrix optics, such as lenses and prisms. We will show that many widely used optics, like photomasks and diffusers, cannot be formulated as ray transfer matrices. These methods cannot be applied once any non-matrix optics is present in a camera.

In our work, we formulate three categories of optics, including linear transform optics (lenses, prisms), dot production optics (photomasks), and convolution optics (diffusers and some phase-plates). In addition, we show how to efficiently render the light field when they are presented together in a camera system. (Convolution optics has not been discussed yet in the current version.)

3. Light Field and Camera

3.1. Light Field

A light field is a function that describes the amount of lights in every direction through every point in a space(Wiki, 2009)(Levoy & Hanrahan, 1996). For an open space, the light field is a 4D function or can be written as a 4D array in the discrete case. Though light field can be generalized to include the temporal and spectrum dimension, we constrain ourselves to the typical 4D light field in this project.

There are various parameterizations of 4D light field, among which the most common is the two-plane parameterization shown in Figure 1 (a). In this project, a variant of UVST space shown in Figure 1(b) is used to simplify the later light field analysis.

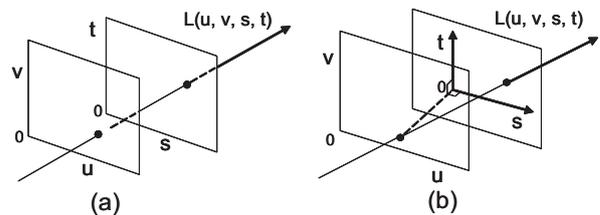


Figure 1. Parameterization of 4D Light Field. (a) The most commonly used UVST parameterization; (b) The variant of UVST parameterization which is used in this paper.

3.2. Camera

Camera is a device that projects 4D light fields to 2D images. To achieve a desirable projection, some optical elements have to be used, including lenses, coded aperture, phase plates, sensor, and so on. As shown in Figure 2, a camera can be simplified as a layered optical processing system, which takes a 4D light field as input, processes the light field with different layers of optical elements, and records a 2D image from the processed light field. Accordingly, every optical element can be defined as a light field transform function: $F : L(U, V, S, T) \rightarrow L'(U, V, S, T)$, and the sensor can be defined as a projection function: $P : L(U, V, S, T) \rightarrow I(U, V)$.

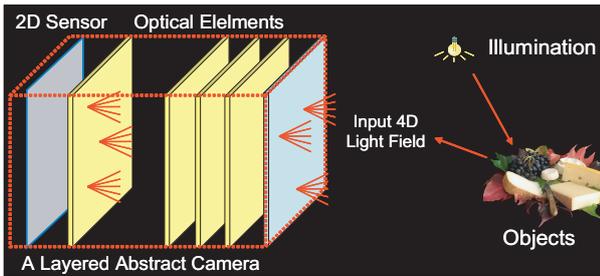


Figure 2. A Layered Abstract Model of Camera.

To simulate a camera, one can do ray tracing as in graphics rendering, that is to trace every light in the light field from the scene, through every optical element, to the sensor. But, this ray tracing is not only computationally prohibited for a decent resolution of 4D light field, but also requires tricky programming techniques due to the optical properties of each element. Actually, we can see that most commonly used optical elements, though behave quite differently from surface reflectance, can be concisely formulated in math. By making use of these properties, we can do much more efficient and precise simulation. In this section, we first formulate three categories of optical elements. In Section 4, we will discuss how to do efficient and precise simulation by using these formulations.

3.2.1. OPTICS: LIGHT FIELD LINEAR TRANSFORM

Linear or Gaussian optics can be formulated as the use of matrix methods in geometrical optics (Halbach, 1964) (Gerrard & Burch, 1994). This idea has been exploited for a long history since 19th century - the age of Gauss. Ray transfer matrix analysis (also known as ABCD matrix analysis) based on the matrix representation has been used for ray tracing in some optical systems. People usually use a 2×2 matrix and trace specific rays in a 2D space and ray transfer matrices for 2D light field have been discussed in (Georgev &

Intwala, 2006). However, with the advance of vision and graphics, light field is becoming a central concept in these research area. It is necessary to extend these transform matrix to the 4D space.

Note that this linear transform only changes the coordinates of rays, but does not change the value. Given a light field, $L(u, v, s, t)$, and a ray transfer matrix, M , the output light field, $L'(u, v, s, t)$ can be computed as:

$$L'(u, v, s, t) = L([u, v, s, t] * M^{-1}). \quad (1)$$

For a discrete light field, $[u, v, s, t] * M^{-1}$ may not be a valid integer index to L , so that interpolation method has to be used. We denote this transformation as $L' = \mathbb{T}(L, M)$.

In this section, we explicitly formulate four ray transfer matrices, which together yield a full freedom of linear transform.

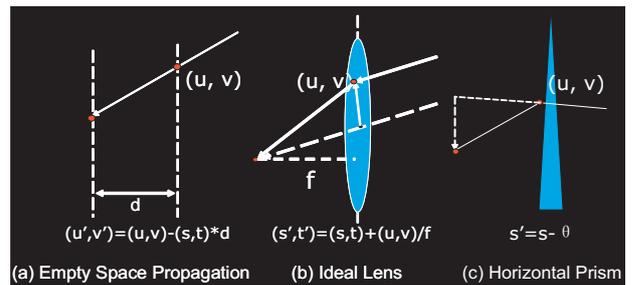


Figure 3. Linear Transform Optics. (a) A lens shears (s, t) ; (b) Empty space propagation shears (u, v) ; (c) A horizontal prism changes s .

Empty Space Propagation: First it should be noted that the light field keeps changing when it propagates over an empty space. As shown in Figure 3(a), a ray (u, v, s, t) will become $(u - s*d, v - t*d, s, t)$ after traveling a distance of d . So, the propagation can be simply formulated as a ray transfer matrix:

$$M_1(d) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -d & 0 & 1 & 0 \\ 0 & -d & 0 & 1 \end{bmatrix}. \quad (2)$$

Lens: In contract to the propagation, an ideal lens will shear S-T instead of U-V as shown in Figure 3 (b). A lens of focal length f can be formulated as:

$$M_2(f) = \begin{bmatrix} 1 & 0 & 1/f & 0 \\ 0 & 1 & 0 & 1/f \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

Prism: A horizontal or vertical prism of angle θ only

changes the s or t coordinator. Their ray transfer matrices are:

$$M_3(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\theta \\ 0 & 0 & 0 & 1 \end{bmatrix}, M_4(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\theta & 1 \end{bmatrix} \quad (4)$$

We can see that by combining M_1, M_2, M_3, M_4 , we have been able to achieve any linear transform for the light field.

3.2.2. OPTICS: LIGHT FIELD PROJECTION

Sensor: A sensor is a device which projects a 4D light field (L) to a 2D image (I) and can be easily formulated as a projection, S:

$$S(L) = \int_{s,t} L(u, v, s, t) ds dt. \quad (5)$$

3.2.3. OPTICS: LIGHT FIELD DOT PRODUCTION

Masks: As discussed in the introduction, people can also insert a mask of some patterns into the lens system. Especially, it is referred to as aperture pattern when the mask is placed at the aperture plane. Any planar mask $K(u, v)$ can also be regarded as a light field mask $K(u, v, s, t)$, where the function value is independent of s and t. Therefore, when a light field $L(u, v, s, t)$ is filtered by a mask $K(u, v)$, we will get a new light field $L'(u, v, s, t) = L(u, v, s, t) \cdot K(u, v, s, t)$.

3.2.4. OPTICS: LIGHT FIELD CONVOLUTION

Diffuser: Diffuser is another different optics element which scatters any incident ray. Although a conventional diffuser yields an identical diffusion pattern over the diffuser surface, advances in holographic research has made it possible to do diffuser coding¹. For a coded diffuser, if the diffusion pattern at the location (u, v) is defined as $D(x, y|u, v)$, then the diffuser can be formulated as a convolution:

$$L'(u, v, s, t) = \int_{x,y} L(u, v, s-x, t-y) \cdot D(x, y|u, v) dx dy \quad (6)$$

$$L'(u, v, s, t) = L(s, t|u, v) \otimes D(s, t|u, v)$$

3.3. Example 1: A Conventional Camera

Here, I take a conventional camera as an example and show how it can be interpreted in this view of light

¹It is also one of my ongoing researches, cooperated with Oliver Cossart.

field transformation. Assume there is a point light source at infinity, therefore we have a cluster of parallel input rays, as shown in Figure 3(a), denoted as $L(u, v, s, t)$. This light field is visualized in Figure 3(c) 1 as a line (a simplified 2D U-S view). The light field is first processed by a lens of focus length f , which can be formulated by the transform matrix $M_2(f)$. This transform shears the light field and yields a new light field as shown in Figure 3(c) 2. Then, when the light propagates across the distance f and arrives at the sensor plane, the light field is further sheared by the transform matrix $M_1(f)$ and we get Figure 3(c) 3. Finally, the sensor projects this light field to the U-V plane and gets an image – a focused point in this case.

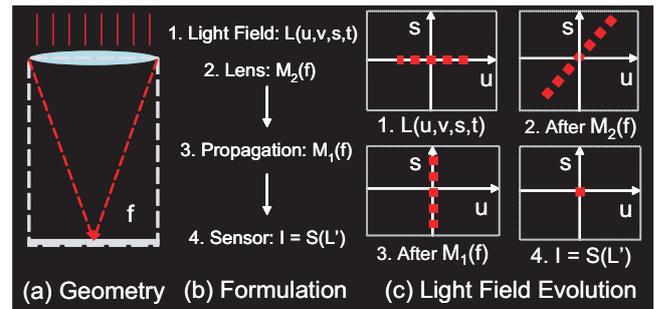


Figure 4. A ideal lens focuses parallel lights to one point in the sensor. (a) Camera Geometry; (b) Formulate the light field processing; (c) Visualize the light field at the sequent stages.

3.4. Example 2: A Camera with a Photomask Inserted

Now, let's look at a more complicated computational camera (Veeraraghavan et al., 2007). The geometry of this camera is shown Figure 5 (a). We can easily formulate this camera as in (b) and accordingly come up with a processing procedure (c). In the next section, we will discuss how to improve the efficiency of the processing and how to better preserve information during the processing.

4. Virtual Computational Camera

4.1. Light Field Generation

(7) First of all, we need to synthesize 4D light fields as inputs to the virtual camera. There are various solutions to this graphics rendering problem. Here, we show one easy way. First, we construct a scene by using some graphics tool, such as PBRT (Physically Based Rendering (Pharr & Humphreys, 2004)) or OpenGL (Woo et al., 1999); then we render a bunch of images with different camera locations (view points).

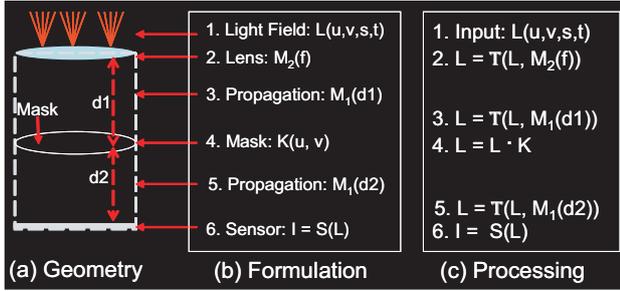


Figure 5. A camera with a planar photomask inserted in the optical path. (a) Camera Geometry; (b) Formulate every optical element

In our implementation, we write a short PBRT script to describe the scene. PBRT enables us synthesize various illumination, object geometries and BRDF profiles. Then, we use Matlab to control the camera position and call the PBRT rendering program. All the rendered images together make up a light field – each image is a 2D slice of the 4D light field at a specific (u, v) location. As input, these images will be loaded into a 4D light field matrix, $L(U, V, S, T)$. Figure 6 (a) and (b) shows two views of a light field of resolution $500 \times 500 \times 20 \times 20$ that we have generated using PBRT. This light field will be used in the later processing.

4.2. The Resolution Issue and My Solution

Due to the 4D nature of light field, a light field usually is a massive data and this poses a great challenge to the computation and memory resource. For example, the above mentioned light field of resolution $500 \times 500 \times 20 \times 20$ takes 100M bytes. Furthermore, it must be noted that when the light field is transferred by lenses or propagates along the space, the information becomes to fuse between the spatial and angular dimension, as shown in Figure 1. As a result, the resolution has to increase during these intermediate steps to avoid information loss. We found at Step 4 of Figure 5(c), it requires a resolution of $250 \times 250 \times 250 \times 250$ or higher to preserve the information well, but this resolution may lead to out-of-memory error in many systems.

To address this problem, we proposed a new processing strategy which can handle this problem efficiently without explicit light field compression (Magnor & Girod, 2000) or increasing the computation intensity. The basic ideas are:

- The input light field is usually more compressive than the intermediate light field, because most scenes tend to be lambertian and have low angular resolution. Therefore, we only do light field

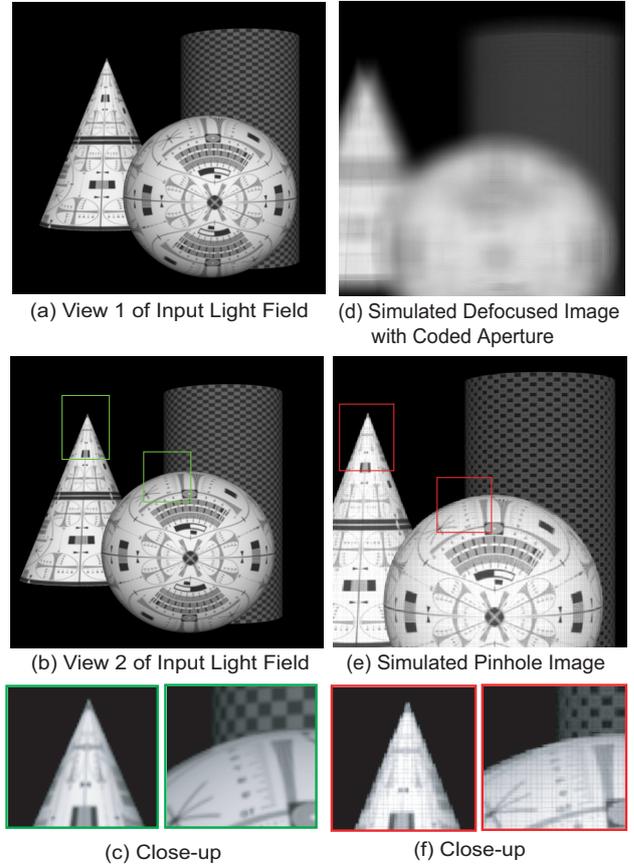


Figure 6. Light Field and Simulation Results. (a) One pinhole view of the input light field (ground truth); (a) Another pinhole view of the input light field (ground truth); (c) Close-ups; (d) A simulated defocused image of a coded aperture camera (inverted); (e) A simulated image of a pinhole; (f) Close-ups.

transform at the sensor plane.

- The output of the whole system is a 2D image. Therefore, we can combine the final project function S and the light field transform function T for acceleration.

According to these ideas, we can rewrite the processing procedure in Figure 5(c) as:

1. Input: $L(u, v, s, t)$
2. $L = \mathbb{T}(L, M_2(f) * M_1(d_1) * M_1(d_2)) \cdot \mathbb{T}(K, M_1(d_2))$
3. $I = S(L)$.

Here, Step 2 and 3 can be merged together if one only needs the final image and is not interested in the final light field.

4.3. Results

By using this strategy, we are able to simulate a computational cameras as in Figure 5 or even more complicated cameras, and process a light field of decent resolution (500 x 500 x 20 x 20) in a reasonable time. Figure 6 (d) shows the simulated defocused image (inverted); (e) shows the simulated pinhole image (inverted). Note that, when a mask is inserted in the middle of the camera, rendering a defocused image for an arbitrary light field is a very challenging problem for most rendering software. In our Matlab implementation, this processing takes 8 minutes in an Intel 3GHz PC.

Compare the close-ups of the simulated pinhole image and one of the input view (ground truth), we can see the results, though is not perfect, have been quite good. The strip artifact appears because I temporarily used the nearest neighbor interpolation method in the \mathbb{T} function.

5. Conclusion and Future Work

In this project, we have proposed a framework of virtual computational camera. Three sets of basic optical operations on light field, including linear transform, dot production, and light field convolution, are defined and implemented as basic functions in Matlab. Furthermore, we propose an efficient strategy for light field processing. With these works done, one can easily simulate and evaluate a virtual computational camera.

For the time limit, there still are several important tasks remaining undone, which includes the light field

convolution, better interpolation algorithm for the function \mathbb{T} , and the 4D Fourier analysis components.

References

- Ahrenberg, L., & Magnor, M. (2006). Light Field Rendering using Matrix Optics. .
- Dowski, Jr, E., & Cathey, W. (1995). Extended depth of field through wave-front coding. *Applied Optics*, *34*, 1859–1866.
- García-Guerrero, E., Méndez, E., Escamilla, H., Leskova, T., & Maradudin, A. (2007). Design and fabrication of random phase diffusers for extending the depth of focus. *Optics Express*, *15*, 910–923.
- Geary, J. (2002). *Introduction to lens design: with practical ZEMAX examples*. Willmann-Bell.
- Georgeiv, T., & Intwala, C. (2006). Light Field rendering. *Adobe Technical Report*.
- Gerrard, A., & Burch, J. (1994). *Introduction to matrix methods in optics*. Courier Dover Publications.
- Glassner, A. (1989). *An introduction to ray tracing*. Morgan Kaufmann.
- Halbach, K. (1964). Matrix representation of Gaussian optics. *American Journal of Physics*, *32*, 90.
- Lee, D., Kweon, I., & Cipolla, R. (1999). A biprism-stereo camera system. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 82–87).
- Levin, A., Fergus, R., Durand, F., & Freeman, W. (2007). Image and depth from a conventional camera with a coded aperture. *International Conference on Computer Graphics and Interactive Techniques*.
- Levoy, M. (1990). Efficient ray tracing of volume data. *ACM Transactions on graphics*, *9*, 245–261.
- Levoy, M., & Hanrahan, P. (1996). Light Field rendering. *SIGGRAPH* (pp. 31–42).
- Magnor, M., & Girod, B. (2000). Data compression for light-field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, *10*, 338–343.
- Nayar, S. K. (2006). Computational Cameras: Redefining the Image. *IEEE Computer Magazine, Special Issue on Computational Photography*, 30–38.
- Ng, R., Levoy, M., Brédif, M., Duval, G., Horowitz, M., & Hanrahan, P. (2005). Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, *2*.

- Pharr, M., & Humphreys, G. (2004). *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann.
- Veeraraghavan, A., Raskar, R., Agrawal, A., Mohan, A., & Tumblin, J. (2007). Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Transactions on Graphics*, 26, 69.
- Wiki (2009). Light field.
- Woo, M., Neider, J., Davis, T., & Shreiner, D. (1999). *OpenGL programming guide*. Addison-Wesley Reading, MA.
- Zhou, C., & Nayar, S. (2009). What are Good Apertures for Defocus Deblurring? *IEEE International Conference on Computational Photography*.