Depth Estimation of Semi-submerged Objects Using a Light Field Camera

by

Juehui Fan

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

In this thesis, we present a new algorithm to estimate depth of real-world scenes containing an object semi-submerged in water using a light field camera. Existing hand-held consumer light field cameras are well-suited for automated refocusing, depth detection in outdoor environment. However, when it comes to surveying marine environment and near water macro photography, all depth estimation algorithms based on traditional perspective camera model will fail because of the refracted rays. In this thesis, a new method is presented that explicitly accommodates the effect of refraction and resolves correct depths of underwater scene points. In particular, a semi-submerged object with opaque Lambertian surface with repeating textures is assumed. After removing the effect of refraction, the reconstructed underwater part of the semi-submerged object has consistent depth and shape with that of the above-water part.

With experiments on synthetic scenes rendered with modeling software Blender and real light field images captured by a Lytro Illum camera, we show that our algorithm can largely remove the effect of refraction for semi-submerged objects using an image from a light field camera.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Herbert Yang for his continuous support and patience throughout my Master program. Dr. Yang introduced light field camera to me, encouraged me to bring up a project idea and help me review my paper again and again. Without his guidance, I can not finish my thesis from scratch.

Secondly, thanks to my family and friends. Without the unconditional love from them, I can not go over the difficulties and move forward.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Light field

The theory of light field has been around for many years. It was first mentioned in [5][6][7]. A light field can be considered as a 7D function which captures each ray's wave length and direction at any time and location. If we can describe the intensities, directions of rays inside a space, then images can be rendered from any perspective. In other words, the propagation of light is recorded instead. Based on [8], the complete light field function (also known as the plenoptic function) is:

$$P = P(\theta, \phi, t, \lambda, x, y, z).$$

To understand plenoptic function, one can imagine that the intensity of light ray is recorded at every possible location (x, y, z), with every possible direction (θ, ϕ), and of every wavelength λ at any time t. The light ray direction is normalized, parametrized by tilt angle and slant angle. However, this idealized plenoptic function is almost impossible to reconstruct. To simplify it, the two-plane parametrization is generally adopted in the community of light field. Practically, light rays inside the space we are interested in is assumed to be continuous, i.e. the intensity of a light ray with the same direction is constant. Hence, the 7D light field function is simplified as:

$$P = P(u, v, s, t, \lambda).$$

The directions of rays are parameterized by two points located on two planes. Such a parameterization is called 2-plane parameterization. A ray

propagates through a point (u,v) on the first plane to a point (s,t) on the second plane. The intensity of wavelength λ is recorded. This simplification is illustrated in Figure 1.1[6].

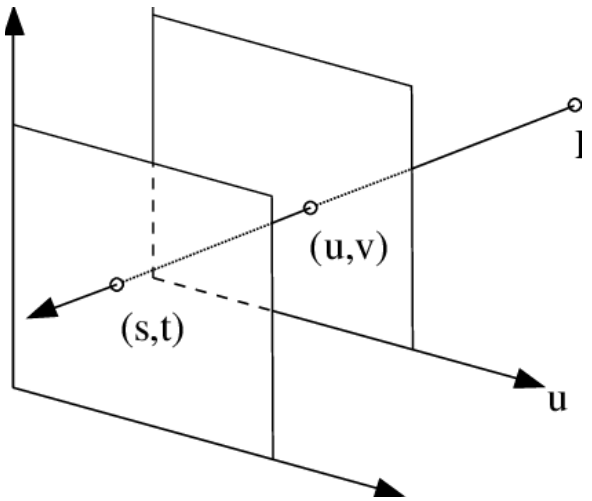


Figure 1.1: Two-plane parameterization of light field

Back to traditional computer vision methods, the three dimensional world is described using geometries. In order to render a 2D image from a new perspective, we need to reconstruct the scene, model the materials, estimate light sources, camera parameters and so on. However, from the perspective of photography, the physical world can be viewed as a light field. Depends on the structure of the scene, light rays may be reflected, refracted, absorbed inside the interested space. No matter how the light rays are transmitted, the light field representation can always help us render new photos from another viewpoint, and then reconstruct the scene.

1.2 Light field camera

For a long time, a new type of camera to record the light field is not available. As in a traditional pinhole camera, a lens is placed in front of the image sensor, then rays from different directions strike the same image sensor pixel. The 2D image of a pinhole camera can be considered as an integration of the light field over a bundle of rays. The aperture size and focal length determine the cone angle of this bundle of rays. Even though the intensity of each pixel is

recorded, the directional information from the original light field is lost due to the integration process. To record a light field, hand-held light field cameras such like Lytro and Raytrix have been designed in recent years [9][10].

A typical lenslet-based light field camera consists of one main lens and an array of micro lenslets. Compared with a traditional camera, a light field camera is able to sample more (15×15 slices for Lytro Illum camera) 2D photos from different viewpoints. Each viewpoint is located at a sub-aperture on the main lens. Rays focused by the main lens are further dispersed by a lenslet array. Thus rays of different directions strike different sensor pixels. With the lenslet array, the lost directional information of a traditional camera is preserved. The underlying principle of a light field camera is both straightforward and elegant. In a word, by placing a lenslet array between the main lens and the image sensor, a lenslet-based light field camera can record directions of rays inside the free space (a space with no occlusions and concavity) inside the camera. The principle of a lenslet-based light field camera is illustrated in Figure 1.2 [9].

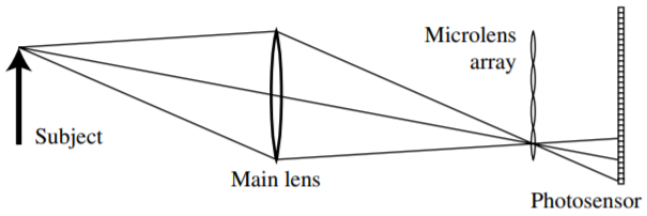


Figure 1.2: Lenslet-based light field camera

Also, the sample rate of directions of light rays depends on the number of sensor pixels under a lenslet. As we can see in Figure 1.2, if more pixels are assigned to record directional or angular information, then fewer pixels are there to record spatial information. This is the trade-off between spatial resolution and angular resolution.

1.3 Thesis work

Obviously, a light field camera has an advantage over a traditional camera because it also records ray's direction. Instead of using a complex stereo rig, one can acquire correspondences from a single image using a light field camera. Besides, images from different viewpoints can be easily decoded from raw data. Due to the limited size of the main lens, these adjacent viewpoints have significant overlap, which provides opportunities to improve the robustness of correspondence matching.

On the other hand, a light field camera has some disadvantages and constraints. They are listed below.

- First, the baseline, which is the distance between two viewpoints, of existing light field cameras is very small. If the baseline is too small, the corresponding range of disparities, which is the difference of image coordinates of an object captured by two viewpoints, must be small. For example, the effective baseline of the Lytro Illum camera is in the order of 0.01 to 0.1cm [11]. Using 2D photos from different viewpoints, the disparity is in the range of sub-pixels.

- Second, due to having some sensor pixels sacrificed to record the angular domain, the spatial resolution of captured image is relatively low. For example, the spatial resolution of Lytro Illum is only about 5 megapixels, while the angular resolution is 40 megarays. Although manufacturing companies tend to use post-processing techniques such as super resolution to address this constraint, the raw data is still a major constraint.

- Third, some light field cameras are very expensive, such as those by Raytrix, partially because these cameras are mainly for the industrial market. In the experiments, we use the Lytro Illum camera because it is affordable (<$1000 CDN) and provides us with sufficient number (11×11) of viewpoints for our experiments.

In the last decades, due to the fast development of microlens manufacturing, lenslet-based light field cameras have entered the consumer market, gain-

ing more attention in the field of computer vision for their distinctive light recording and refocusing capabilities. In order to fully exploit the potential of consumer light field cameras, we present a new algorithm to recover scene's depth of a semi-submerged object using a single light field image. In particular, our depth estimation result can remove the effect of refraction. Assume that the relative refractive index of water or any other transparent media is known, the correct shape and depth of the refracted scene points can be resolved. First, we estimate the depth information of the scene and ignore the effect of refraction. In this part, interpolation is used to estimate disparity in the range of sub-pixels. Then, using texture information, we estimate the interface between water and air. Finally, Snell's law is applied to re-triangulate the underwater part, leading to a final depth estimation without the error caused by refraction. To our best knowledge, this is the first proposed method of accommodating refraction in depth estimation using a light field camera. The contributions of the research can be summarized as follows:

- Taking pictures of semi-submerged objects is very common in macro photography. After removing the effect of refraction, correct underwater depth information is beneficial for refocusing, scene reconstruction, view synthesis and many other applications.

- In our algorithm, we also estimate the equation for the water surface. Although water is assumed to be stationary, the estimated result can be further used as input to other water surface estimation algorithms, or underwater reconstruction algorithms.

- Only a single image from a light field camera is required. Hence, our algorithm can work for dynamic scenes, which is hard for other single image depth estimation algorithms.

- Our algorithm can work with light field data from other sources. Apart from the lenslet light field camera, light field data can also be acquired with a moving camera, or even learned from a single 2D capture [12].

As more and more light field data acquiring techniques emerge, more application of depth estimation using light field data can be found.

1.4 Thesis outline

The remainder of this thesis is organized as follows: chapter 2 reviews related work; chapter 3 presents the underwater geometric model and describes an algorithm to resolve depth for semi-submerged object; chapter 4 provides experimental results on synthetic data and real data; chapter 5 presents conclusions and limitations of our work.

Chapter 2

Related Work

2.1 Depth cues from light field data

A hand-held light field camera enables us to capture a scene from different viewpoints simultaneously, encouraging increased interest in the fields of 3D reconstruction, depth estimation and view synthesis. There has been progress in the field of depth estimation using light field cameras. Previous attempts [4][13] employ multiple cues like defocus, correspondence and shading. Another interesting alternative approach [14] involves using PCA over viewpoints on a circle around the center view to enhance robustness. Other methods [15][16][17] involve the geometric analysis on the epipolar plane image of light field data. Different methods may emphasize on a typical challenge in the field of depth estimation, e.g. in the presence of occlusion boundaries, specularity, or fine structures.

Because a light field camera samples a 4D light field, there are multiple depth cues from a single light field exposure. Throughout this section, we discuss the depth cues from light field data.

2.1.1 Correspondence

The correspondence cue extracted from light field data is the same as that in traditional multi-view stereo problem. Methods developed from stereo setup are modified and applied in light field's depth estimation algorithms [4][13][18][19]. We first describe the traditional stereo setup and then analyze the specific characteristics of light field camera and the corresponding depth

estimation algorithm.

In multiple view stereo, it is assumed that the same scene is captured by multiple cameras. To start with, we assume that there are only two cameras or two viewpoints. The geometric relationship between the two views is called epipolar geometry. The epipolar geometry is depicted in Figure 2.1.

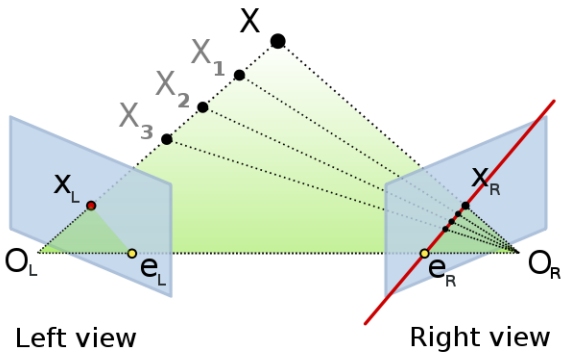


Figure 2.1: Epipolar Geometry. This is also how human eyes perceive depth information. Modified from [1].

In Figure 2.1, scene point X is captured by two cameras. This is equivalent to that scene point X projects rays to both camera centers, intersecting two photosensors at two image points X_L and X_R. The camera centers, scene point X and two rays must lie in a common plane. The line connecting the two camera centers O_L and O_R is named the baseline. From two captured pictures, the pair of corresponding image points X_L and X_R is called a correspondence. When the two image planes are parallel, the difference of locations of X_L and X_R is called disparity. Obviously, disparity is affected by the depth of X. In particular, the farther the scene point, the smaller is the disparity. This is also the principle of depth perception of the human vision system. Because disparity is used in binocular vision, it is regarded as a binocular depth cue.

If the position of scene point X changes from X_1 to X_3, then the intersected image point X_R also moves along a line accordingly, which enables us to estimate the depth of X. That line formed by all possible positions of X_R is called the epipolar line. Hence, the search for correspondences reduces to a 1D search on the epipolar line.

The fundamental matrix F which describes the relationship of correspon-

dences is a unique 3×3 rank 2 homogeneous matrix which satisfies,

$$x^{'T} F x = 0 \tag{2.1}$$

and $x^{'}$ and x can be any pair of correspondence. The properties of epipolar geometry and the equation of Fundamental matrix are discussed in details in ref. [1].

In a light field camera, a lenslet array is placed between the sensor and the main lens. Each viewpoint is actually a sub-aperture image. The number of microlenses is the number of viewpoints. The microlens array can be modeled as a grid of viewpoints. As can be seen in Figure 2.2, different viewpoints C_i and C_j are on the same plane such that the transformation between any two viewpoints is a translation. Because adjacent views are next to each other on the same lens, the baseline of two adjacent viewpoints is much smaller compared to that in traditional stereo.

To handle the micro baseline in light field data, we use spline interpolation to estimate sub-pixels. To handle the micro-baseline in depth estimation using a lenslet-based light field camera, other methods [4][20][19] use bilinear interpolation, bicubic interpolation or phase shift theorem, respectively. In [20], they interpolate to sub-pixels using bicubic interpolation to calculate optical flow for each pair of images. Their depth estimation method is built on a structure of motion framework. In [19], the cost volumes for correspondences are calculated after estimating sub-pixels in the Fourier domain.

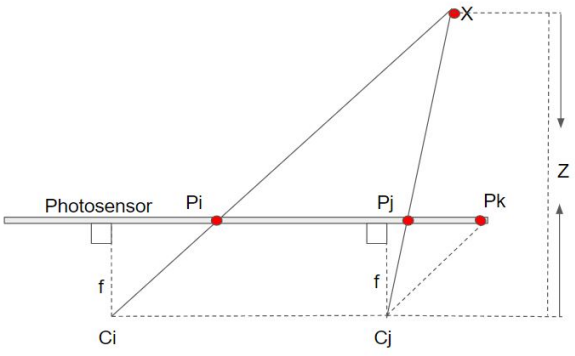


Figure 2.2: Two-view geometry in a light field camera.

As shown in Figure 2.2, scene point X is captured from two viewpoints

simultaneously. The line connecting viewpoints C_i and C_j is the baseline. Because ray XC_i intersects the photosensor at the location P_i, scene point X is observed at the pixel located at P_i. Similarly, from viewpoint C_j, scene point X is observed at the pixel located at P_j. Let the line C_jP_K be parallel to the line C_iX, then triangle $C_jP_kP_j$ and triangle C_iXC_j are similar triangles. The difference between P_j and P_k is the difference between pixel locations from two viewpoints, so the length of P_jP_k is the disparity. The relationship between disparity and depth can be derived from similar triangles.

$$\frac{d}{b} = \frac{f}{z}. \tag{2.2}$$

In the equation above, d is the disparity between a pair of correspondence, b is the baseline of two viewpoints, f is the focal length, and z is the depth of the observed scene point.

If we stack pictures from different viewpoints together into a volume, then the image on the side of the volume we got is called epipolar plane image. The concept of epipolar plane image was first mentioned in [21]. Figure 2.3 is an epipolar plane image constructed from a light field data captured by a Lytro Illum camera. In the epipolar plane image, each row is taken from one viewpoint. Along the red line is the correspondence found in each viewpoint. As the translation of the current viewpoint to the first viewpoint is increasing along this red line, the disparity is also increasing. In other words, the tangent of a line in epipolar plane image can be regraded as $\frac{\delta d}{\delta b}$. As shown in equation 2.2, this is another depth cue.

The epipolar plane image is a two-dimensional slice of the full light field data, in other words, a flatland light field. In equation 2.3, the eipolar plane image is constructed by stacking the y spatial dimension and the v angular dimension when u equals to u_i, x equals to x_i. The u coordinate and x coordinate are fixed. As long as we fix one spatial dimension and one angular dimension, the flatland two dimensional image is an epipolar plane image,

$$EPI(u_i, x_i) = L\left(u_i, v, x_i, y\right). \tag{2.3}$$

Many methods exploit the potential of depth estimation using epipolar plane image, such like [22][15][23]. Depth estimation on epipolar plane image

is straightforward and more efficient, but the constraint is that the light field data has to be densely sampled.

Figure 2.3: An epipolar plane image: Along the red line is a set of correspondences found from different adjacent viewpoints. The tangent of this red line is a depth cue: $\frac{\delta d}{\delta b}$.

Also, in a traditional stereo setup, we need multiple exposures from multiple cameras. Using a light field camera, we only need one exposure from one camera. The microlens array allows us to retrieve direction and magnitude of rays at the same time, providing with multiple viewpoints in one exposure.

A common assumption for correspondence-based methods is: the scene point is Lambertian or diffuse. For a Lambertian scene point, the image points of the same scene point from different viewpoints have the same intensity. However, for specular objects, intensities of the same scene point when viewed from different viewpoints will be different. Hence, with specularity, the correspondence depth cue does not work correctly.

Correspondence is more robust on scenes with high textures and edges. If the scene is textureless, then objects with different depths will have similar colors and are no longer distinguishable. Another interesting topic is to use image gradients to guide the depth propagation process. In an image, the regions with high image gradients might be edges or texture variations. In [24], they use bidirectional photo-consistency to differentiate texture edges from silhouette edges in order to propagate depth information to low-gradient regions. In [19], depth filtering guided by the central view image is applied to the depth map to preserve edges.

2.1.2 Defocus

Based on the Lambertian assumption, another depth cue from a light field data is the defocus cue. If a part of the scene is focused, then its image patch

is sharper. Otherwise, a not-in-focus scene is blurry. This kind of monocular depth cue can provide information on the depth from one viewpoint.

Taking the light field data as input, if we calculate the average intensity among different viewpoints, then the average image is similar to a photo from a traditional pinhole camera. After we refocus the light field image to different depths, then multiple defocus cues will be available.

Below is the refocusing effect of a first generation Lytro prototype camera [9]. Refocusing using the light field data can be done by a simple shear transformation.

Figure 2.4: Light field camera can refocus a captured image to different depths. Rather than by simulation using blurring, refocusing of a light field camera is physically-based.

2.1.3 Specularity

Both correspondence and defocus cues are not robust in the presence of specularity. Instead of the Lambertian assumption, other methods tend to use more complex but physically correct models. In papers [25][26], the BRDF (Bidirectional reflectance distribution function) model is used to handle non-Lambertian effects. The coefficients of the specular term is determined by two directions: the direction of the incoming light and the viewing direction. Because of the redundancy of light field data, they derive another BRDF in-

variant to solve for the depth and normal by assuming that the surface is smooth or a set of bicubic patches. The equation below is the BRDF function, x the position of a scene point, n the normal of the scene surface, s the incoming light direction determined by the position of the light source and the position of the scene point, and v the viewing direction,

$$\rho(x, n, s, v) = (\rho_d(x, n, s) + \rho_s(x, \hat{n}^T \hat{h}))(\hat{n}^T \hat{s}). \tag{2.4}$$

Noted that a common drawback of methods based on the BRDF model is that they are more suitable for controlled scenes. Because in a BRDF function, the location of the light source must be known or it requires to be estimated along with depth. Other depth estimation methods [18][27] which explicitly handle specularity classify the scene into diffuse parts and specular parts, and then iteratively propagate the estimated depth from the diffuse part to the specular part.

Other methods like [28][12] trained convolutional neural networks to synthesize new views and estimate the depth maps. Because the ground truth for view synthesis is available, it is feasible to train a CNN to synthesize a new view. Jointly with the new views, the depth map is estimated indirectly. Methods based on CNN can tackle the problems of specularity and occlusion without incorporating strict assumptions and physical-accurate models. However, the training data needs to be chosen judiciously. For example, in [12], due to the diversity of one training data set, their network can not work robustly even though there are 3243 images in the training data.

2.2 Accommodating refraction

Even though the methods mentioned above can estimate depth of a realistic scene captured using a light field camera, none of them allows us to accommodate the effect of refraction. Realistic photography is often composed of various objects made from different materials, including the effect of refraction. Transparent media such as air, water or glass do not have the same refractive index, which is determined by the speed of light inside them. According to

Snell's law, when light rays enter another medium with a different refractive index, they are refracted or bent, following a nonlinear path. Illustrated in Figure 2.5, a pencil is semi-submerged in water. The picture of it is distorted because of refraction.

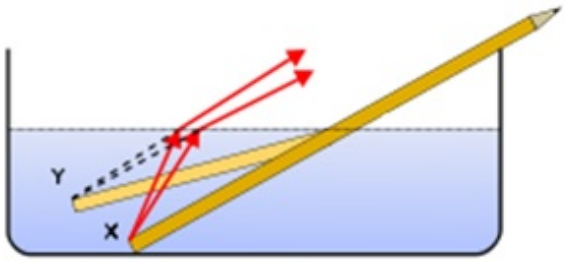


Figure 2.5: A pencil is semi-submerged in a water tank. The underwater part is refracted. The correct position is at X. Due to refraction, the wrong position Y is observed.

This kind of non-linear path change forces the real image formation process to deviate from the traditional perspective camera model. In this case, because of the invalid camera model, the corresponding depth map and 3D reconstruction result using a method for air will no longer be valid. Hence, we have to explicitly model the effect of refraction to recover the actual depth of underwater scene points. Methods like [29][30][31][32] take the nonlinear path into account and they build the methods by involving refractive plane into the general pinhole camera model. Another option is to use a virtual camera [33][34]. The virtual camera is first proposed by [34]. All rays in water intersect a common axis defined by the cameras center of projection and the normal of the interface between two medias.

In our proposed algorithm, we incorporate the depth estimation method from [4]. Furthermore, the effect of refraction is also explicitly considered by incorporating the refraction into the pinhole camera model.

To remove the effect of refraction from our estimated depth map, identifying the underwater part of the semi-submerged object is a crucial step because only the underwater pixel's depth needs to be corrected. Besides, to compute the actual non-linear paths of light rays, we must know the normal of the water

surface in advance. In this thesis, we assume that the water is stationary and does not have waves. To directly recover the shape of dynamic water surface from light field images is beyond the scope of this thesis.

In the field of water surface reconstruction, we are not aware of any existing research that could recover the water surface using a light field camera. In [23], they assume two signals can be detected in the light field data, one from the object, one from the refractive or reflective surface. Then a structure tensor is used to separate these signals and to reconstruct the occluding reflective or refractive surface. But the signal of clear water is almost undetectable. Data from a consumer light field camera is much noisier, making it even harder to apply the structure tensor method. Another available approach is described in ref[35]. With a known pattern placed underwater and two cameras mounted on a stereo rig to capture correspondences, this method specially targets at flowing water. But as we mentioned, the baseline of a consumer light field camera is very small, making this method not applicable.

2.2.1 Texture analysis

In our method, we take advantage of the boundary pixels between air and water to estimate the normal of the water surface using texture analysis. To automatically estimate the water surface, we assume that the semi-submerged object is covered with similar textures such that the water surface can be identified by analyzing the change of texture caused by refraction.

Back in the 1950s, Gibson first proposed that surface orientation can be perceived from texture analysis [36]. Due to differences in orientation and depth of the surface, textures are perceived differently. This is what has been termed as texture gradient by Gibson. For example, by observing windows on a wall, we can infer the orientation of this wall. As the window (texel) gets smaller, it gives us a clue that the wall is getting further away from us. As depicted in Figure 3.5, the windows on the building is a kind of texture. As a wall of the building gets further way, the window texture gets finer.

To recover the scene structure from textures, there are mainly two approaches. The first approach is based on size and density of texels which vary

Figure 2.6: A monocular depth cue: the size and orientation of texture will vary with depth.

with the distance from the observer. As the surface gets further away, the texture gets finer and appears smoother [37]. Based on this observation, the texture gradient information is gathered from patches inside an image as a depth cue. However, the gradient information relies on the size of the image patch, which in turn makes it of limited use when the interested surface gets smaller. To avoid this drawback, the second approach uses texture cue extracted from each texel by considering its aspect ratio. When the plane of a circle is slanted from the viewing direction, the circle appears to be an ellipse. Similarly, the aspect ratio of each texel also reveals the surface structure. This kind of method has been fully discussed in [38][39]. They use statistical models to estimate the probability of different surface orientation considering the observed distribution of texels with various aspect ratios. In this thesis, we combine both approaches to improve the efficiency of our algorithm.

Chapter 3

Refraction Corrected Depth Estimation

In our algorithm, we assume that the surface of the semi-submerged object has similar textures. The input to our algorithm is a light field image $LF(u, v, x, y)$, where (u, v) are the angular coordinates, the position of viewpoint, and (x, y) are the spatial coordinates, the 2D pixel's location. The output is a depth map after removing the effect of refraction.

3.1 Overview

Briefly, our algorithm consists of three steps:

1. Depth estimation using correspondence. In this step, we ignore the effect of refraction and estimate a temporary depth map using correspondence, from which the incorrect underwater depth result is estimated.

2. Estimate the normal of the water surface using texture analysis. Next, to recognize the underwater part, we analyze the orientation of the surface from texture gradient to extract boundary pixels between air and water. The calculated normal of the water surface allows us to further calculate the real path of refracted rays in the next step.

3. Calculate the real depth using Snell's law and depth estimation from step 1. We recalculate underwater scene points using the correct light rays and triangulate where the underwater objects are.

3.2 Depth estimation using correspondence

In the first depth estimation step, we ignore the effect of refraction and specularity. The actual scene points can be modeled as a composition of diffuse part and specular part. A Lambertian scene point demonstrates photo-consistency among different viewpoints. In other words, if we look at the same Lambertian scene point from another view, the observed color should be the same.

Given a light field image $LF(u, v, x, y)$, we can vary the (u,v) angular coordinates to get 2D spatial images from other slightly different viewpoints. Assumed that the captured scene is Lambertian. If the scene point is refocused to the correct depth, then ideally, different viewpoints will capture the same color for the same scene point. In contrast, if the scene point is refocused to the wrong depth, the observed color will be different in different views. In order to distinguish between correct and incorrect depths, we refocus the light field image to different depths and the depth is determined as the correct depth when the color variance among different viewpoints reaches the minimum.

Ng et al. first explained how to refocus a 4D light field data in [9]. Basically, a refocused light field $LF'(\alpha)$ is a sheared version of the original one:

$$LF'(\alpha) = \int\int L\left(u', v', u' + \frac{x' - u'}{\alpha}, v' + \frac{y' - v'}{\alpha}\right). \tag{3.1}$$

The new refocused light field data is just a summation of sheared subaperture images, with shear value α. After the light field image is refocused with different α, we can compare their color variance among all viewpoints. The α with the minimum variance, i.e. the highest photo consistency, is chosen as our estimation. The equation of color variance at pixel (x, y) with shear value α is:

$$\sigma_\alpha(x, y)^2 = \frac{1}{N} \sum_{(u',v')} (L_\alpha(x, y, u', v') - \overline{L}_\alpha(x, y))^2. \tag{3.2}$$

In the above equation, the total number of viewpoints (u,v) is denoted as N and $\overline{L}_\alpha(x, y)$ is the mean value of color variance. This method was first given by Tao et al. [4]. In their algorithm, they combine this result with another defocus response. However, our main focus is to avoid the effect of

refraction so we only make use of the correspondence cue. Even though depth derived from correspondences may yield bad results if the scene has repeating textures or occlusions, it still opens up a possibility for us to discover the effect of refraction, and then gradually improve the current depth result in the following steps.

3.2.1 Interpolation to sub-pixel

From equation 3.1, we can see the shear transformation in light field will reach sub-pixel level. In order to improve the accuracy of the shear transformation, we need to do interpolation to estimate the intensity value at a sub-pixel location. Among interpolation algorithms, bilinear interpolation, bicubic interpolation and spline interpolation are widely used. Many publicly available modules such like GNU's ALGLIB library have implemented these numerical methods. But in order to increase the efficiency of our implementation, we implement this part in C and wrap it using mex to work with other matlab codes. Part of our implementation is based on the code from [40].

Bilinear and bicubic interpolation are both polynomial interpolation methods. The difference is that bilinear interpolation uses the nearest 4 neighbors but bicubic interpolation uses the nearest 16 neighbors. The equations for them can be summarized as follows. i and j are indices for the x and y coordinates. For bicubic interpolation, i_{max} and j_{max} are equal to 3, while for bilinear interpolation, i_{max} and j_{max} are equal to 1.

$$f(x,y) = \sum_{i=0}^{i_{max}} \sum_{j=0}^{j_{max}} a_{ij} x^i y^j \tag{3.3}$$

Spline is another popular method for interpolation especially in the field of computer-aided geometric design. The shape of the curve interpolated by spline is controlled by several control points. For example, in the industry of automobile, a designer can design a curve by shifting control point's position, which is far more convenient than using an equation. The main advantage of spline interpolation can be displayed in Runge's example [41], where high order interpolation using a global polynomial often exhibits oscillations. If we

imagine these sample points as training samples in the field of machine learning, then Runge's example is like over fitting. This phenomenon is illustrated in Figure 3.1. The classical Runge's phenomenon is about the polynomial interpolation of Runge's function, whose equation is:

$$f(x) = \frac{1}{1+25x^2}, x \in [-1, 1]. \tag{3.4}$$

Normally, higher order polynomial interpolation should be more accurate compared with lower order polynomials. However, if we interpolate Runge's function using lower order polynomials whose equation is 3.5 and high order polynomials whose equation is 3.6, then the result is not what is predicted. The detailed derivation process for 3.5 and 3.6 can be found in paper [2].

$$p_1(x) = 1 - \frac{3225}{754}x^2 + \frac{1250}{377}x^4 \tag{3.5}$$

$$p_2 x = 1 - \frac{98366225}{7450274}x^2 + \frac{228601250}{3725137}x^4 - \frac{383000000}{3725137}x^6 + \frac{200000000}{3725137}x^8 \tag{3.6}$$

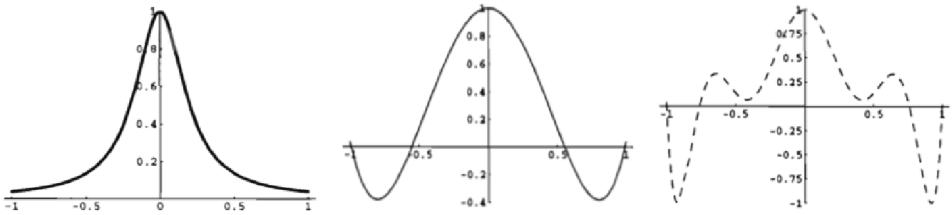


Figure 3.1: Runge's example [2]. Runge's function is plotted on the left. The rest two figures are polynomial interpolation of runge's function. The rightmost curve is interpolated with higher order polynomials (3.6) but exhibit oscillation. Piecewise spline interpolation can avoid this kind of oscillation.

Compared with traditional interpolation methods, spline is like a "piecewise" interpolation. Assume we have $n+1$ data points $\{(x_i, y_i) : i = 0, 1, ..., n\}$, then there are n intervals, each defined by a pair of knots (x_i, y_i) and (x_{i+1}, y_{i+1}). For each interval, the polynomials p_i that we use may not be the same, which are local polynomials rather than global polynomials. But both first and second derivatives are continuous everywhere. In addition, we have,

$$\begin{aligned} p_i^{'}(x_i) &= p_{i+1}^{'}(x_i) \\ p_i^{''}(x_i) &= p_{i+1}^{''}(x_i) \end{aligned} \tag{3.7}$$

The smoothness of a spline can be further expanded to higher order of n, which means that at each knot (x_i, y_i), adjacent polynomials share common derivative values of order n. In our algorithm, we use spline interpolation to estimate corresponding points at sub-pixel locations.

To this end, the first step of our depth estimation has finished. Please be noted that the current depth map does not recognize any underwater part, but only provides an initial guess for the underwater depth. As a result, the current underwater result is completely wrong. It is a very common phenomenon in our daily lives that refraction will make our visual depth estimation inaccurate. For example, if a fisherman wants to spear a fish swimming in water, the spear needs to go deeper than that observed visually because the virtual depth acquired by the eyes is not accurate, which is shallower than the real depth of the fish. It is because the light ray is refracted but we do not realize this when our eyes are doing backward ray tracing to "estimate" the depth of an underwater object. Similarly, the underwater depth map for this part is regarded as the virtual depth, which is incorrect but will be corrected in our algorithm.

3.3 Estimate normal of water surface

In this part, we assume that the water surface is flat and stationary, with one consistent normal. We combine a texture descriptor method and a statistical model in texture analysis to pick out pixels along the boundary between air and water. Because of refraction, underwater distortion leads to inconsistency of surface orientation and depth along the boundary. For example, in synthetic data, we generate the correct depth map and the incorrect depth map in Figure 3.2 and Figure 3.3, respectively. For the underwater part, the correct depth map is denoted in red and the incorrect depth map in green. Darker color means closer. In Figure 3.2, a straight ruler is semi-submerged into water, such that its underwater part is completely distorted, both in its shape and depth. In Figure 3.3, the orientation of the surface of a cone is also changed due to refraction.

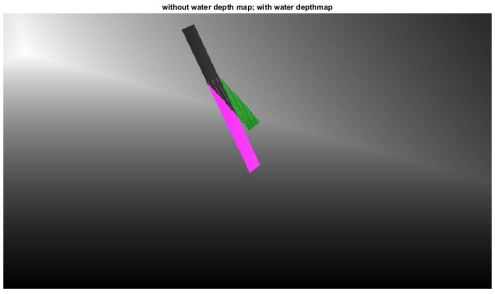


Figure 3.2: Depth map of a ruler in water. Correct depth in color red, wrong depth in green.

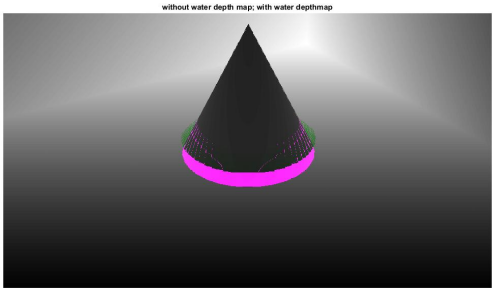


Figure 3.3: Depth map of a cone in water. Correct depth in color red, wrong depth in green.

Given the texture information, this inconsistency can be identified at the same time revealing the position and orientation of the water surface. To begin texture gradient analysis, we first gather patches with a width of 50 pixels using the center light field image and calculate a texture descriptor for each patch to describe the gradient of texture's size and density. Gradients are calculated from the power spectrum of the Fourier transform of the image, which is first proposed in [37]. After transforming the coordinates of the power spectrum into the polar coordinate system, the size and density of repeating patterns can be represented by the radius of the power spectrum at each peak direction. A snapshot of the texture descriptor analysis is shown in Figure 3.4.

To this end, image patches where size and density display inconsistency are recorded for the next step to refine the search range.

Secondly, we apply the method in [38] to estimate the probability of differ-

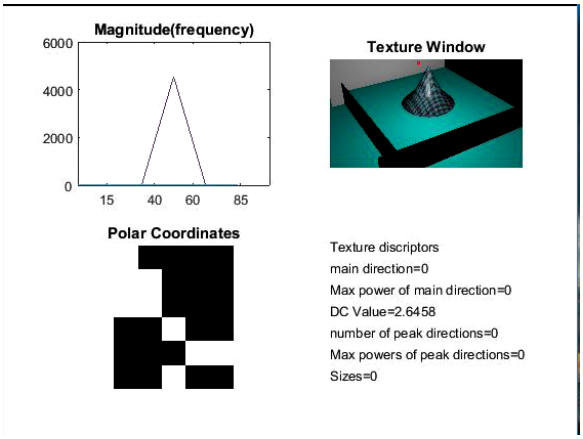


Figure 3.4: Snapshot of texture descriptor.

ent surface orientations, but only on the image patches found above. In order to estimate the surface orientation, we need to calculate every candidate orientation's probability, our combined method is more efficient by narrowing down the search range. Next, boundary pixels are chosen from pixels whose surface orientations display a sudden change compared with its neighboring pixels. Based on all observed curves' orientation α_i, we can estimate the probability of each surface orientation (σ, τ):

$$\prod_{i=1}^{n} \frac{\pi^{-2} sin\sigma cos\sigma}{cos^2(\alpha_i - \tau) + sin^2(\alpha_i - \tau) cos^2\sigma}. \tag{3.8}$$

Boundary pixels found in this step are illustrated in Figure 3.5. We refer the interested reader to the original papers for a more comprehensive description of this statistical model. Then, the best fitting plane is calculated by minimizing the orthogonal distances from the set of boundary pixels to the water surface as an initial estimation of the water surface.

3.4 Remove the effect of refraction

At the interface between air and water, all refracted rays obey Snell's law, which allows us to compute the correct depth. The underwater light path can be calculated by making use of the normal calculated from texture analysis. According to Snell's law, the relationship between the refracted ray and the original ray depends on the relative index of refraction (IOR) and the normal

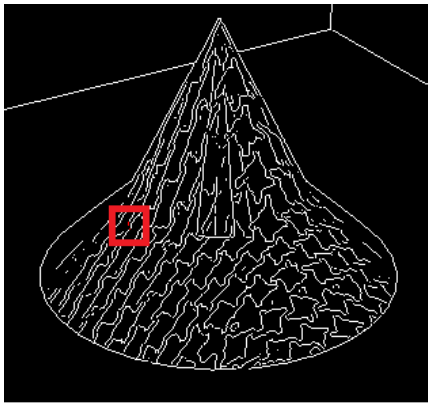


Figure 3.5: An image patch (red square) and boundary pixels (red dots)

of the interface:

$$\frac{sin\theta_1}{sin\theta_2} = \frac{IOR1}{IOR2} = \eta. \tag{3.9}$$

Angle θ_1 is the angle between the ray in air and the water surface normal, while θ_2 is the angle between the ray in water and the normal. Instead of angles, Snell's law can also be represented using rays:

$$\vec{w} = \eta\vec{a} + (-\eta\vec{a} \cdot \vec{n} - \sqrt{1 - \eta^2 + \eta^2(\vec{a} \cdot \vec{n})^2})\vec{n}. \tag{3.10}$$

In equation 3.10, $\vec{a}$ is the refracted ray in air, while $\vec{w}$ is the original ray from an underwater scene point, which is exactly the correct ray we are looking for. $\vec{n}$ is the normal of the water surface and the relative refractive index η is assumed to be known. In the experiments, we assume the relative refractive index of water is 1.33. With Snell's law, the correct direction of underwater ray $\vec{w}$ can be calculated.

In Figure 3.6, the wrong underwater scene point is denoted as F while the correct underwater scene point is denoted as R. C_1 and C_2 denote two viewpoints on the main lens of the light field camera. $\vec{n}$ denotes the normal of the water surface. Although in this figure, the water surface is parallel to the photosensor. In our experiments, the position of the water surface is not constrained. It is noteworthy that if the water surface is not parallel to the sensor plane, our method can still work.

Based on the depth map from Section 3.2, the incorrect underwater position F can be computed. Although wrong positions lead to wrong pixel locations

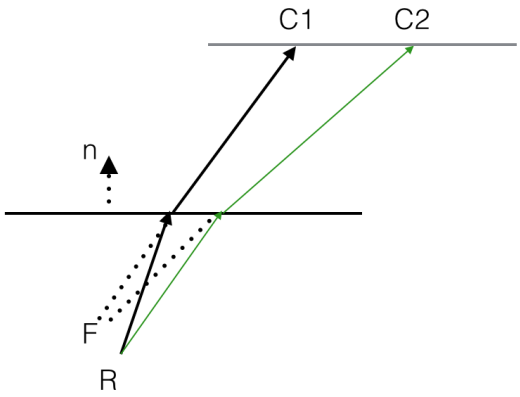


Figure 3.6: Refracted Rays (see text for explanation)

and depth results, they still unveil the ray direction in air, $C_1 - F$. By applying Snell's law, two correct rays $\vec{w}$ are used to triangulate the correct position of the underwater scene point. All points are in the camera coordinate system. In the real experiments, we use The Light Field Toolbox [42][43] to estimate the intrinsic parameters of the camera.

Since the estimated water surface from Section 3.3 is not accurate, a boundary constraint is developed to modify the normal of the water surface. Because boundary pixels are close to the water, their recalculated locations must be very close to their original pixel locations. We gradually vary the orientation (σ, τ) of the water surface and calculate the corrected pixel locations for all boundary pixels. The orientation which can minimize the distance between the corrected pixels and the old pixels is selected as the final estimation, which is then used to correct the depths of all underwater pixels. With a new orientation (σ_i, τ_i), the new equation of the water surface is recalculated, as $a_i x + b_i y + c_i z + d_i = 0$. Using the new water surface, we can re-triangulate the boundary pixels whose locations are BP based on the Snell's law shown above and get the new pixel locations BP'. Finally, the water surface is reselected to minimize the difference between BP and BP'. In equation 3.11, j is the index for boundary pixels.

$$(\sigma, \tau) = \min_{(\sigma,\tau)} \sum (BP(j) - BP'(j)) \tag{3.11}$$

The point clouds of wrong underwater scene points (red) and corrected

scene points (blue) are shown in Figure 3.7 and Figure 3.8. The point cloud is from our synthetic experiments.

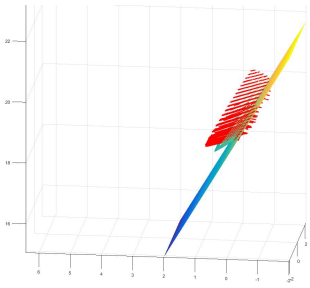


Figure 3.7: Underwater point cloud. The plane represents the water surface. The red point cloud is wrong. The blue point cloud is corrected from the wrong point cloud.

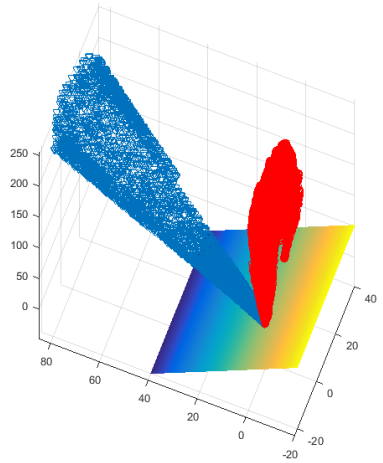


Figure 3.8: Another viewpoint of the underwater point cloud. The plane represents the water surface. The blue point cloud is corrected from the wrong point cloud using Snell's law.

In Figure 3.9 and Figure 3.10, we demonstrate the point clouds from two synthetic experiments. In Figure 3.9, the water surface is at $z = 3$. In Figure 3.10, the water surface is at $z = 1$. Because of refraction, the traditional depth estimation method can not work for the underwater part, leading to a wrong, refracted point cloud, which is illustrated as the red point cloud. Reversing the process of Snell's law, we are able to recover the real positions of the underwater points.

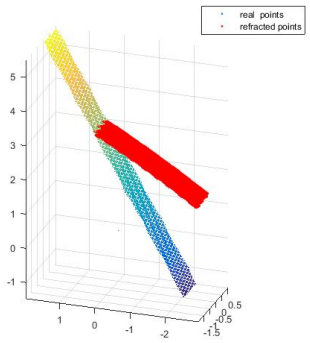


Figure 3.9: This is the point cloud in world coordinates. With a water surface at $z = 3$, the scene points whose z value is less than 3 are refracted. In the pinhole camera model, the light ray is assumed to be straight. Traditional reconstruction algorithm can only reconstruct a wrong, bent point cloud, which is shown in red.

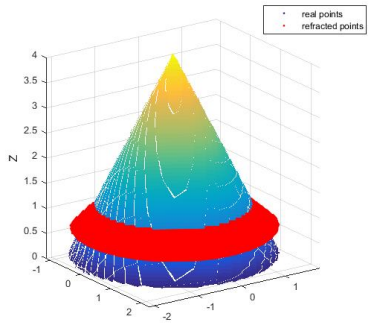


Figure 3.10: This is the point cloud in world coordinates. With a water surface at $z = 1$, the scene points whose z values are less than 1 are refracted. The wrong point cloud is shown in red. The corrected point cloud is shown in blue.

Chapter 4

Experiments

To verify our algorithms, we did experiments on synthetic data and real data. In the synthetic experiments, we apply our methods to synthesized 3D scenes rendered by an open-source modeling software, Blender. For simple scenes, the blender file is created by us and are rendered using a non-photo-realistic ray tracing engine, Blender Internal Renderer. For more complicated photo-realistic scene, we use the blender files from a benchmark dataset and did some modifications to include refraction. The complicated scene is rendered by a photo-realistic ray tracing engine, Blender's Cycles Renderer. In the real experiments, we take photos for semi-submerged objects using a Lytro Illum light field camera.

4.1 Synthetic data

In a light field camera, there is a microlens array to retrieve directional information of rays. However, it's hard to model a lens array in Blender. To model a light field camera, a grid of pinhole cameras are modeled to capture photos from different viewpoints. They share the same focal plane and the baseline of adjacent cameras are the same. We model the light field camera by using an addon created by the University of Konstanz and the HCI group at Heidelberg University [3]. The output of our work is the depth map of the central viewpoint.

4.1.1 Dataset from Blender's internal render engine

There are two ray tracing engines in Blender. One is the built-in Blender Internal Renderer and another one is the photo-realistic Cycle's Renderer. At first, we create two simple scenes and render photos using Blender's Internal Renderer. Even though Blender Internal Renderer is not physically accurate, it is faster and due to our Lambertian assumption, the realism of Blender Internal is acceptable.

In the synthetic experiments, we apply our method to two different scenes. Although many light field benchmark data sets are available right now, we cannot use them directly like other light field research work. The reason is that all of them do not have water or any other refractive materials and hence the effect of refraction cannot be observed in them. So we build our own 3D scenes using Blender 2.77, an open source modeling and animation software. 121 cameras are arranged on a 11×11 grid to simulate a real light field camera. Each camera's FOV is 40 degrees. The refractive index of water is set at 1.33.

The first scene is similar to the popular physical experiment in a typical high school, i.e. a pencil is semi-submerged in water. There is a water tank and a ruler in the scene. The straight ruler is partially submerged in the water tank. The water tank is surrounded by two gray walls. We can find obvious underwater distortions of this ruler caused by refraction. The ruler appears to be bent and its above water texture looks different from that under the water. In the second scene, the submerged cone and water tank are placed at a different position. At the same time, we also change the position of the point light source. However, the setting of our 121 camera rig is still the same. Because after some tests, this setting is the best one to simulate a consumer Lytro camera.

From our results, we show that our method can remove the effect of refraction to some extent. The corrected underwater part demonstrates consistency in shape and depth with the above-water part. The refracted underwater scene is always smaller in size compared with the same scene without water. Especially at the intersecting positions between air and water, scene points which

can be observed without water can not be captured because of refraction. Due to this kind of distortion, the corrected pixels can not be perfectly connected with the above-water pixels. Similarly, in the segmented coded depth map, the upper above-water part is not fully connected with the underwater part. But basically, the corrected depth map can provide a good insight for the underwater portion's depth and shape.

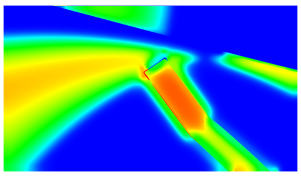
To provide a quantitative analysis, we measure the RMSE (root mean squared error) with the ground disparity map. After removing the effect of refraction, for the first scene, the RMSE drops from 0.025 to 0.024, for the second scene, the RMSE drops from 0.031 to 0.030. After the correction, the shape of underwater portion is corrected, leaving the data of the original pixels unaccessible. Because of refraction, some background pixels are occluded. But after the effect of refraction is removed, those background pixels can not be recovered because they are not captured by the camera. And the RMSE highly depends on the size of the underwater portion. If the underwater part is larger, then there are more pixels corrected, the improvement is larger.

4.1.2 Realistic benchmark dataset

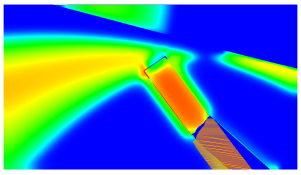
Rendering a scene with the photo-realistic ray tracer, the Blender's Cycles Renderer, is time-consuming. Cycles renderer is a backward path tracing engine. In order to make the synthesized capture more photo-realistic and avoid artifacts, this kind of engine will trace a huge amount of rays. Cycles Renderer is able to sample thousands of rays for each pixel. Hence, the number of samples will affect the rendering speed. For example, if the sample size is changed from 3000 to 1000, then the rendering time for a single image will decrease from 16min20s to 6mins.

To speed up the rendering process, we choose GPU computing and decrease the sample size in the ray tracing engine from 3000 to 1000. Using a computer with the Nvidia GeForce GTX 970 grahics card, i7-6700 CPU and 16GB memory, it takes about 4 hours to render a synthetic scene. We render one realistic scene using cycles' renderer, which is from a light field benchmark dataset[3]. To model the effect of refraction, we add a cube with refractive

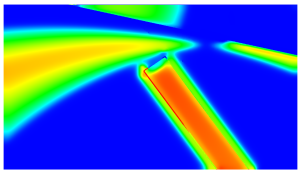
(a) Pencil in the water



(b) Uncorrected depth map

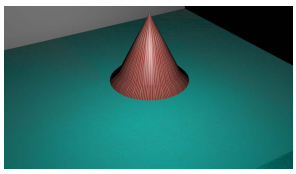


(c) Corrected depth map without refraction

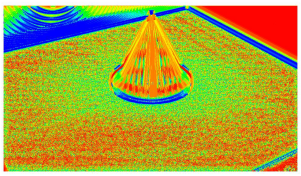


(d) Depth map after water is removed in the synthetic scene

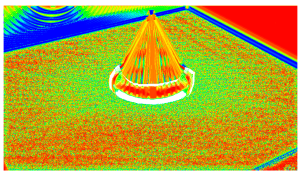
Figure 4.1: Synthetic experiments' results: Figures a,b,c and d are results of the first synthetic scene, where a straight pencil is submerged in water. a is the central sub-aperture light-field image. Figure b shows the depth estimation result using correspondences, and c is the corrected depth after removing the effect of refraction. The color varies from blue to red, the more reddish, the closer. Figure d is the depth estimation results using correspondences after removing the water in synthetic scenes, in other words, the ground truth.



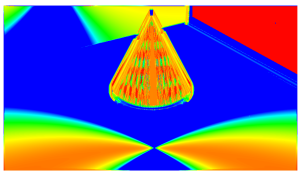
(a) Cone in the water



(b) Uncorrected depth map

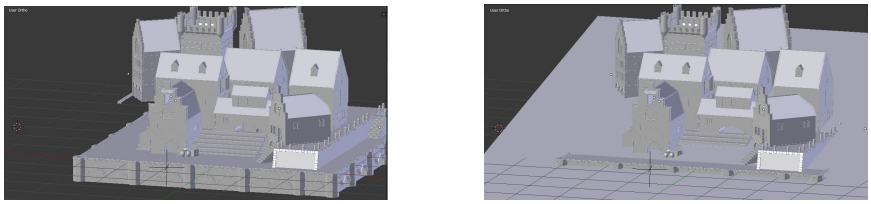


(c) Corrected depth map without refraction



(d) Depth map after water is removed in the synthetic scene

Figure 4.2: Figures a,b,c and d are results of the second scene, where a cone is submerged in a water tank. In Figure c, old underwater pixels are removed, leaving old positions white. Figure d is the depth estimation results using correspondences after removing the water in synthetic scenes.



(a) Blender model without water (b) Blender model with water

Figure 4.3: 3D model in the software Blender. Figure a is the semi-submerged object: a medieval city, which is provided by the benchmark dataset[3]. Figure b is the model after water is added. A cube with refractive material will refract rays just like real-world water.

material to represent water.

In Figure 4.3, we show the original 3D model and the 3D model with the refractive cube. In this experiment, a medieval city is semi-submerged in water. In particular, the front stairs are semi-submerged.

In Figure 4.4, we show the central view images in this synthetic data with and without water. As you can see from the images, this synthetic experiment is more challenging. There are some shadows and highlights on the water surface because of reflection of water.

In Figure 4.5, Figure a is the uncorrected depth map based on correspondences. Figure b is the corrected depth map after correcting the effect of refraction. Figure c is the ground truth depth map. It is exported from the blender file without the refractive cube.

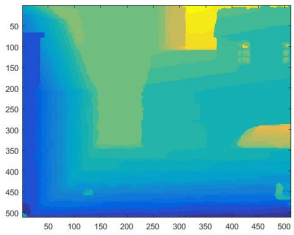
4.2 Real data

To work with a Lytro Illum camera, a light field MatLab toolbox [42] is used for the purpose of decoding and calibration. The decoded light field data is a 5D matrix. If we fix the 2 dimensions representing the aperture location on the main lens plane, then the squeezed new 3D data are the sub-aperture images we need. The size of decoded light field raw data is $15 \times 15 \times 434 \times 625 \times 4$. With a single capture, there are 15×15 sub-apertures in the raw data, which is also called angular resolution. The 434×625 represents the spatial resolution, in particular, the size of sensors to record densities of light rays.

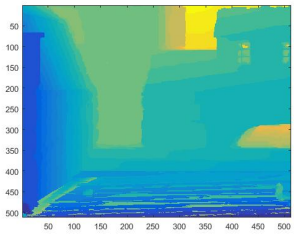
(a) Realistic Scene with water

(b) Realistic Scene without water

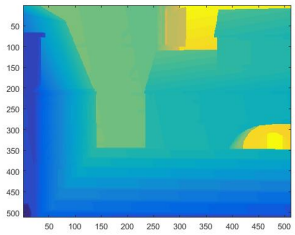
Figure 4.4: Central-view image from the synthetic data. Figure a is the image captured with water in the scene. Figure b is the image captured without water.



(a) Uncorrected depth map



(b) Corrected depth map without refraction



(c) Ground truth depth map without refraction

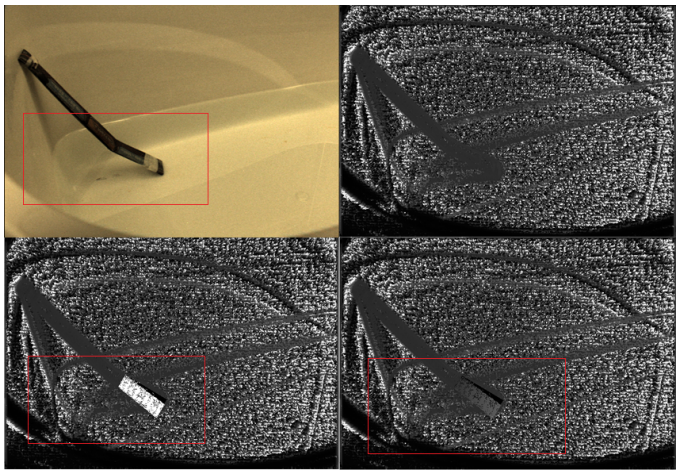
Figure 4.5: Uncorrected and corrected depth maps. Figure c is the ground depth map without water in the synthetic data.

Physically, the raw sensor pixels are hexagonally aligned on the sensor plane. Taken advantage of the vignetting effect, a white image taken by the Illum camera is able to help us detect the center of each sensor pixel. Finally, after the decoding process, raw pixels can be aligned within a grid, which is congruous with the 4D parametrization of a light field, i.e. a pair of coordinates (u,v) is given by the location in the micro-lens space, another pair (x,y) is given by the sub-aperture location on the main lens. This parametrization is equivalent to identify each ray's direction from the main lens to the micro lens. By now, it is notable that propagating the rays through the main lens, we can calibrate the rays to the camera coordinate system. From this toolbox, a calibration matrix, H matrix is formulated as:

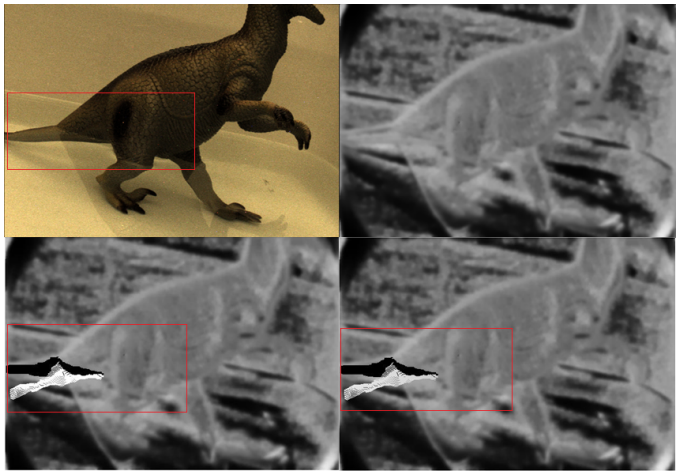
$$\begin{bmatrix} s \\ t \\ u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} H_{1,1} & 0 & H_{1,3} & 0 & H_{1,5} \\ 0 & H_{2,2} & H_{2,3} & 0 & H_{2,5} \\ H_{3,1} & 0 & H_{3,3} & 0 & H_{3,5} \\ 0 & H_{4,2} & H_{4,3} & 0 & H_{4,5} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \\ l \\ 1 \end{bmatrix}$$

In the real experiments, we only make use of 11×11 sub-apertures, in other words, the diameter we explored in the main lens plane is 11. Margin apertures are abandoned to reduce noise and to increase processing speed. Our results also indicates that the aperture size we used is reasonable. The whole processing time is about 25 minutes, 20 for depth estimation using correspondences, 5 mins for the next two steps. The CPU of our computer is an Intel 3.40GHz Core i7, with 16GB of memory.

The results of real experiments are shown in Figure 4.6. Since real data's texture is not as obvious as that of synthetic data, which make texture analysis invalid, we manually segment the underwater part. Because the distortion of the ruler scene is not so serious as the toy dinosaur's tail, its disconnection effect, also found in the synthetic data, is also not serious. Due to the concavities on the toy dinosaur's surface, its depth estimation result from correspondences is very noisy, resulting in some noise in the corrected depth map after removing the effect of refraction. In contrast, the corrected depth map of the ruler is much better. This is also the major limitation of our method.



(a) Real data: A ruler in a water tank



(b) Real data: A toy in a water tank

Figure 4.6: Real experiments' results: The upper row is the decoded light-field image captured with the Lytro camera and the corresponding depth map calculated using correspondence[4]. Because concavities on the surface of the toy dinosaur, the depth map using correspondences is very noisy. So we combine the defocus cue to improve it. The second row shows the shape comparison of original pixels (black) and corrected underwater pixels (white), and the correct depth after removing the effect of refraction. We only focus on the tail of the toy dinosaur, which is marked by a red box. The underwater distorted part is also marked in the ruler's image.

It depends on the virtual depth, the first depth estimation result to estimate ray's direction in air. If the scene is textureless or exhibits large displacement, then correspondences acquired with a consumer light field camera is not accurate, limited by its small baseline. In the future, we can employ other kinds of light-field cameras to overcome this constraint. Overall, the recovered shape

and depth of underwater part is closer to the ground truth after removing the effect of refraction.

Chapter 5

Conclusion

5.1 Summary

In summary, we propose a new method to remove the effect of refraction of a semi-submerged object using a light-field camera. With the corrected depth information and pixel positions, the real structure of underwater scene without distortions caused by refraction can be recovered. As far as we know, this is the first depth estimation algorithm to explicitly accommodate refraction using a consumer light field camera. By detecting boundary pixels on a semi-submerged object, we are able to estimate the water surface and apply it to recalculate the real positions of the underwater part. Although our method is affected by the result from Section 3.2, the next two steps targeting at refraction can be combined with more accurate depth estimation algorithms.

5.2 Contributions

The following summarizes the contributions of this thesis:

- Taking pictures of semi-submerged objects is very common in macro photography. After removing the effect of refraction, correct underwater depth information is beneficial for refocusing, scene reconstruction, view synthesis and many other applications. This thesis can serve as a staring point to incorporate refraction as an extension to the depth estimation problem.

- In our algorithm, we also estimate the equation for water surface. Although water is assumed to be stationary, the estimation result can be further applied as input to other water surface estimation algorithms, or underwater reconstruction algorithm.

- Our algorithm can work for dynamic scenes. Because only one capture is required from a light field camera.

- Our algorithm can work with light field data from other sources. As more and more light field data acquiring techniques emerging, more application of depth estimation using light field data can be found.

5.3 Future work

The following discussions give directions for future research:

- Narrow baseline of lenslet-based camera: The inherent narrow baseline of lenslet-based camera is the major reason why the depth estimation result is inaccurate. In order to handle this drawback, we should either estimate sub-pixels or use extra captures with a larger baseline. In the future, we want to lay emphasis on videos from light field camera to improve the accuracy of reconstruction. Recently, a new light field camera named Lytro Cinema has been released. With a video from a light field camera, it is interesting to combine the strength of a narrow baseline with a large baseline and to provide a real-time accurate depth map.

- Non-stationary water surface: When the water surface is no longer stationary, how to recover the normal of water surface according to the observed underwater distortion will be another interesting problem to explore. In our algorithm, a stationary water surface is assumed. In the future, we can optimize the reconstruction of semi-submerged objects and the water surface in a single framework.

- Combine depth cues: In our algorithm, only one depth cue, the correspondence is discovered to estimate the depth map. However, it is easily

affected by noise, repeating patterns and specularity. In the future, we want to combine correspondence with other depth cues, e.g. defocus, shading.

- Convolutional neural network: During the past several decades, convolutional neural network has made great contributions in many fields. Recent research work like [28][12] have demonstrated the feasibility of applying CNN to depth estimation using a light field camera. Even though the ground truth of depth is hard to retrieve in real life, the ground truth of views can help us train networks to synthesize new viewpoints. By using the epipolar geometry, the depth map can also be estimated indirectly.

Bibliography

[1] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[2] Temple H Fay and Porter G Webster. On runge's example for polynomial interpolation. *Problems, Resources, and Issues in Mathematics Undergraduate Studies*, 5(1):73–79, 1995.

[3] Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Asian Conference on Computer Vision*. Springer, 2016.

[4] Michael W Tao, Sunil Hadap, Jitendra Malik, and Ravi Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 673–680, 2013.

[5] Edward H Adelson and John YA Wang. Single lens stereo with a plenoptic camera. *IEEE transactions on pattern analysis and machine intelligence*, 14(2):99–106, 1992.

[6] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1996.

[7] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[8] Edward H Adelson and James R Bergen. The plenoptic function and the elements of early vision. 1991.

[9] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11):1–11, 2005.

[10] Ren Ng. *Digital light field photography*. PhD thesis, stanford university, 2006.

[11] Ting-Chun Wang, Manmohan Chandraker, Alexei A Efros, and Ravi Ramamoorthi. Svbrdf-invariant shape and reflectance estimation from light-field cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5451–5459, 2016.

[12] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. *arXiv preprint arXiv:1708.03292*, 2017.

[13] Michael W Tao, Pratul P Srinivasan, Jitendra Malik, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Depth from shading, defocus, and correspondence using light-field angular coherence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1940–1948, 2015.

[14] Stefan Heber, Rene Ranftl, and Thomas Pock. Variational shape from light field. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 66–79. Springer, 2013.

[15] Sven Wanner and Bastian Goldluecke. Globally consistent depth labeling of 4d light fields. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 41–48. IEEE, 2012.

[16] Sven Wanner and Bastian Goldluecke. Variational light field analysis for disparity estimation and super-resolution. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):606–619, 2014.

[17] Bastian Goldluecke and Sven Wanner. The variational structure of disparity and regularization of 4d light fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1003–1010, 2013.

[18] Michael Tao, Jong-Chyi Su, Ting-Chun Wang, Jitendra Malik, and Ravi Ramamoorthi. Depth estimation and specular removal for glossy surfaces using point and line consistency with light-field cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[19] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. Accurate depth map estimation from a lenslet light field camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1547–1555, 2015.

[20] Neel Joshi and C Lawrence Zitnick. Micro-baseline stereo. *Technical Report MSR-TR-2014–73*, page 8, 2014.

[21] Robert C Bolles, H Harlyn Baker, and David H Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1(1):7–55, 1987.

[22] Michael Bronstein, Jean Favre, Kai Hormann, et al. Epipolar plane image refocusing for improved depth estimation and occlusion handling. 2013.

[23] Sven Wanner and Bastian Goldluecke. Reconstructing reflective and transparent surfaces from epipolar plane images. In *German Conference on Pattern Recognition*, pages 1–10. Springer, 2013.

[24] Kaan Yucer, Changil Kim, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. Depth from gradients in dense light fields for object reconstruction. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 249–257. IEEE, 2016.

[25] Ting-Chun Wang, Manmohan Chandraker, Alexei Efros, and Ravi Ramamoorthi. SVBRDF-invariant shape and reflectance estimation from

light-field cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[26] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, and Manmohan Chandraker. Robust energy minimization for brdf-invariant shape from light fields. *IEEE PAMI*, 35(12):2941–2955, 2013.

[27] Michael W Tao, Ting-Chun Wang, Jitendra Malik, and Ravi Ramamoorthi. Depth estimation for glossy surfaces with light-field cameras. In *Proceedings of the IEEE European Conference on Computer Vision Workshops (ECCVW)*, 2014.

[28] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):193, 2016.

[29] Akira Shibata, Hiromitsu Fujii, Atsushi Yamashita, and Hajime Asama. Scale-reconstructable structure from motion using refraction with a single camera. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5239–5244. IEEE, 2015.

[30] Akira Shibata, Hiromitsu Fujii, Atsushi Yamashita, and Hajime Asama. Absolute scale structure from motion using a refractive plate. In *System Integration (SII), 2015 IEEE/SICE International Symposium on*, pages 540–545. IEEE, 2015.

[31] Zhihu Chen, Kwan-Yee K Wong, Yasuyuki Matsushita, Xiaolong Zhu, and Miaomiao Liu. Self-calibrating depth from refraction. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 635–642. IEEE, 2011.

[32] Yao-Jen Chang and Tsuhan Chen. Multi-view 3d reconstruction for scenes under the refractive plane with known vertical direction. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 351–358. IEEE, 2011.

[33] Anne Jordt, Kevin Köser, and Reinhard Koch. Refractive 3d reconstruction on underwater images. *Methods in Oceanography*, 15:90–113, 2016.

[34] Amit Agrawal, Srikumar Ramalingam, Yuichi Taguchi, and Visesh Chari. A theory of multi-layer flat refractive geometry. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3346–3353. IEEE, 2012.

[35] Nigel JW Morris and Kiriakos N Kutulakos. Dynamic refraction stereo. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1518–1531, 2011.

[36] James J Gibson. The perception of the visual world. 1950.

[37] Ruzena Bajcsy and Lawrence Lieberman. Texture gradient as a depth cue. *Computer Graphics and Image Processing*, 5(1):52–67, 1976.

[38] Andrew P Witkin. Recovering surface shape and orientation from texture. *Artificial intelligence*, 17(1-3):17–45, 1981.

[39] Paul A Warren and Pascal Mamassian. Recovery of surface pose from texture orientation statistics under perspective projection. *Biological cybernetics*, 103(3):199–212, 2010.

[40] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes in C*, volume 2. Cambridge university press Cambridge, 1996.

[41] Carl Runge. Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten. *Zeitschrift für Mathematik und Physik*, 46(224-243):20, 1901.

[42] Donald G Dansereau, Oscar Pizarro, and Stefan B Williams. Decoding, calibration and rectification for lenselet-based plenoptic cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1027–1034, 2013.

[43] Donald G Dansereau, Oscar Pizarro, and Stefan B Williams. Linear volumetric focus for light field cameras. *ACM Trans. Graph.*, 34(2):15–1, 2015.