Surface Reconstruction Based on Plenoptic Image with Convolution-Based Iterative Algorithm

by

Yimin Fu

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 6, 2016

Keywords: Light-field, deconvolution, distance estimation, 3D-volume reconstruction, iterative algorithm

Stanley J. Reeves, Professor of Electrical and Computer Engineering
Brian S. Thurow, W. Allen and Martha Reed Associate Professor of Aerospace Engineering
Thomas S. Denney Jr., Director, Auburn University MRI Research Center, Professor of Electrical and Computer Engineering

Abstract

Plenoptic imaging is a revolutionary photographic technique which has been brought to public awareness recently. It makes post-refocusing possible by capturing both spatial and directional information of light. In order to do that, a micro-lens is placed before the imaging plane of the camera. Each micro-lens records angular information from its location at the cost of spatial resolution. A focal stack is generated by stacking up refocused images of the same object. By adjusting the depth at which each image focuses, it models the light ray around the nominal focal plane of the camera in 3D. The main purpose of this thesis is to reconstruct the surface based on focal stacks. There were previous attempts to accomplish the same task, including the gradient method and the stereo method; however, they are unable to reconstruct a smooth object. Later research proposed a deconvolution method that can address the smooth object problem. The thesis demonstrates the limitation of the deconvolution method and proposes an iterative method to more precisely reconstruct the surface of the object. In order to implement the iterative algorithm, the thesis also mathematically models the process of image blur as the vectorized object multiplied with a convolution matrix which represents the point-spread function (PSF). The PSF describes the spread of light from the object across the entire focal stack. Various results of estimating depth from different methods are presented to compare the performance of each algorithm.

Acknowledgements

I would like to thank my family for their support of my decision to pursue a graduate degree abroad. From my country it's not an easy thing to pull off. I wouldn't be in this place without them loving me and backing me up. I'm really grateful for their sacrifice in exchange for me to pursue happiness and dream. It's such a great deed I can never appreciate it enough.

I would like to thank my advisor Dr. Stanley Reeves for his guidance. Dr. Stanley Reeves will always be that guy to steer me towards the right direction when I'm confused either about academics or life itself. I'm marveled by his dedication to research and education and the amount of effort he put into helping my graduate research. It's such a rewarding experience when I'm under his advisement for the past year.

Finally, I would like to thank Jeffrey Thomas Bolan for his development of the light-field toolkit. This thesis would not be possible without his talent.

Table of Content

List of Figures

Chapter 1

Introduction

A plenoptic camera is a camera that uses a micro-lens array to capture both directional and spatial information of light rays [13]. The information consists not only of the intensity of the light but also where it comes from and at what angle it comes through the main lens. So the user is able to trace the light ray to recreate refocused images at different depths with the help of a computer. Furthermore, the camera still holds promising potential yet to be fully developed. For example, it is possible to reconstruct the surface of the object in the picture by processing the focal stack. A focal stack is a set of images of the same scene stacked up in order of their focused depth. It is like a model of the light ray. The convergence of light and the difference of the depth become distinguishable after proper handling of the focal stack, and it has made the use of plenoptic cameras in measuring distance and 3D modeling a likely possibility.

This thesis is a continuation of the research 3D Surface Reconstruction Based On Plenoptic Image [23]. The deconvolution method models the blur in the focal stack as the object convolved with a PSF (point spread function), and the PSF is simply the focal stack of a point at the origin in the plenoptic image. By reversing the process—namely deconvolving the focal stack with this PSF—theoretically one is able to retrieve the surface information of the object. Initial experiments were conducted, and the result was compared to some of the traditional methods like gradient method and stereo method. The deconvolution method was on par with them and even better in some cases.

However, the deconvolution method failed to address the shift-variant nature in the focal stack. Because of the limitation of how far and how close one could refocus the image, there must be boundaries in the focal stack. The blur of an object not in the center will be truncated asymmetrically, thus creating shift-variance in the stack. This thesis proposes an iterative algorithm

to work around the shift-variance issue by only using convolution. By minimizing the error one hopes to get an accurate reconstruction of the object surface.

Chapter 2

Plenoptic Imaging

Plenoptic imaging is also known as light-field imaging is a type of photography that contrasts with a conventional camera, which only records the value of light deposited at certain spatial locations. A plenoptic image captures both the intensity and direction of the light ray coming through the lens, hence recording a light field. By using this light field one can easily refocus an image taken previously at a different depth with the help of a computer.

2.1 Previous Work

The concept of a plenoptic camera was envisioned by Leonardo da Vinci as early as the 16th century. He imagined such a device should capture every optical aspect of a scene. He believed the light is like radiant pyramids. If one is able to obtain all the information contained within, he/she could go along the pyramid to recreate any view from any point in a space. This versatility of light rays has been proven to be true. In 1908 the first light-field camera was proposed by Gabriel Lippmann. Lippmann's experiment included using a plastic sheet embossed with a regular array of micro-lenses to capture the directional information. Crude integral photographs were made. Despite having issues with stereo matching, his experiment still shed light on post-refocusing with a plenoptic image.

In 1992, Adelson and Wang proposed the design of a plenoptic camera which is almost the same as today's standard plenoptic camera: an array of micro-lenses is placed at the focal plane of the camera main lens. The image sensor is positioned slightly behind the micro-lenses. Using such images, the displacement of image parts that are not in focus can be analyzed and depth information can be extracted [18].

Into the 21st century the fully digitized photography industry has made the use of computers

in processing photos universal. The digital plenoptic image can be easily connected to a computer. Through the use of some simple software, users can trace the light rays backwards/forwards to refocus anywhere inside a photo.

2.2 Camera Structure and Fundamentals

There are two primary architectures concerning the placement of the micro-lens array. Plenoptic 1.0 places the micro-lens one focal length f from the sensor plane as opposed to Plenoptic 2.0, which places the array in such a way that it is focused at the sensor plane but not necessarily in the focal plane of the main lens. A comparison of two types of camera is in the figure 2.1. The two architectures basically represent the fundamental trade-off in plenoptic imaging between spatial resolution and directional resolution, where 1.0 has more directional resolution than 2.0 and vice versa. Only Plenoptic 1.0 is considered in this thesis for simplicity.

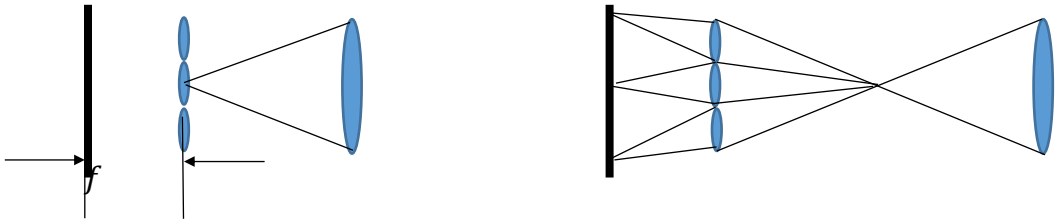

Figure 2.1: Plenoptic 1.0 (left) and Plenoptic 2.0 (right) [1]

The plenoptic system uses a 2-plane approach to characterize the light field with the first plane being the camera main lens in which the points on the aperture are given by coordinates (u, v) [10]. The second plane is the film or micro-lens plane in which the points are given by (x, y) coordinates. Combining the information from both planes we get a light ray with its direction specified by where it passes through the two planes and its intensity specified by parameter L. Hence a light ray is defined as $L(x, y, u, v)$ as a four-dimensional impulse [9].

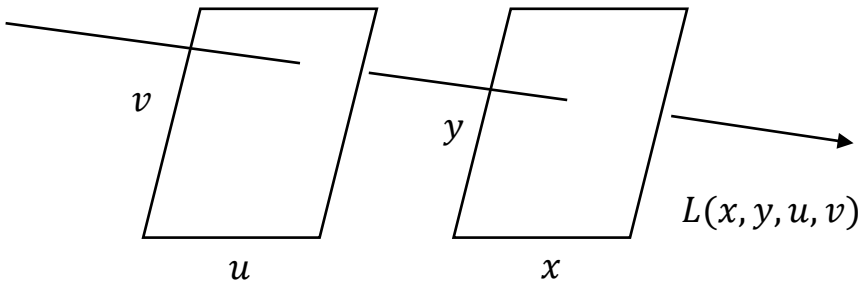

Figure 2.2: Using 2-plane approach to characterize light field

In reality the aperture itself does not specify at which location the light was coming through. In order to record the (u, v) coordinates in the film plane we use a micro-lens array. Each micro-lens acts like a viewpoint directing at the main lens gathering light coming from every (u, v) point on the main lens and spread them to the corresponding pixel behind the said micro-lens on the film plane. Data-fetching of plenoptic image works like this: Each pixel represents one light ray. The value of the pixel is the intensity of the light. The location of the micro-lens under which the pixel is located indicates (x, y). The micro-lens will bend the light rays coming through itself onto an area the shape of a lens on the film plane, and straightforwardly enough one can look at the pixel on the disc-shaped area to find the intensity coming from a specific (u, v) location of the main lens.

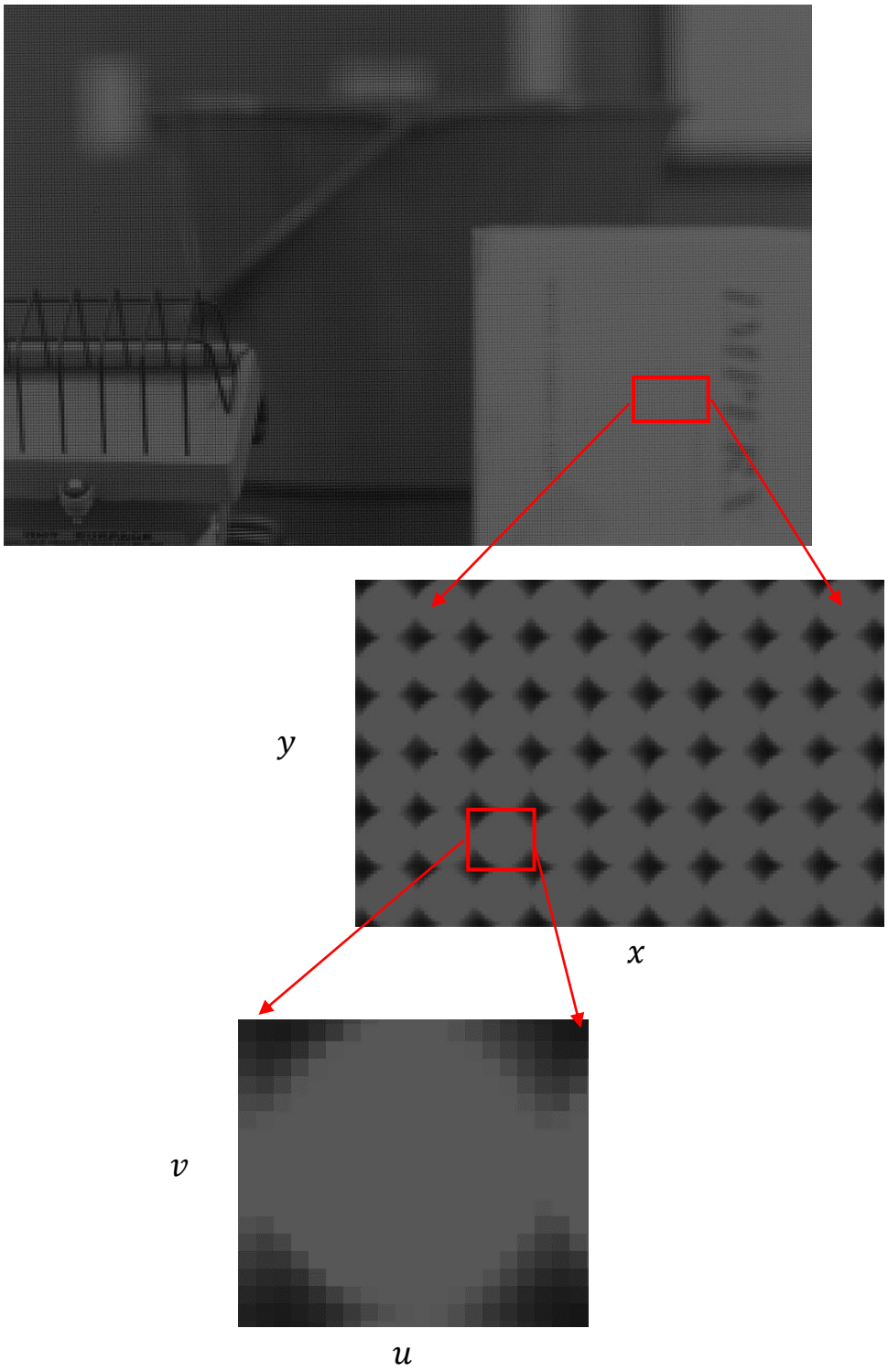

Figure 2.3: Capture of the 2-plane 4-dimensional data. Raw plenoptic image (top). Magnified plenoptic image with spatial information (middle). Further magnified plenoptic image with angular information (bottom).

As mentioned above, the size of the micro-lens is directly related to the directional resolution, i.e., the resolution of (u, v) coordinates. However, a too-big micro-lens will sacrifice spatial resolution as we only have as many (x, y) points as the number of micro-lens, and each (x, y) point will be translated into one pixel in the refocused image. Thus the spatial resolution will be worse than a conventional camera with the same lens and same film plane. Finding the point of balance in this directional-spatial tradeoff is crucial when designing a plenoptic camera for a certain purpose.

2.3 Digital Refocusing

Consider an object that is out of focus as in Figure 2.4. The object is placed at a distance less than the focal length of the camera. So, the light reflected by the object comes through the lens and converges onto a location behind the film plane. For a conventional camera only the light ray pattern as it intersects the film plane will be recorded, so we get a blurry image as the result of the energy of the light rays dispersed and overlapped with each other.

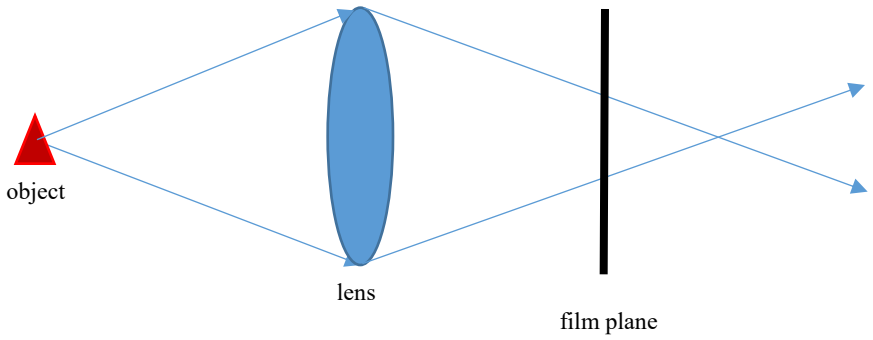

Figure 2.4: An object focused on the right side of the film plane (too close to the lens)

In a plenoptic camera, not only the displacement of the light ray where it intersects the film plane gets recorded but also the direction, as illustrated by the arrows in Figure 2.4. With the intensity and the direction of light rays known, we can deduce the behavior of the light field in a 3D space. By calculating both the location and intensity of the light at a virtual focal plane, the

out-of-focus object can be refocused as if we have moved the film plane backwards to meet the point of convergence of the light rays [15], as in Figure 2.4.

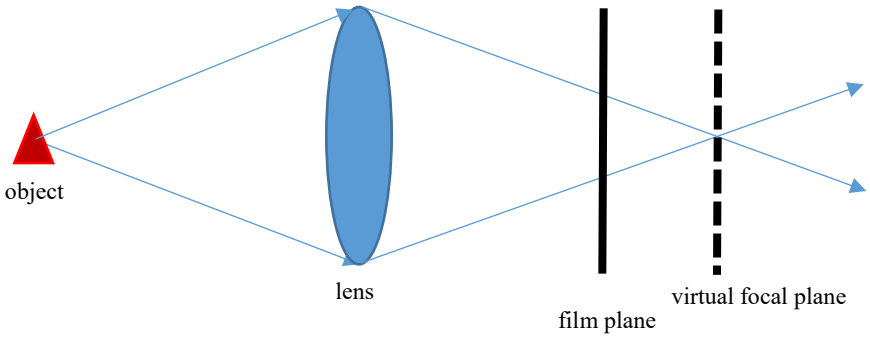

Figure 2.5: Tracing the light ray to the virtual focal plane of the object

2.3.1 Mathematical Model

After we obtain the light-field data $L(x, y, u, v)$, the irradiance of a given pixel (x, y) on the film plane is computed as follows:

$$E_F = \frac{1}{F^2} \iint L_F(x, y, u, v) du dv \tag{2.1}$$

where F is the distance between the camera aperture plane and the imaging plane. E_F is the irradiance at (x, y), calculated as the integration of all the light ray intersections with the film plane at (x, y) from all the points (u, v) on the main lens.

Since we already know the light-field $L_F(x, y, u, v)$ on the sensor plane, it is straightforward to trace the light geometrically onto a virtual plane using similar triangle theorem. Suppose we want to refocus the image at a plane at a distance F' from the main lens. As described in Figure 2.5, $L_F(x, y, u, v)$ on the film plane will become $L_F(u + (x' - u)\frac{F}{F'}, v + (y' - v)\frac{F}{F'}, u, v)$ on the virtual plane. Defining the ratio of the new focal plane to the old as $\alpha = F'/F$, the light field on a virtual plane F' from the main lens becomes:

$$L_{F'}(x', y', u, v) = L_F(u\left(1 - \frac{1}{\alpha}\right) + \frac{x'}{\alpha}, v\left(1 - \frac{1}{\alpha}\right) + \frac{y'}{\alpha}, u, v) \tag{2.2}$$

This equation formalizes the shear of the light field onto a different depth and lays the foundation for the refocusing algorithm. The focused depth of a refocused image will be referred to as relative depth α in this thesis.

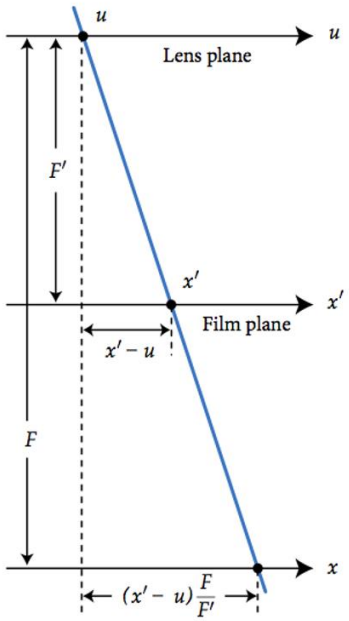

Figure 2.6: Tracing light-field onto another plane using similar triangle theorem [2]

2.3.2 Integral-based Refocusing

Section 2.3.1 derives the equation for calculating the light field at a specific depth. The light contributing to a pixel at a virtual plane is the set of light rays converging onto that location. If we already have the light field at the refocusing depth, Then the refocusing process simply involves integrating out the directional variables in $L(x, y, u, v)$:

$$E_{\alpha F}(x', y') = \frac{1}{\alpha^2 F^2} \iint L_F(u\left(1 - \frac{1}{\alpha}\right) + \frac{x'}{\alpha}, v\left(1 - \frac{1}{\alpha}\right) + \frac{y'}{\alpha}, u, v) du dv \tag{2.3}$$

Since early plenoptic research, researchers have been using this algorithm to generate accurate

pictures focused at different depth. The integral-based refocusing algorithm is reliable but does have one major flaw—the number of integrations required is equal to the number of pixels in the refocused image, whichcan be computationally intensive. It usually takes minutes to generated a refocused image with a standard resolution of 256×256.

2.3.3 FFT Method

Because of the aforementioned issue with integral-based algorithm, a new method to more efficiently refocus plenoptic image was proposed. To understand this method, we must introduce the Fourier-slice theorem [4]. The theorem relates a 2D image to the 4D plenoptic data as follows:

$$\boldsymbol{F^2} \circ \boldsymbol{I_2^4} \circ \boldsymbol{B_\alpha} = \boldsymbol{S_2^4} \circ \frac{\boldsymbol{B_\alpha^{-T}}}{|\boldsymbol{B_\alpha^{-T}}|} \circ \boldsymbol{F^4} \tag{2.4}$$

where $\boldsymbol{F^M}$ represents the M-dimensional Fourier transform, $\boldsymbol{I_M^N}$ represents the projection from M dimensions to N dimensions, $\boldsymbol{S_M^N}$ represents the slicing operation whereby the last N−M dimensions of a function are set to 0, and $\boldsymbol{B_\alpha}$ is the shearing operator. Here, $\boldsymbol{B_\alpha}$ is given by

$$\boldsymbol{B_\alpha} = \begin{pmatrix} \alpha & 0 & 1-\alpha & 0 \\ 0 & \alpha & 0 & 1-\alpha \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Let R denote the 4D FFT of the radiance array. By following the slice theorem the irradiance of a given pixel at a refocused plane can be computed by first taking a 2D slice of R in the direction denoted by the shearing operator and then doing an inverse 2D FFT on the slice:

$$E_{\alpha F} = \boldsymbol{F^{-2}} \circ \frac{\boldsymbol{S_2^4}}{F^2} \circ \boldsymbol{B_\alpha^{-T}} \circ \boldsymbol{F^4} \tag{2.5}$$

By implementing the FFT the algorithm greatly reduces the time for refocusing images, rendering them in a matter of seconds.

2.3.4 Focal Stack

A focal stack is a very important concept in this thesis. Namely, it is a stack of the refocused images of the same scene usually characterized by (x, y, z) coordinates with (x, y) representing the location on each image and z indexing the depth at which each refocused image is synthesized. There will be an α relating to each slice of the image.

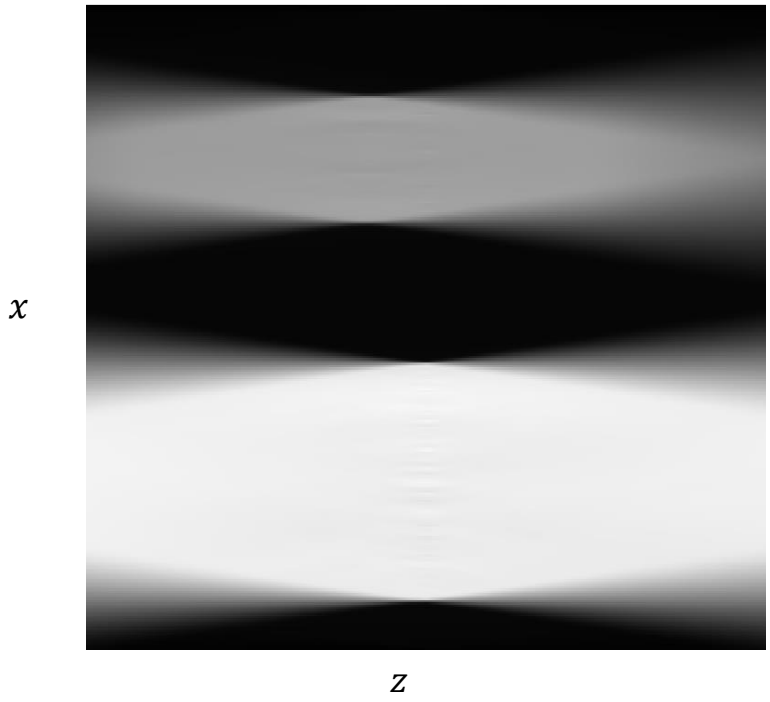

Figure 2.7: A focal stack viewed from (x, z) plane

As shown in Figure 2.7, the focal stack was generated from the plenoptic image of two pieces of white metal plate with different depths as evident by their respective set of light rays converging on different z locations. A focal stack records the behavior of light in 3D space: As light gets away from the plane of focus it radiates at different angles and its energy gets dispersed, thus creating the blurring effect in the defocused image [7]. With the blurring of objects modeled in the focal stack, it helps to further mathematically model the process of blurring so that this process can be reversed to retrieve the surface of the objects.

Figure 2.8: Raw plenoptic image of the focal stack in figure 2.5

2.3.5 Plenoptic Image Toolkit

In this research the generation of most of the refocused images and focal stacks use the Plenoptic Image Toolkit, which was primarily developed by Jeffrey Bolan, a graduate student in Aerospace Engineering at Auburn University. The toolkit incorporates an easy-to-use graphical user interface (GUI) and FFT refocusing algorithm, which greatly reduces the trouble from data fetching, calibration, and FFT refocusing.

As shown in Figure 2.5, the initialization GUI window requires you to select three directories. One is for a calibration image, which is the image for locating the position of each micro-lens. The second is for the plenoptic image you want to refocus. The last one is for the output refocused image. Also you need to input a few parameters mostly concerning the setting of the camera itself.

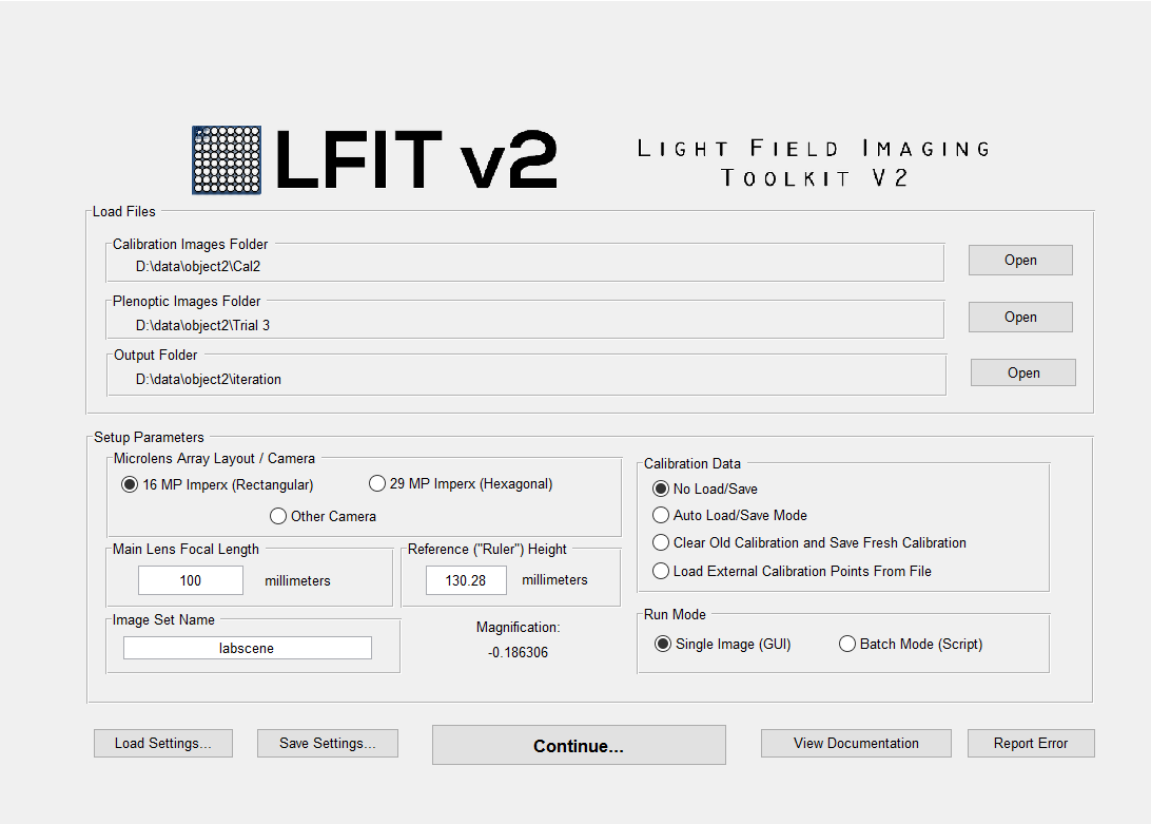

Figure 2.9: Initialization GUI window of plenoptic toolkit

After the initialization, the toolkit will prompt you to select three adjacent points in the calibration image for measuring the distance between micro-lenses to be able to sample angular information. In the final GUI window you must choose the α at which to refocus the image and then click Display and Save. Alternatively, you can choose to output a focal stack by specifying the range of α you want it to cover and the step size of α between adjacent slices in the stack.

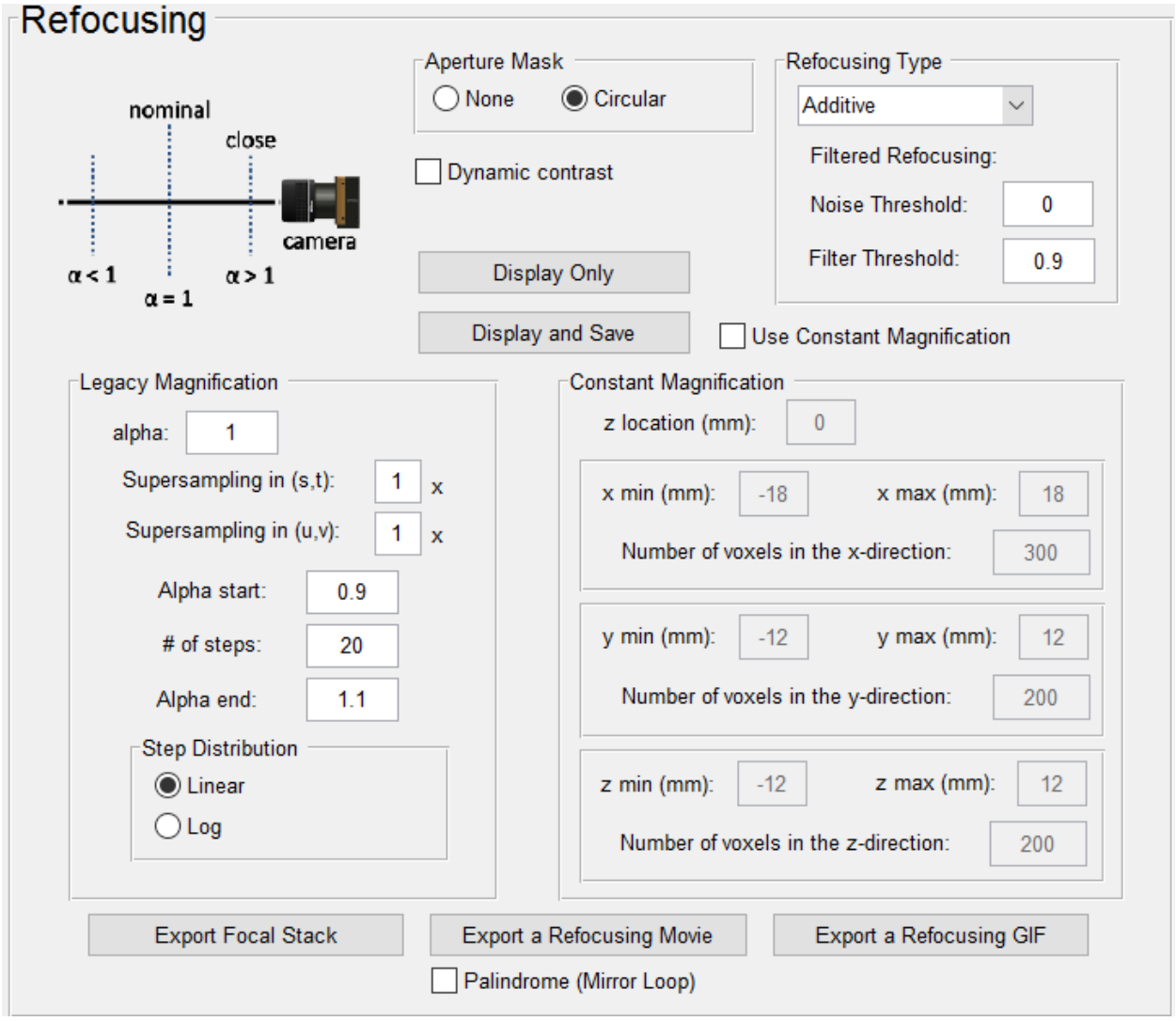

Figure 2.10: Refocus image and export focal stack in the final GUI window

Chapter 3

Surface Reconstruction

Surface information is stored in the focal stack as the alpha at which each slice is focused ranging from one extreme to the other. Assuming the alpha covers the full depth of the scene, every bit of surface will be in focus at one point in one of those slices. Finding the parts of the surface that are in focus and to which slice (and its depth) the in-focused surface belongs is the main task of surface reconstruction.

3.1 Gradient Method

The gradient method makes use of the fact that a focused image is sharper than a defocused one; thus the former will have more gradient energy. If we could find an operator to filter the focal stack slice by slice and quantize the high-frequency components of a given location, then we would be able to tell where that location is focused by finding the biggest high-frequency component in the slice-wise direction [11]. One of the most commonly used high-pass filters in image processing is the Laplacian operator, mostly because of its capability to detect gradients in multiple directions. Image blur disperses the object in a radial fashion so a Laplacian operator will be most suitable.

Figure 3.1: The focal stack in Figure 2.4 processed with Laplacian operator

After filtering with a Laplacian operator, we square every pixel in the focal stack to get absolute value, then convolve each slice with an $L \times L$ window for data consistency. The value of L represents a tradeoff: The smaller the window the more accurate depth estimation can be by finding the point with the biggest high-frequency components. But it does get inconsistent when L is too small. For example, when the window falls on a location right beside an edge of the focused image so the value of the according point in the focused image will be zero however it will not in a defocused one.

As can be seen in Figure 3.1, only the edges got recovered as the white lines in the picture. The area between the two white lines where there is supposed to be an object surface is dark. the gradient method cannot distinguish a smooth surface because there is no considerable transition of data in the surface area as in Figure 2.4, hence no gradient.

3.2 Stereo Algorithm

In order to implement the stereo algorithm [5] we start by introducing a sub-image generated from a raw plenoptic image. As mentioned above, the disc-shaped area behind each micro-lens represents angular information of light. The relative location of a point on the disc on the main lens is the angle at which the light came through. So if we only select the pixels with the same (u, v) under every micro-lens and combine them into one picture then we will have a sub-image.

This sub-image is also called a perspective image. As its name suggests, it provides one perspective of the object because every pixel has the same angular information. If we generate another sub-image with the set of pixels that have a different (u, v), then we will have two different perspectives of the same object. The stereo algorithm compares the difference between the two perspectives and estimates depth based on the fact that the depth of an object changes the perspective shift.

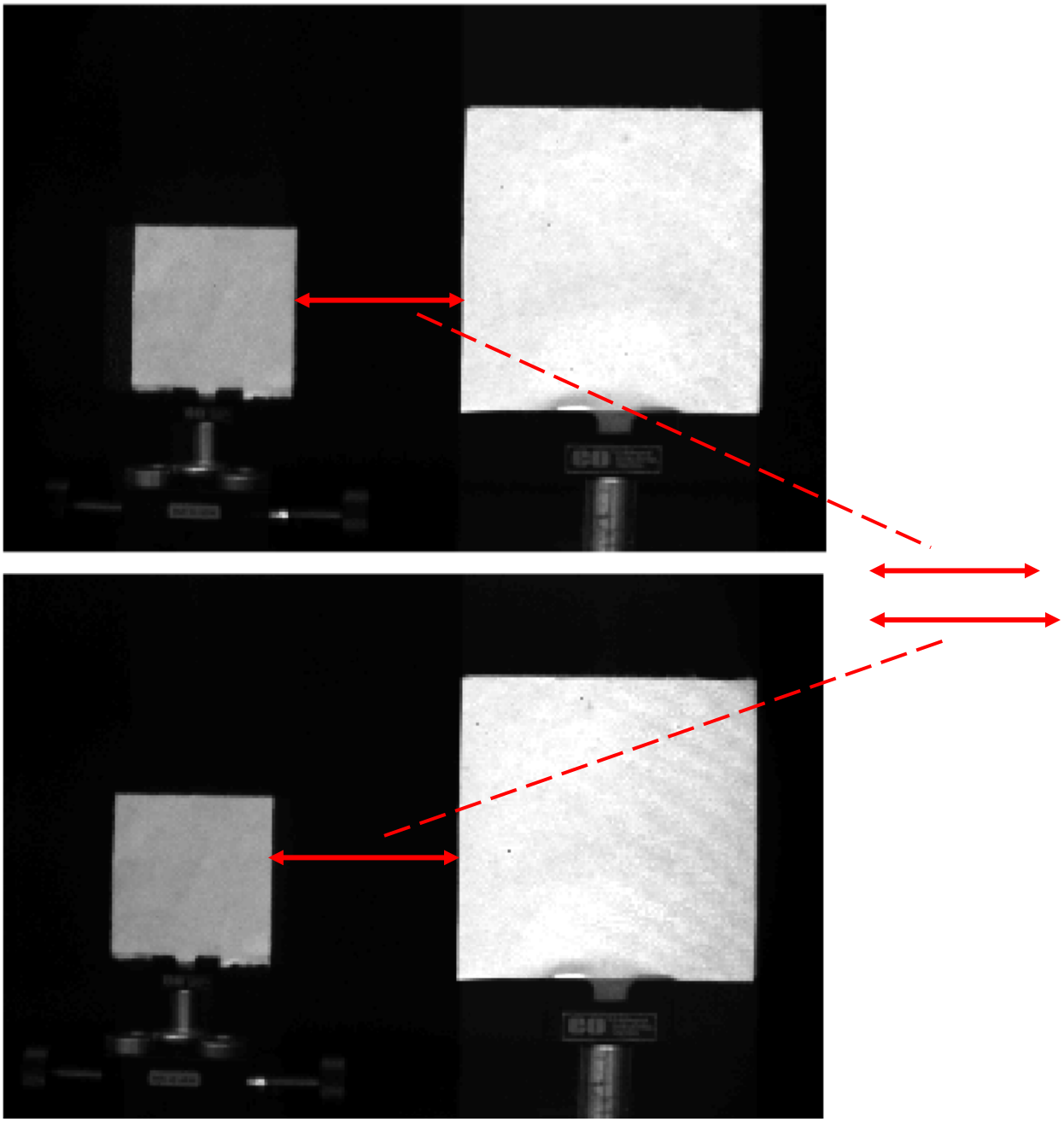

Figure 3.2: Two sub-images of the same plenoptic image with different perspectives. The top image is viewed from the right as opposed to the bottom image being viewed from the left.

Suppose we have a plenoptic camera with the focal length F and the diameter of main lens d known. As shown in Figure 3.3: To generate two perspective images, we consider the projection

of light from two different locations on the main lens through the object onto a virtual plane [12] with distance to the main lens unknown. We begin by placing an object in the focal plane of the main lens.

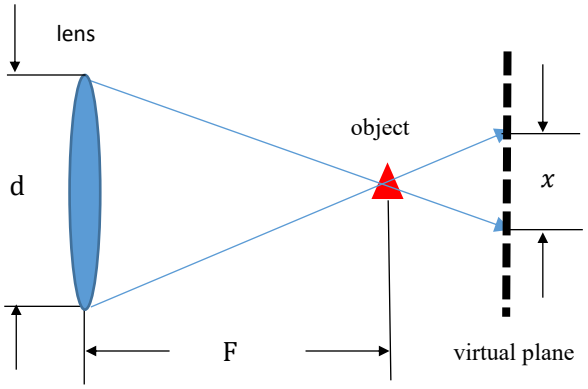

Figure 3.3: Generating two perspective images

Each arrow in Figure 3.3 represents the location of the object in each sub-image. We measure the disparity between the same object in two sub-images to get x [8]. The distance between the virtual plane and the main lens F_0 can be easily calculated using the similar triangle theorem as

$$F_0 = F(1 + \frac{x}{d}) \tag{3.1}$$

Next step we consider the images we want to measure which are taken by the same camera. We generate different perspective images (Figure 3.4) and measure the change of the location of the objects. The triangles at both sides of the object still satisfy the similar triangle theorem, and the depth D of the object can be calculated as follows:

$$D = F_0(\frac{d}{d+x'}) \tag{3.1}$$

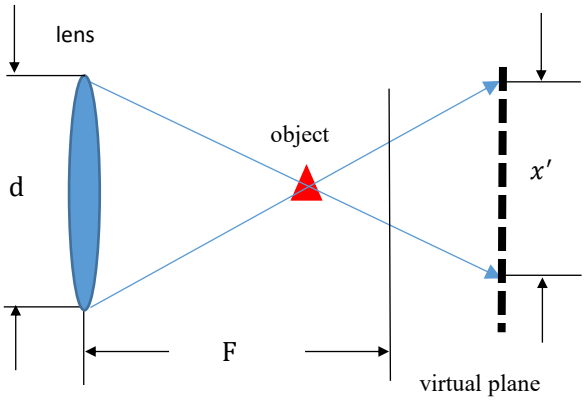

Figure 3.4: Generate two perspective images of a random object

The limitation of the stereo algorithm is that it requires the surface of the object to have some recognizable feature or pattern to make comparison viable. If the object is smooth and blank, as in Figure 3.2, then stereo algorithm may not be a good choice.

3.3 Deconvolution Method

Gradient and stereo methods both heavily rely on the object being characteristized by sharp edges and distinct patterns. The performance varies significantly on a case-by-case basis. Thus, we wish to develop an algorithm that works in every scenario and essentially reverses the blurring process.

3.3.1 Convolution Model

The deconvolution method treats the blur in the focal stack as the result of a surface convolving with a PSF. We define the blur in the focal stack as y, the original object or surface as x, and the PSF as h. The generation of focal stack can be written as [6]:

$$y = x * h \tag{3.1}$$

According to the convolution theorem, taking the Fourier transform of both side of (3.1) yields:

$$Y = XH \tag{3.2}$$

where the frequency arguments are suppressed for simplicity. So if H is known and Y is noiseless, x can be easily recovered by dividing Y by H and then taking an inverse FFT of X:

$$X = Y/H$$
$$x = ifft(X) \tag{3.3}$$

However, that's usually not the case. The process of imaging is hardly noiseless, and the PSF is often not perfectly matched with h. If such noise exists, doing deconvolution by Y/H will be undesirable due to the noise-amplifying issue [14]. Suppose $y = y_0 + n$, with n representing noise:

$$X = (Y_0 + N)/H = Y_0/H + N/H \tag{3.4}$$

The error term N/H can be quite huge if N is divided by a very small value in H.

A common way to deal with noise in deconvolution is to use a Wiener filter. A Wiener deconvolution is given by:

$$\hat{F}(\omega_x, \omega_y, \omega_z) = [\frac{H^*(\omega_x,\omega_y,\omega_z)}{|H^*(\omega_x,\omega_y,\omega_z)|^2+K}]G(\omega_x, \omega_y, \omega_z) \tag{3.5}$$

G represents the focal stack you want to deconvolve and H^* is the conjugate of PSF. K is a regularization term. Increasing the value of K will reduce the artifact and noise in the deconvolved image, but it makes the result fuzzier. A compromise has to be made between a sharp reconstruction and a less noisy one. In the figures below you can see the impact that different values of K have on the reconstructed object:

At a large regularization term (K>0.0001), there is no significant artifact due to the error. However, the reconstruction of surface is fuzzy and stretched across multiple slices. Attempt to increase the reconstruction accuracy by reducing regularization term will result in an increase of artifact as well as shown in the last figure in Figure 3.5.

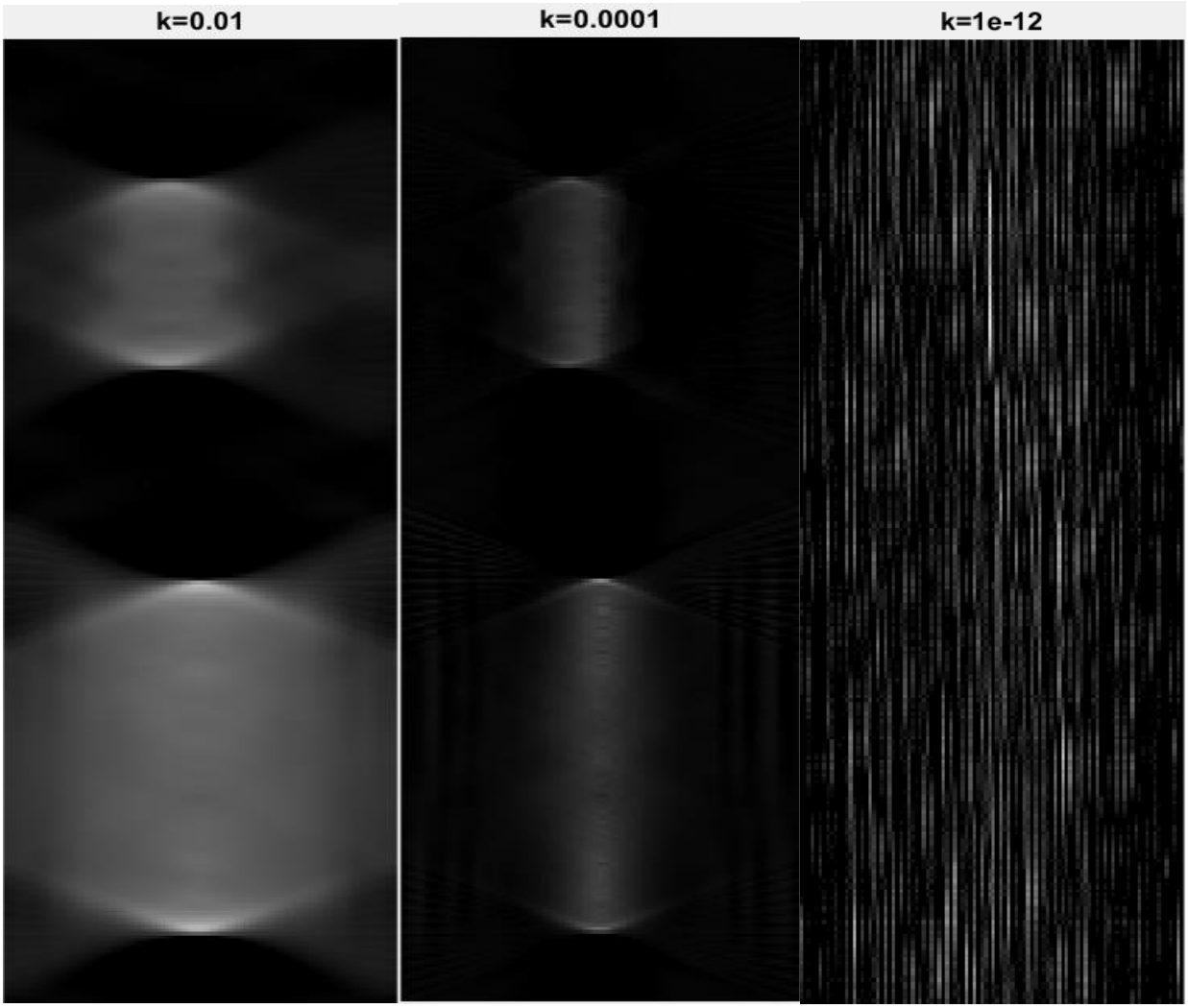

Figure 3.5: Reconstructed objects by Wiener filter with different values of K

3.3.2 Point Spread Function

The main reason behind finding a PSF is to mathematically model the process of image blur in the focal stack. This model can be used in both the deconvolution method and the iterative algorithm. So it is very important to find a PSF that accurately represents the spread of objects in the focal stack [17]. Refocusing basically does light-tracing, as it traces the set of light rays coming from one object. Therefore, it should have a symmetric spread being on the left or the right of the

focal plane as shown in Figure 3.4.

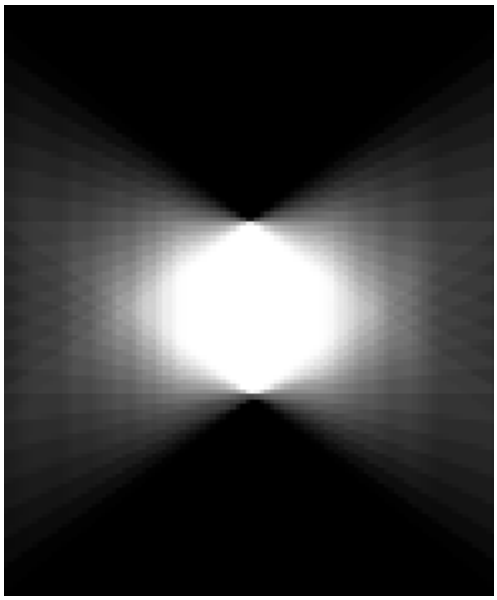

Figure 3.6: Spread of an object in focal stack

In theory this PSF should resemble a double symmetric cone with the apex being the focal point.

The PSF in nature is the system impulse response [16]. In term of the focal stack, a PSF can well be a focal stack generated from one point. Generally, there are two ways of creating a PSF, analytically and experimentally [3]. To analytically generate a PSF requires mathematically modeling the imaging system, which is quite complicated and hard to account for system error and different camera setup. Because of the availability of the easy-to-use plenoptic toolkit, one can treat the whole imaging system as a black box. The toolkit can compensate for all sorts of shift-invariant error in the refocused image. This thesis mostly concerns generating PSF experimentally.

Since the PSF is the system impulse response, instinctively, one would want to take a picture of just one pixel to generate a focal stack, but that is incorrect. Recall that the disc-shaped area in the plenoptic image translates to only one pixel in the refocused image, which is why they are called virtual pixels. In short, a micro-lens is the smallest unit in a focal stack. So in order to generate a correct PSF it is necessary to light up all the area behind one micro-lens or alternatively

light up all the area that is sampled by the plenoptic toolkit, which is normally slightly smaller than the micro-lens. Failure to do so will result in the mismatch of the PSF as it is missing part of the angular information. A comparison of different PSFs generated from lighting up different areas in the raw plenoptic image can be seen in Figure 3.7.

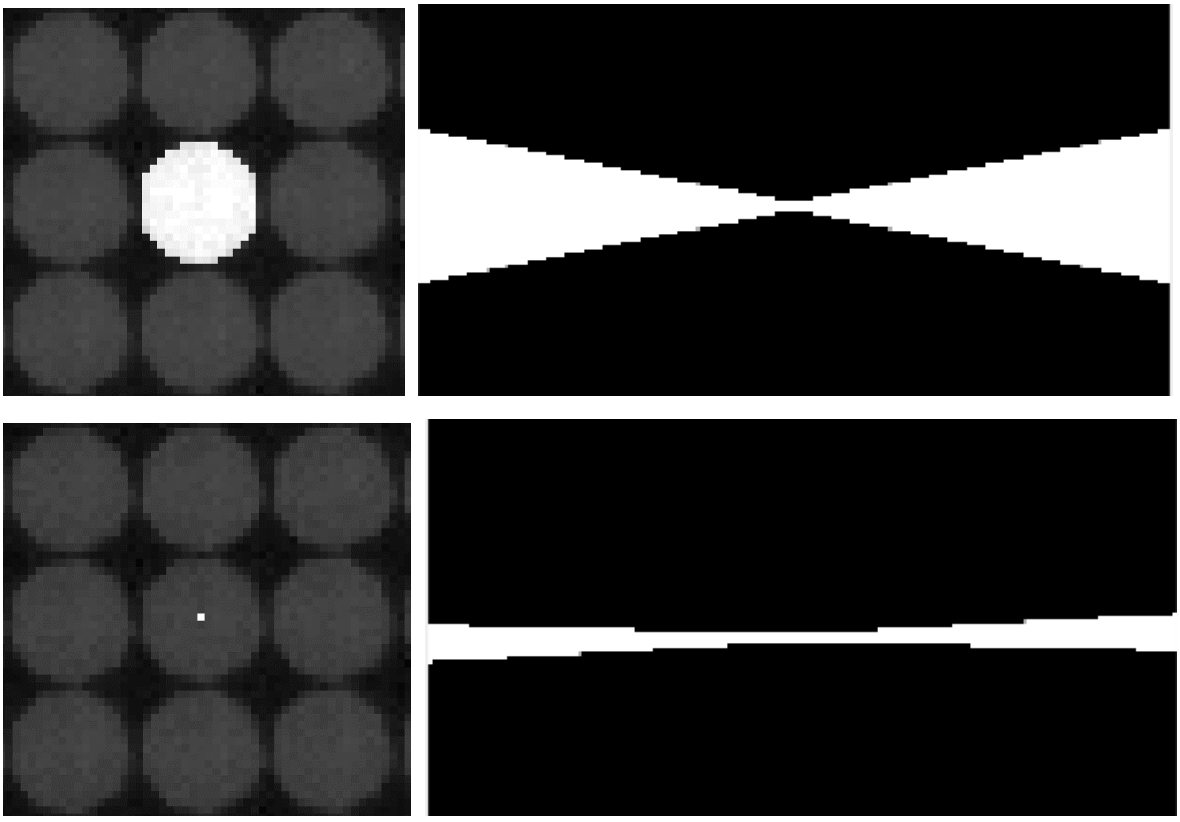

Figure 3.7: PSF generated with full angular coverage (top). PSF generated from one pixel (bottom). (all PSFs in this thesis are displayed in the log domain for better visualization).

Another thing to note is the sampling rate. In the plenoptic toolkit there is a setting about the angular sampling rate which relates to how many units of (u, v) points get sampled per micro-lens. Increasing the angular sampling rate will increase the quality of the refocused image at the cost of time simply because as the number of (u, v) points increases the software will take into account more light rays. As with the default angular sampling rate with a multiplier of 1, a PSF at the end of the focal stack will appear sparse and contain a lot of dark areas. Because the refocusing

traces a small set of light rays that are collected by just one micro-lens. As the ray-tracing goes far away from the focal plane, the light rays will appear few and far between when intersecting with the virtual plane. By increasing the aforementioned setting, when there are more light rays to disperse, the PSF will retain its consistency further out in the focal stack. PSFs with different angular sampling rates can be seen in Figure 3.8.

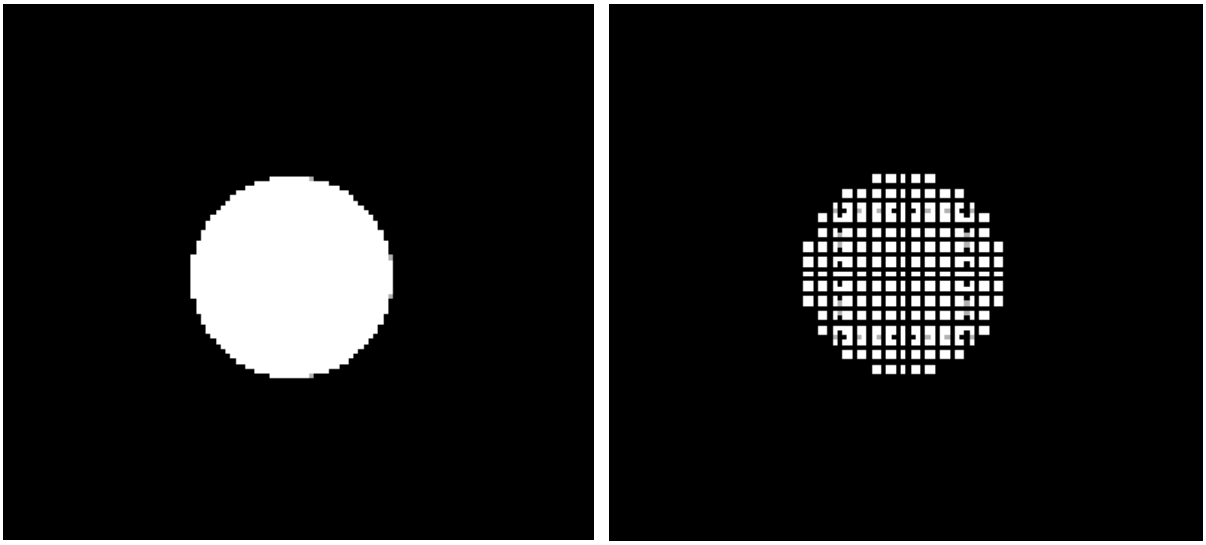

Figure 3.8: PSFs generated with different angular sampling rates. Super-sampling with a multiplier of 3 (left). Default sampling rate (right). viewed from (x, y) plane at $\alpha = 1.2$.

Although a super-sampled PSF may be good for demonstration, that's not what we need. A normal refocused plenoptic image with the default sampling rate looks sufficiently good. Different sampling rates between the object stack and the PSF will result in huge mismatches. So there is really no need to super-sample in (u, v) coordinates.

3.4 Iterative Algorithm

As mentioned previously, deconvolution cannot solve the shift-variance issue in the focal stack. As long as we are given an object that is not in the center of the focal stack (which usually is the nominal focal plane $(\alpha = 1)$ of the main lens), we can expect a lot of artifacts after the deconvolution. To generate a focal stack from an object away from the focal plane is equivalent to convolving that object with an asymmetric PSF. But strict deconvolution requires that we deconvolve that stack with a symmetric PSF. The algorithm makes up for the mismatch by creating negative values and converging energy onto the center as illustrated in Figure 3.9.

However, if we could manage to find the actual asymmetric PSF, the objects can be reconstructed correctly without the artifact. But this would defeat the purpose of this research (to find surface depth), because you need to know the depth of the object first to calculate how you should modify the shape of the PSF correspondently. Besides, it will not work on an image with multiple objects or an object with an uneven surface. In conclusion, the convolution describing the generation of a focal stack is basically irreversible by a direct form. An algorithm that only involves convolution is needed for solving this problem. Iterative algorithms are very good examples of using an initial guess to generate successive approximations to solve shift-variant problem without formalizing an inverse function.

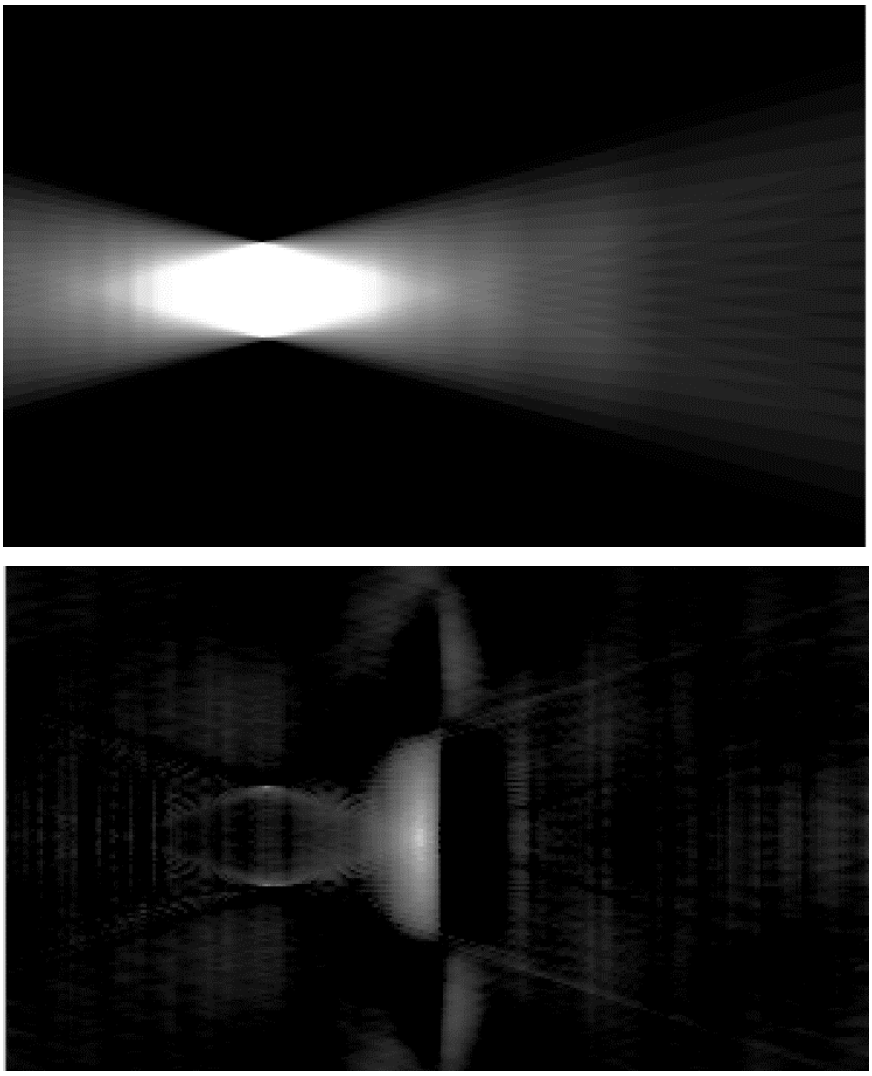

Figure 3.9: An asymmetric focal stack as the object is not on the nominal focal plane of the camera (top). artifact results from the mismatch (bottom).

3.4.1 Mathematical Model for Iterative Algorithm

If the data transformation (in this case: generating a focal stack) can be put into the form $f(x) = y$ and x is an unknown fixed point of function f, then one may begin with a point x_k and by successively updating its value minimize the difference between $f(x)$ and $f(x_k)$. Through a series of well-conditioned iterations, it is possible to arrive at a solution that is almost

the same as x.

Form $f(x) = y$ implies that y is the observed focal stack; x is the object/surface we want to recover; f stands for the process of blurring and truncation. In the toolkit as we select the range of α we want it to cover in the focal stack, that α range essentially decides how far the object is going to spread before it is truncated by the boundary. In order to implement truncation in f we start by doing a linear convolution of x and the PSF, followed by truncating the result back to the size of x. Note that because we want the PSF to cover all the blur in the focal stack, consider the extreme case: When an object is on the boundary of the focal stack, to stretch it across the entire α range, the PSF needs to be twice as large as y in the z direction.

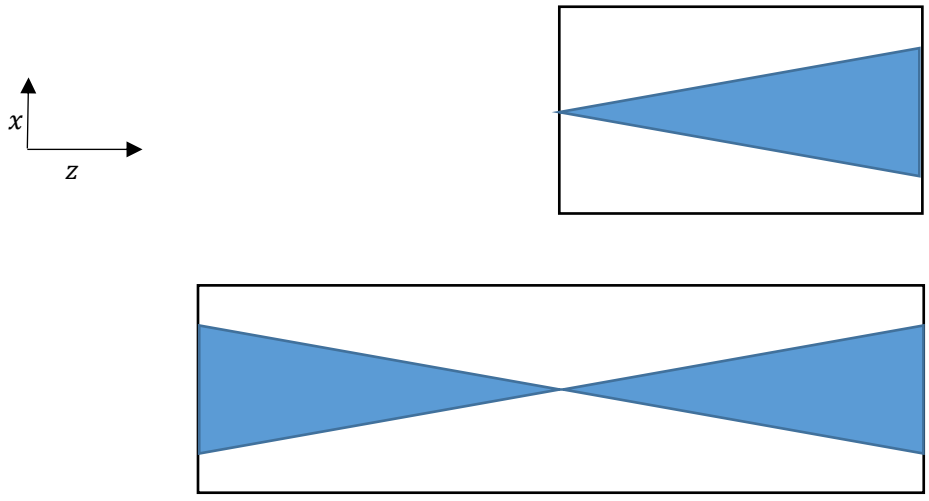

Figure 3.10: observed focal stack (viewed from (x, z) plane) with the object focused at the left boundary (top). PSF needed to cover all the blur (bottom).

As we are implementing linear convolution with FFTs in Matlab, there is a series of padding involved to avoid overlap due to circular convolution. Here are the steps for calculating $f(x_k)$ with K being the number of iterations:

1. Make sure x_k has the same size as y.
2. Pad x_k in the z direction to the size of PSF.
3. Pad both x_k and PSF in the x and y direction to leave enough margin to contain

circularly extended blur generated by circular convolution which will be soon truncated.

4. Perform circular convolution of x_k and PSF by FFT.
5. Truncate the result back to the size of y.

Consider the extreme case when convolving the point in the top right corner of x_k with the PSF in Figure 3.10. The circularly extended blur is illustrated by the yellows areas. The object space is illustrated by the red rectangle while the outer dash line rectangle representing the minimum padding required to contain all the circularly extended blur after the circular convolution.

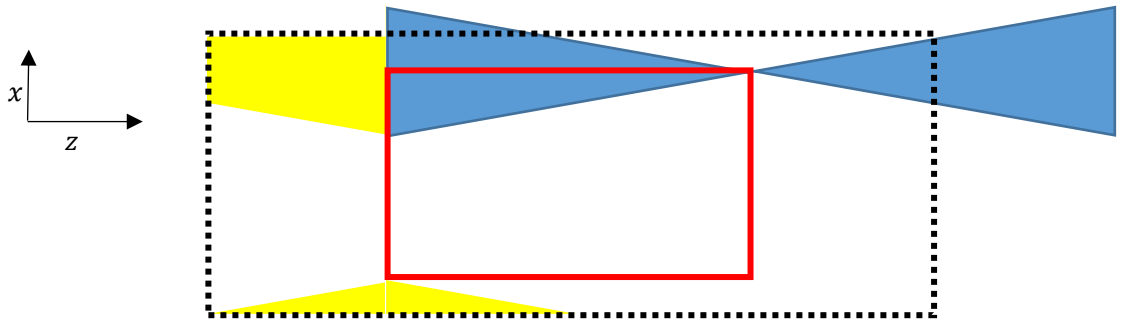

Figure 3.11: Avoiding the circularly extended blur into the object space by padding

3.4.2 Steepest Descent

From the last section we learned how to model the process of generating a focal stack, and the remaining piece is to find an optimization algorithm to minimize the error term $||y - f(x_k)||$ iteratively.

Steepest descent—also known as gradient descent—is a first-order minimization algorithm. It finds the local minimum of a function by taking steps proportional to the negative of the gradient [21]. The function we want minimize is given by $||y - f(x_k)||$. If we can find its gradient $\nabla||y - f(x_k)||$, then change x_k to $x_{k+1} = x_k - \beta \times \nabla||y - f(x_k)||$, if β is small enough, it is guaranteed that $||y - f(x_k)|| \geq ||y - f(x_{k+1})||$. As we continue to update the value of x_k this way, hopefully x_k will arrive at a location where $||y - f(x_k)|| \approx 0$ and x_k will closely

resemble the object x we are looking for.

The question remains, what is $\nabla||y - f(x_k)||$? It is tricky to calculate the gradient because f is shift-variant. But luckily we can include the convolution and the truncation into one single matrix H, namely making $||y - f(x_k)|| = ||y - H\vec{x}||$. Taking the gradient of $||y - H\vec{x}||^2$ gives us $-2H^T(\vec{y} - H\vec{x}_k)$. As long as H^T exists we can calculate ∇f pretty easily. The question remains how to define matrix H. Consider a very simple example. Suppose we want to use matrix convolution to calculate $[1\ \ 1\ \ 1] * [1\ \ 1]$. The convolution is given by:

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

As you can see, each row of the convolution matrix is responsible for one point in the result. You are free to change how you arrange the order of rows in the matrix as long as you can map the resulting vector accordingly back to its original dimensions. Truncation wants to keep the size of the result the same as x. To implement that in H we need to get rid of the rows which will be creating points outside of x, essentially making H contain as many rows as the number of points in x. Because the number of columns in a convolution matrix is equal to the number of points of the object you want to convolve with, H will be a square matrix with equal rows and columns.

The whole discussion about H is not for analytically creating a convolution matrix that does the same thing as f. Quite the contrary, we need not know what H is as long as it has been proven to exist. Consider the original problem again: Calculate $\nabla||y - f(x_k)||$ which is equivalent to $\nabla||y - f(x_k)||^2$in term of minimizing the function again equivalent to $\nabla||\vec{y} - H\vec{x}_k||^2$ which is equal to $-2H^T(\vec{y} - H\vec{x}_k)$. $H\vec{x}_k$ can be implemented by convolving x_k with the PSF and truncating the result. H^T is just the convolution matrix of what H represents but with flipped (negated) coordinates. In this case H^T will be implemented exactly the same as H because PSF in theory is symmetric. So the final iterative form with steepest descent is given by:

$$\vec{x}_{k+1} = \vec{x}_k + \ \beta[H^T(\vec{y} - H\vec{x}_k)] \tag{3.6}$$

K represents the number of iteration, and β is the step size. Since we are dealing with real-world intensity images, there will be no negative value. Thus, it is helpful to further condition the iteration by using a nonnegativity constraint. The modified iterative form with steepest descent is given by:

$$\vec{x}_{k+1} = C\{\vec{x}_k + \beta[H^T(\vec{y} - H\vec{x}_k)]\} \tag{3.7}$$

$C\{\}$ sets every negative value to zero.

3.4.3 Conjugate Gradients

Steepest descent searches the local minimum by going along the direction of local downhill gradient ∇f. The problem with that is that usually the negative gradient will not point right towards the minimum. This results in steepest descent going down a long narrow "valley", taking many shifts and turns before finally reaching the destination. In Figure 3.9 you can see how inefficient doing line minimization with steepest decent can be:

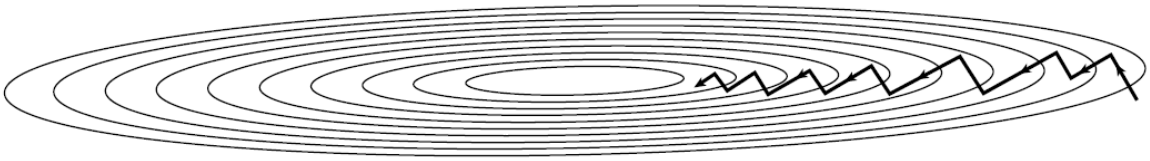

Figure 3.12: Convergence of steepest decent [4]

Figure 3.9 shows the scenario with 2D minimization. However, we are dealing with a massive multidimensional problem to minimize $||y - f(x_k)||$, which has as many dimensions as the number of pixels in the focal stack. The process of iteration is going to be very time-consuming. We need to find a better algorithm with faster convergence speed. One candidate is conjugate gradients.

Conjugate gradients minimizes the error term by proceeding not down the gradient but rather

in a direction constructed to be orthogonal to the old gradient in the range space and in so far as possible to all the previous gradients in that space. By making the directions of line minimization conjugate to one another, the algorithm is able to maintain the line search direction orthogonal to the previous gradient direction when it starts with a new direction. In other words, it will not spoil or redo the minimization obtained from the previous iterations [19], and that makes the algorithm much more efficient than steepest descent.

To implement the conjugate gradient method, first we need to put the function into quadratic form [22]:

$$f(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{x}^T \boldsymbol{b} \tag{3.8}$$

where A is an SPD (symmetric positive definite) matrix. With that form the gradient of the function can be calculated as:

$$\boldsymbol{\nabla} f(\vec{x}) = A\vec{x} - \boldsymbol{b} \tag{3.9}$$

The change of gradient as we move along some direction can be further calculated as:

$$\delta(\boldsymbol{\nabla} f) = A \cdot \delta(\vec{x}) \tag{3.10}$$

Suppose we have moved along direction $\vec{u}$, to ensure that gradient stays orthogonal to the next direction $\vec{v}$, we need:

$$0 = \vec{u} \cdot \delta(\boldsymbol{\nabla} f) = \vec{u} \cdot A \cdot \vec{v} \tag{3.11}$$

If (3.11) holds true for $\vec{u}$ and $\vec{v}$, they are said to be conjugate. A conjugate set is a set of vectors satisfy (3.11) pairwise. If line minimization is performed along the direction of a conjugate set, then you don't need to redo any of those directions.

For our problem, function $||y - f(x_k)||$ is equivalent to $||y - f(x_k)||^2$ gradient-wise. Recall that this function is implemented as $||\vec{y} - H\vec{x}_k||^2$, and $||\vec{y} - H\vec{x}_k||^2$ is equal to $(\vec{y}^T -$

$\vec{x}_k{}^T H^T)(\vec{y} - H\vec{x}_k)$. Therefore, the SPD in form (3.9) should be $H^T H$ regardless of the rest of the quadratic form.

We begin by setting two initial vector $\vec{r}_0$ and $\vec{p}_0$ equal to the gradient of $||y - f(x_k)||$. The conjugate gradient method then constructs two sequences of vectors from the recurrence:

$$\vec{r}_{k+1} = \vec{r}_k - \lambda_k H^T H \vec{p}_k \qquad \vec{p}_{k+1} = \vec{r}_{k+1} + \gamma_k \vec{p}_k \qquad k = 0, 1, 2 \ldots$$

Those vectors will satisfy the orthogonality and conjugacy conditions:

$$\vec{r}_i \cdot \vec{r}_j = 0 \qquad \vec{p}_i^T \cdot H^T H \cdot \vec{p}_j = 0 \qquad \vec{r}_i \cdot \vec{p}_j = 0 \qquad j < i$$

Scalars λ_k, γ_k are given by

$$\lambda_k = \frac{\vec{r}_k{}^T \cdot \vec{r}_k}{||H\vec{p}_k||^2} \qquad \gamma_k = \frac{\vec{r}_{k+1}{}^T \cdot \vec{r}_{k+1}}{\vec{r}_k{}^T \cdot \vec{r}_k}$$

If we follow the direction of $\vec{p}$ successively with step size set to equal λ_k [21]. After N (number of dimensions) iterations we would exhaust all the "non-interfering" directions and efficiently arrive at the minimum of the function [20]. For example: A 2-dimensional line minimization problem can be solved by conjugate gradients with optimal step size in just 2 iterations.

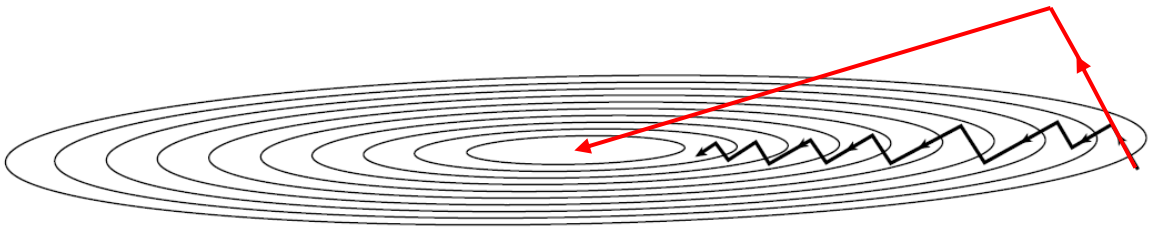

Figure 3.13: Comparison of the convergence of steepest decent (in black) and conjugate gradient (in red)

Chapter 4

Results from Surface Reconstruction

In terms of surface reconstruction, the gradient method and stereo method have problems in smooth areas, which accounts for most of objects in one image. Furthermore, these have been discussed and analyzed in full detail in a recent thesis[1]. Therefore, this chapter mainly focuses on showcasing results from deconvolution and iterative algorithms and demonstrates how much iterative algorithm can improve the results.

Considering the computationally intensive nature of iterative algorithms, it is best to start with a very simple or synthesized focal stack to check if it actually works in theory and work our way up to more complicated focal stacks generated from real data.

4.1 Simple Synthesized Focal Stack

Figure 4.1 shows a synthesized focal stack with a size of $31\times31\times41$, which is much smaller than a conventional focal stack generated from real data. A real focal stack usually contains tens of millions of pixels, and the computation time for iterative algorithm could rise dramatically.

[1]Haiqiao Zhang 3D Surface Reconstruction Based On Plenoptic Image

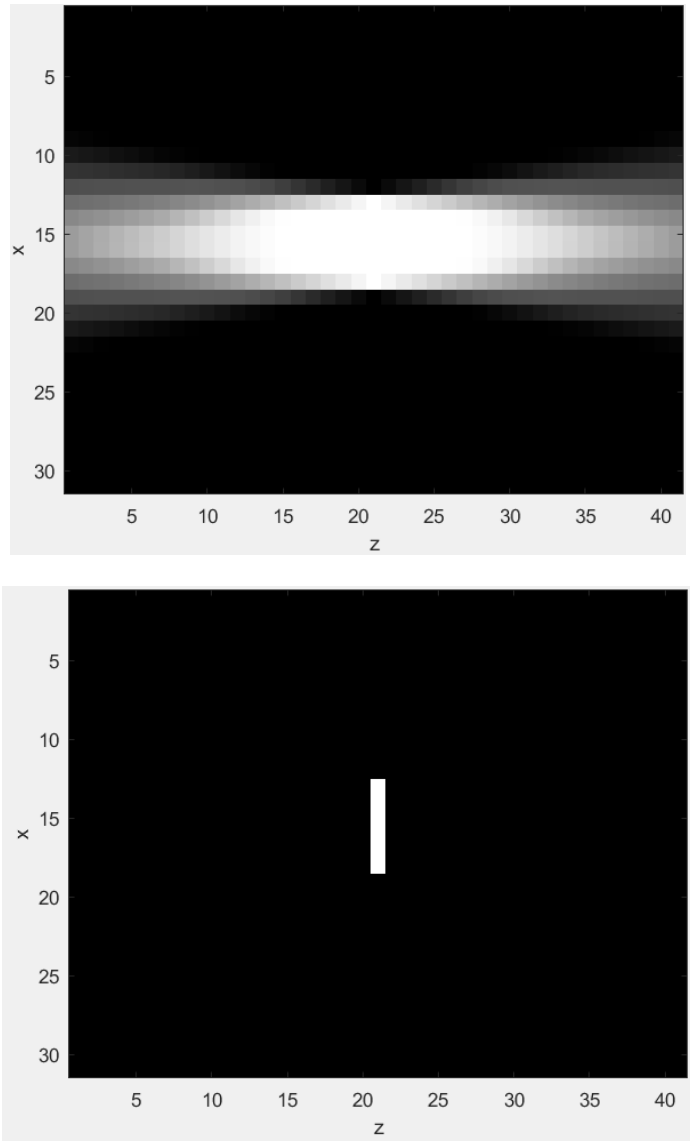

Figure 4.1: Synthesized focal stack (top). Theoretical reconstruction (bottom)

It was generated by lighting up a very small area in a plenoptic image, then using the toolkit to export a focal stack and truncate the size accordingly. Because the area lighted up was smooth and uniform it should represent a surface that is perfectly focused on the nominal focal plane. The reconstructed surface should be in the center of the focal stack as indicated by the bottom figure in Figure 4.1.

4.1.1 Deconvolution Method

First we reconstruct the object using the deconvolution method:

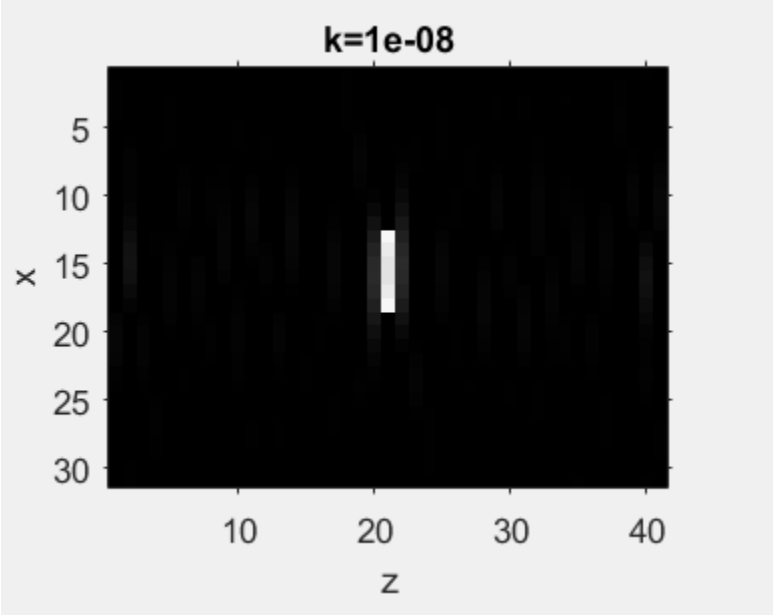

Figure 4.2: Surface reconstruction by deconvolution (synthesized focal stack)

As Figure 4.2 suggests, deconvolution works well as long as the object featured in the focal stack is on the nominal focal plane. What if it is not? Figure 4.3 depicts a synthesized focal stack of an object that is focused on the left side of the nominal focal plane.

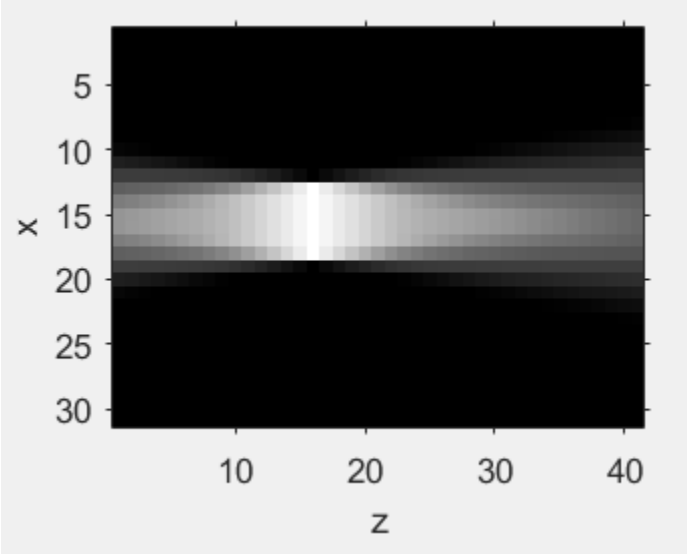

Figure 4.3: Asymmetric synthesized focal stack

This focal stack was generated from the same plenoptic images from Figure 4.1. It was focused at slice $z = 16$. The shift in focal stack was done by changing the range of α to be not symmetric about the nominal focal plane ($\alpha = 1$). It is equivalent to moving the main lens away from the object, but this approach is much more convenient. The focal stack was then deconvolved by two PSFs— one the general symmetric PSF and the other the customized PSF to match the asymmetry in the stack.

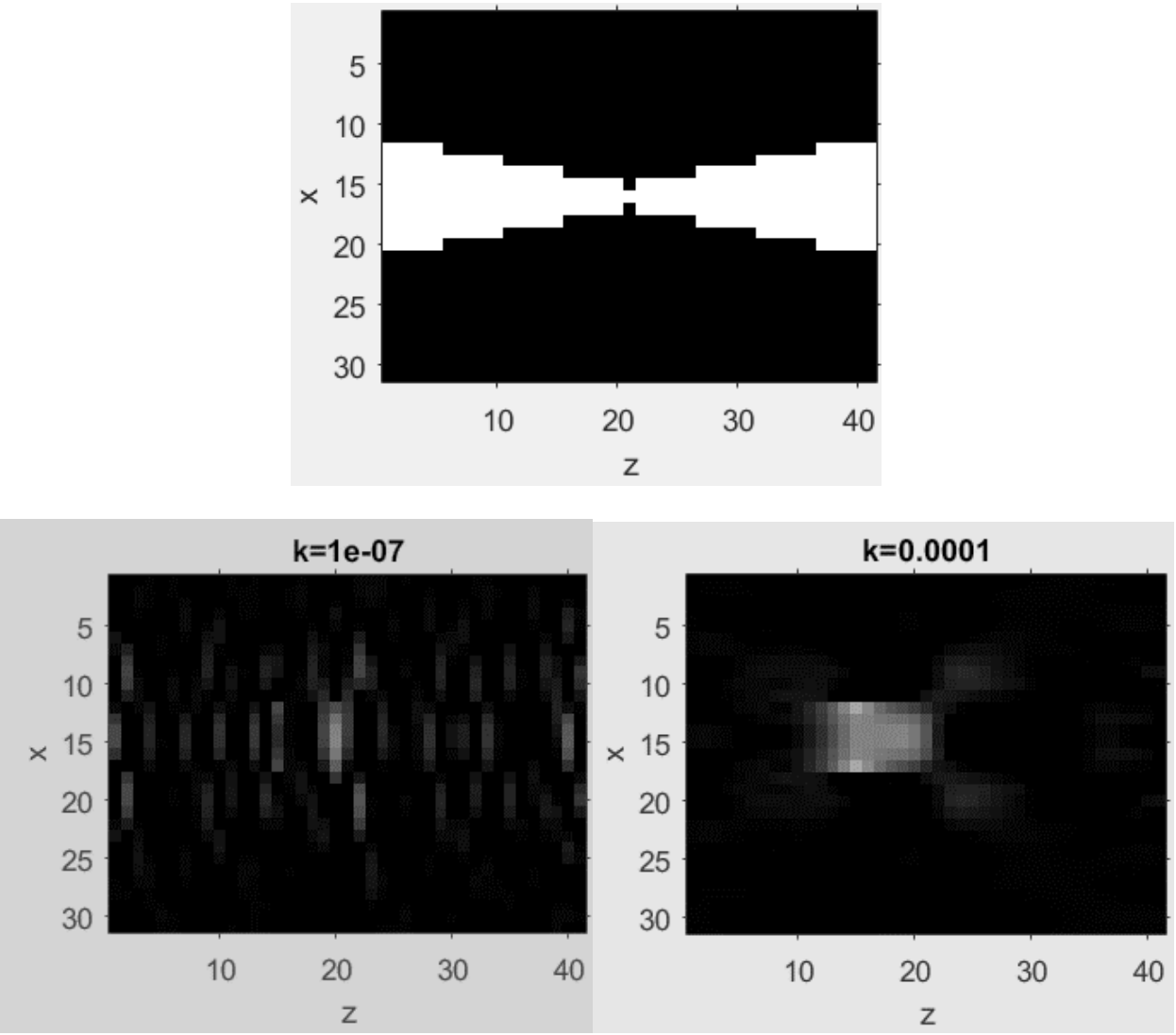

Figure 4.4: General PSF used in the deconvolution (top). Reconstruction with different regularization term value (bottom).

As you can see at $K = 1e - 07$ the reconstruction was completely overshadowed by the artifact result from the mismatch. By increasing the value of K, the algorithm will try to mitigate the artifact by treating the mismatch as some kind of noise, making the reconstruction still retain its supposed location in the focal stack. But the precision is sacrificed as the reconstruction appears very fuzzy.

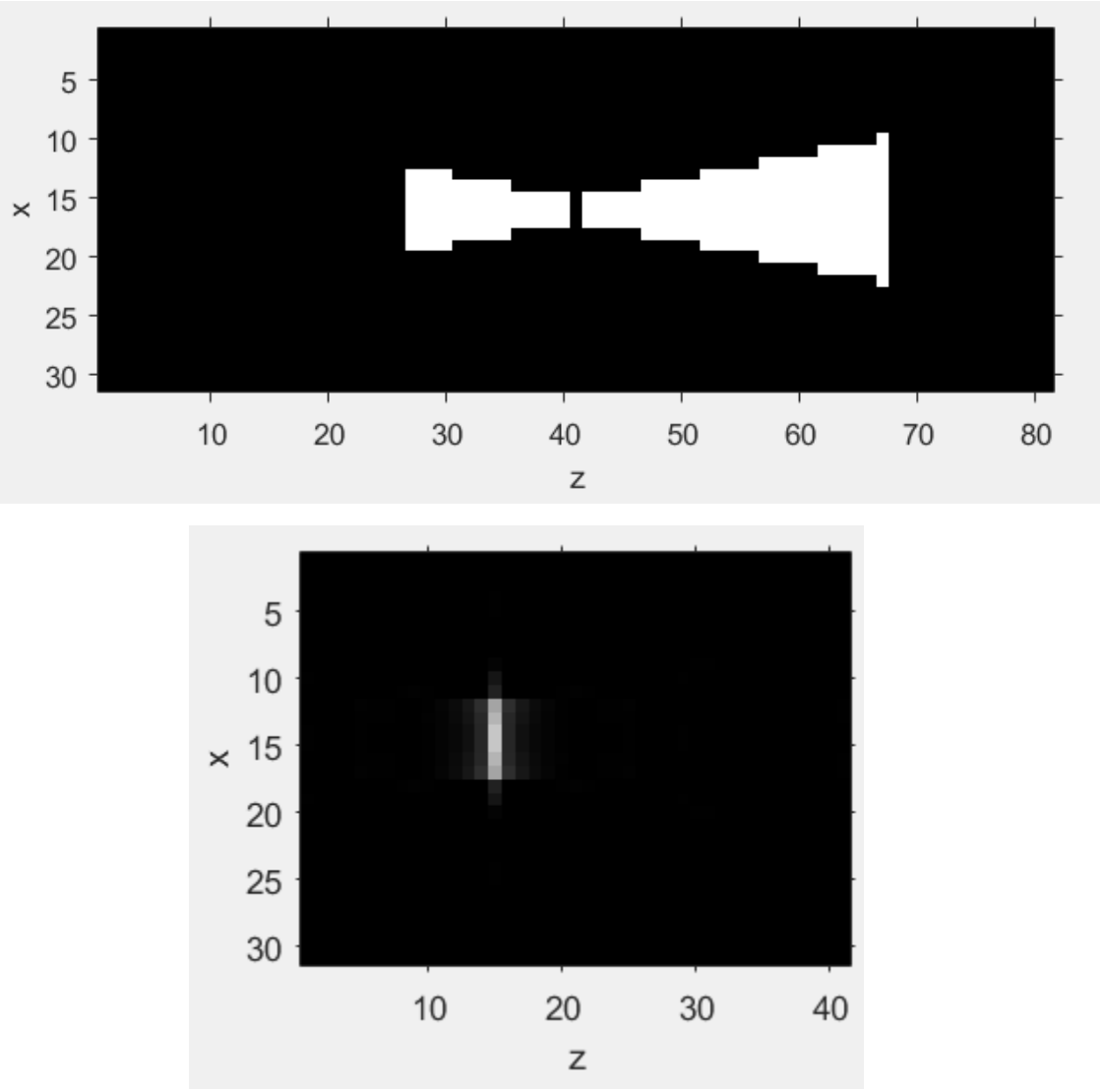

Figure 4.5: Customized PSF used in the deconvolution (top). Reconstruction (bottom)

Figure 4.5 shows the mismatch issue is resolved by using the customized PSF. However, to customize a PSF for an asymmetric focal stack requires knowing the depth of the object beforehand. Luckily we can do that because this is a synthesized focal stack with depth known. However, it would not be the case for any real data. Furthermore, the plenoptic image may contain multiple objects with different depths, which will totally invalidate the idea of customizing the PSF.

4.1.2 Steepest Descent Method

When doing steepest descent one may need to adjust the step size β dynamically. Contrary to instinct, β does not really need to get smaller through time. As the iteration goes on, the line search may end up in an area where the change of gradient is slow, and it may point to the function minimum directly. In that scenario β can be increased considerably to speed up the convergence. In this thesis the method I used is simply trial and error. First, set an initial β; Increase the value of β per 100 iterations also record the last value of the reconstruction before changing β; Compare the value of error term $||y - f(x_k)||$ to its value 100 iterations earlier. If it decreases, keep increasing the value of β, otherwise reset the reconstruction to the last value recorded and decrease the value of β by 10 times as much as the amount it increased per 100 iterations. The reason for doing so is to prevent possible deadlock. Line search may get stuck for a small change of β.

Figure 4.6 shows the reconstruction of the shifted focal stack in Figure 4.3 using steepest descent. As iterations went over 12,000, artifacts and error are still significant, and they are not likely to get resolved in a short time. Figure 4.7 depicts the changing of error over iterations. One can clearly see the tendency of error reduction rate slowing down in later iterations. Since a real focal stack will be much larger, the time for one iteration rises significantly. To reconstruct a surface may easily take days or even weeks. In conclusion, steepest descent is not the ideal solution to the problem.

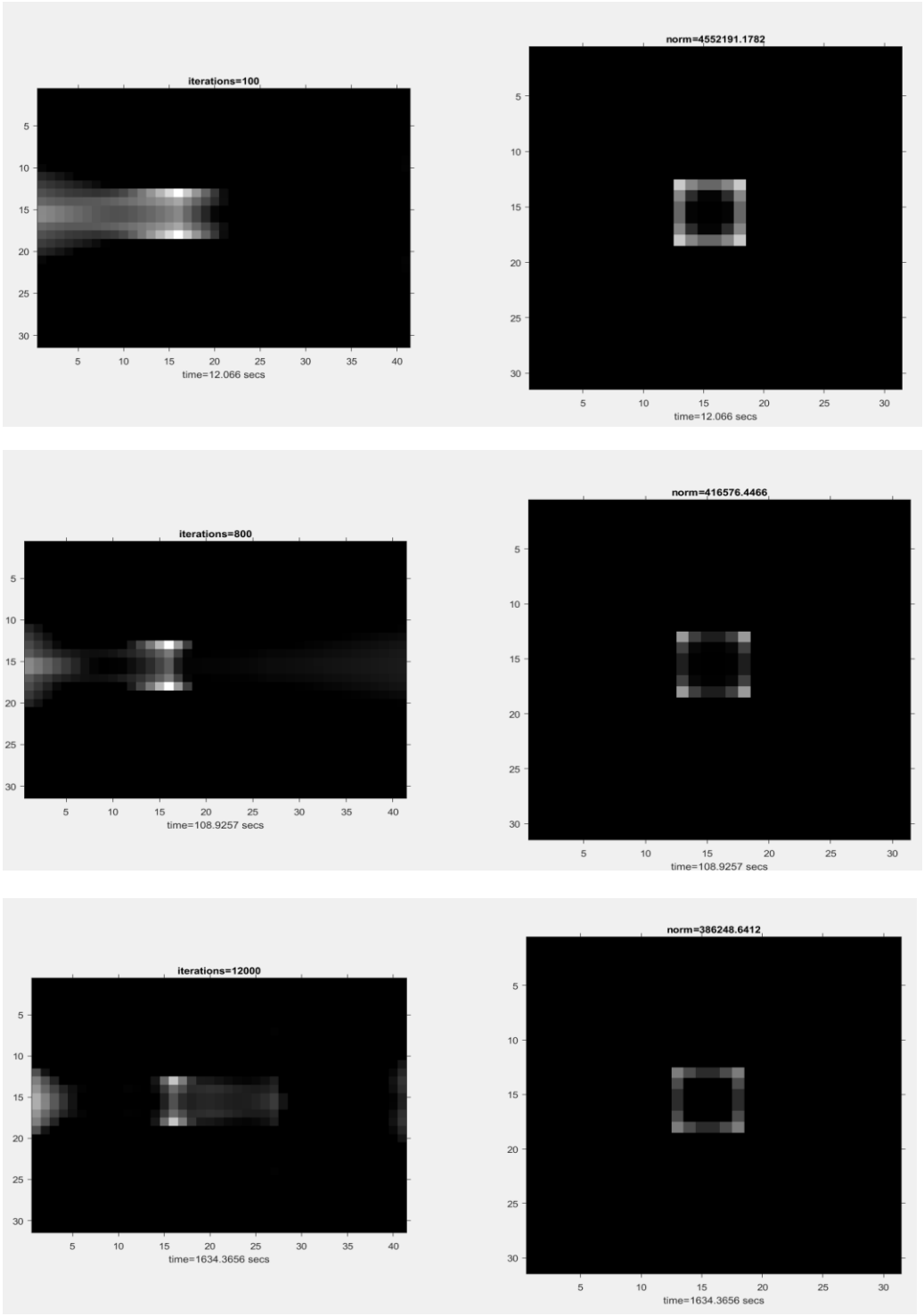

Figure 4.6: Reconstruction from steepest descent as iterations increase (synthesized focal stack, viewed from both (x, z) and (x, y) plane). "norm" is value of error term; "time" is the duration of iterations.

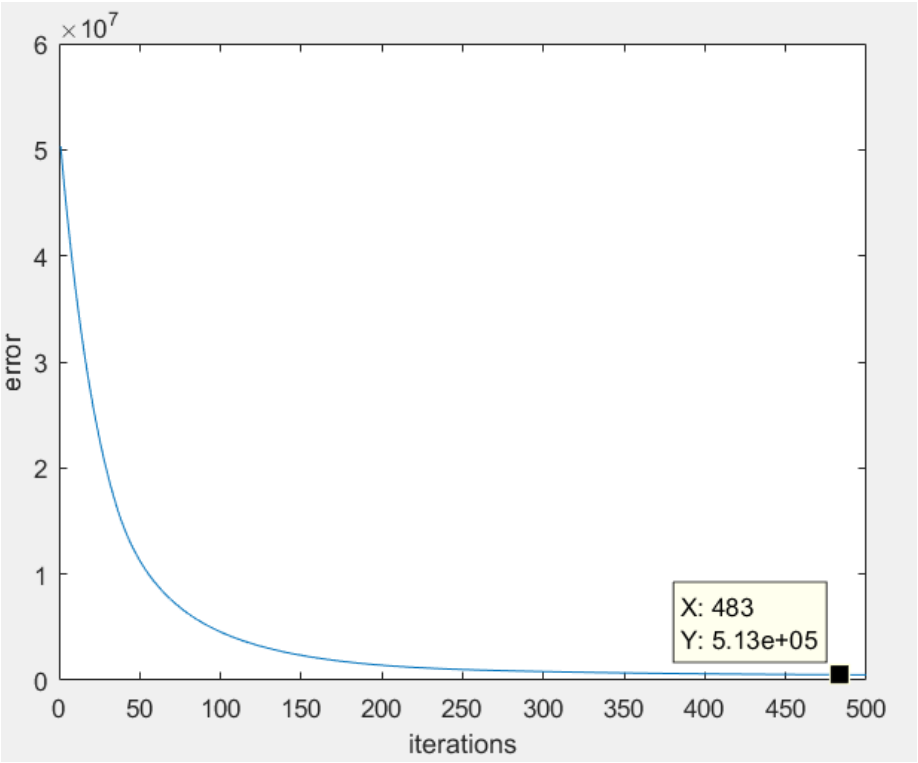

Figure 4.7: Error graph over iterations (steepest descent)

4.1.3 Conjugate Gradient Method

The conjugate gradient method is implemented exactly the same as in Section 3.4.3. First set two initial vectors, one the gradient of the function to minimize and the other in a way that maintains its conjugacy to the first one. Update the value of reconstruction with optimal step size β along the direction of the second vector. It should arrive at the minimum of $||y - f(x_k)||$ in N iterations with N being the number of elements in x_k. So the duration of iteration is directly linked to the size of the plenoptic image. For the case of the synthesized focal stack, the number of iterations should be $31 \times 31 \times 41 = 39401$. Of course it is not required to perform that many iterations. You only need to reduce the blur to a point where you can extract the object, and it is an observation-oriented problem. Future research should come up with an algorithm to decide the optimal iteration number.

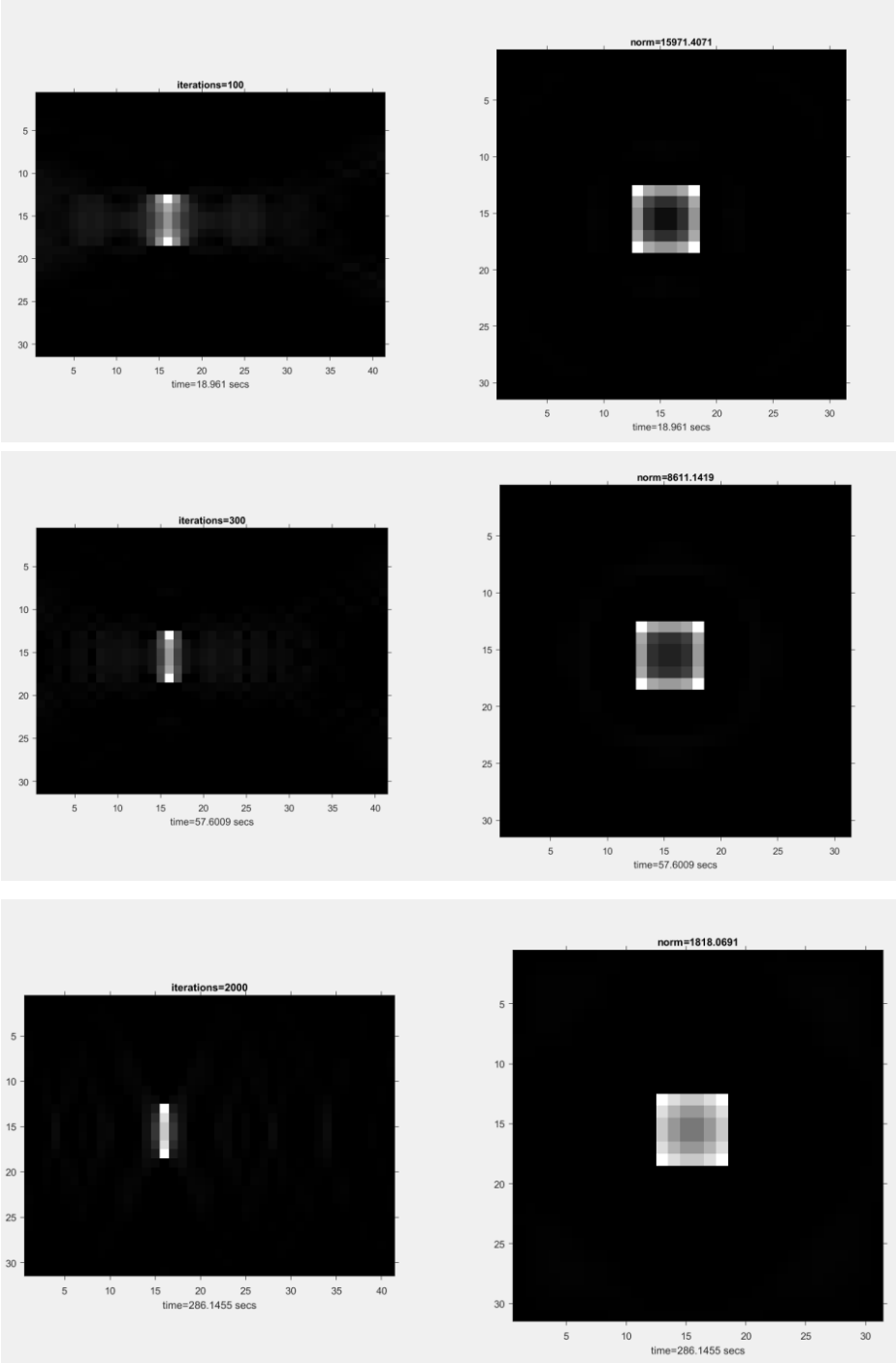

Figure 4.8: Reconstruction from conjugate gradients as iterations increase (synthesized focal stack, viewed from both (x, z) and (x, y) plane). "norm" is the value of error term, "time" is the duration of iterations.

From Figure 4.8 you can see conjugate gradients speeds up the convergence quite significantly. The error was reduced to within 2000 in thousands of iterations (bottom figure in Figure 4.8). It looks almost the same as the result from deconvolution without the mismatch issue.

However, it shares the same problem with steepest descent. First of all, they use the same "route" to locate the function minimum: they all went to the edges first then slowly recovered the information in the middle, and the speed of convergence is reduced exponentially as shown in both Figure 4.7 and 4.9. After both algorithms get close to a minimum, only the edges of the object are defined. The next thousands of iterations will have only a small effect. This poses a huge problem with the real data: simply because every iteration may take tens of seconds, even thousands of iterations will be unacceptable time-wise even though hundreds of thousands may be necessary. This problem will be further discussed in the next section.

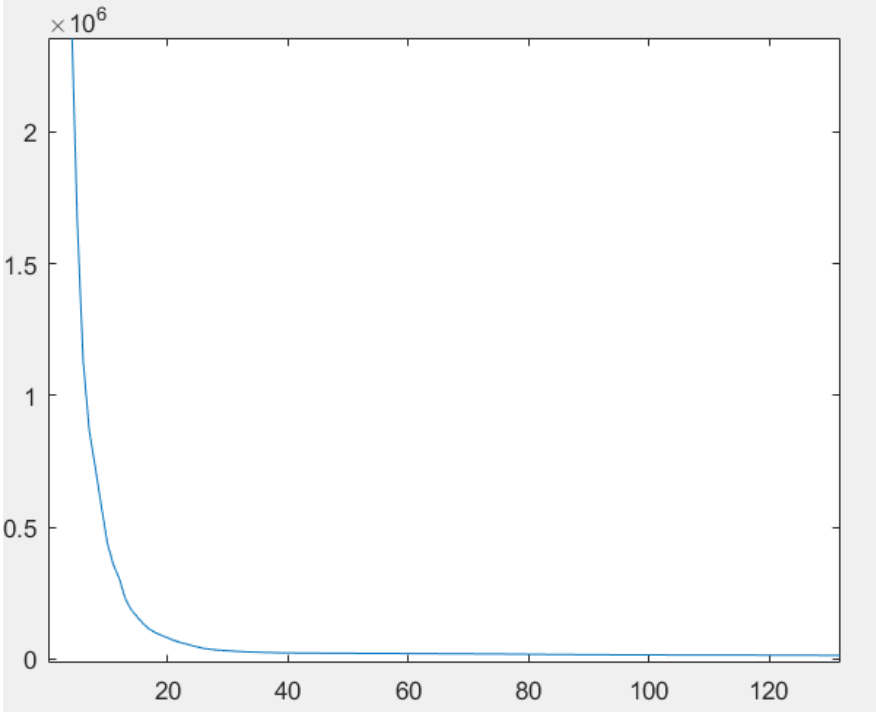

Figure 4.9: Error graph over iterations (conjugate gradient)

4.2 Focal Stack from Real Data

The plenoptic image in this experiment was made by taking pictures of two metal plates at different distances. The one on the right is in focus and the one on the left is further away and out of focus. Their respective distance from the camera is 28 cm and 57.6 cm.

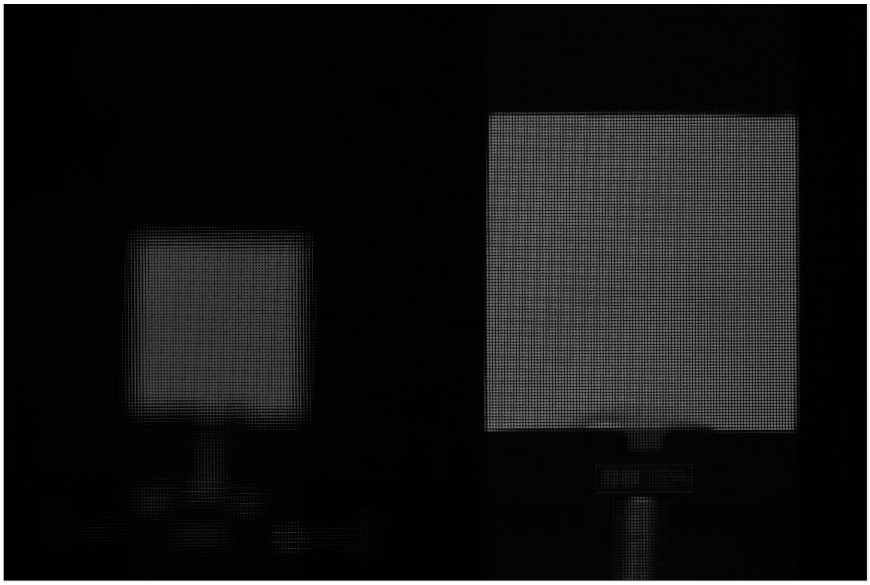

Figure 4.10: Raw plenoptic image used for reconstruction

As it is the convention in the plenoptic toolkit to arrange the image order in the focal stack as their refocused depth decreasing from left to right (with the center being the nominal focal plane). In Figure 4.11 the metal plate at 57.6 cm is on the left at slice 27 while the one at 28 cm is on the center at slice 51.

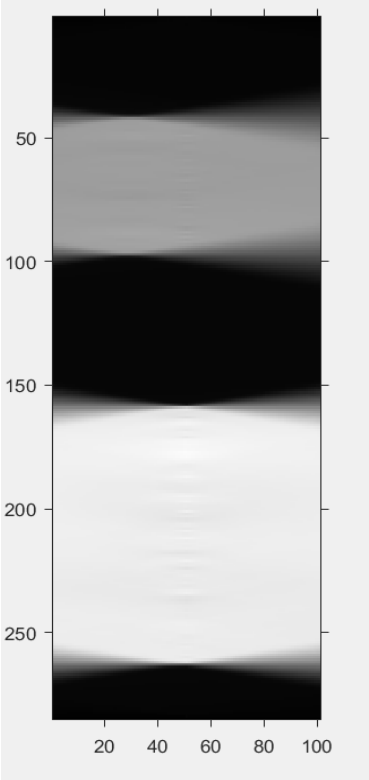

Figure 4.11: Focal stack of the real data

The number of slices in the focal stack was purposely reduced to 100 to save time when doing iterations. More slices mean more precision in object depth but more data to process.

4.2.1 Deconvolution Method

The general unmodified PSF was used for reconstruction. The result is given in Figure 4.12.

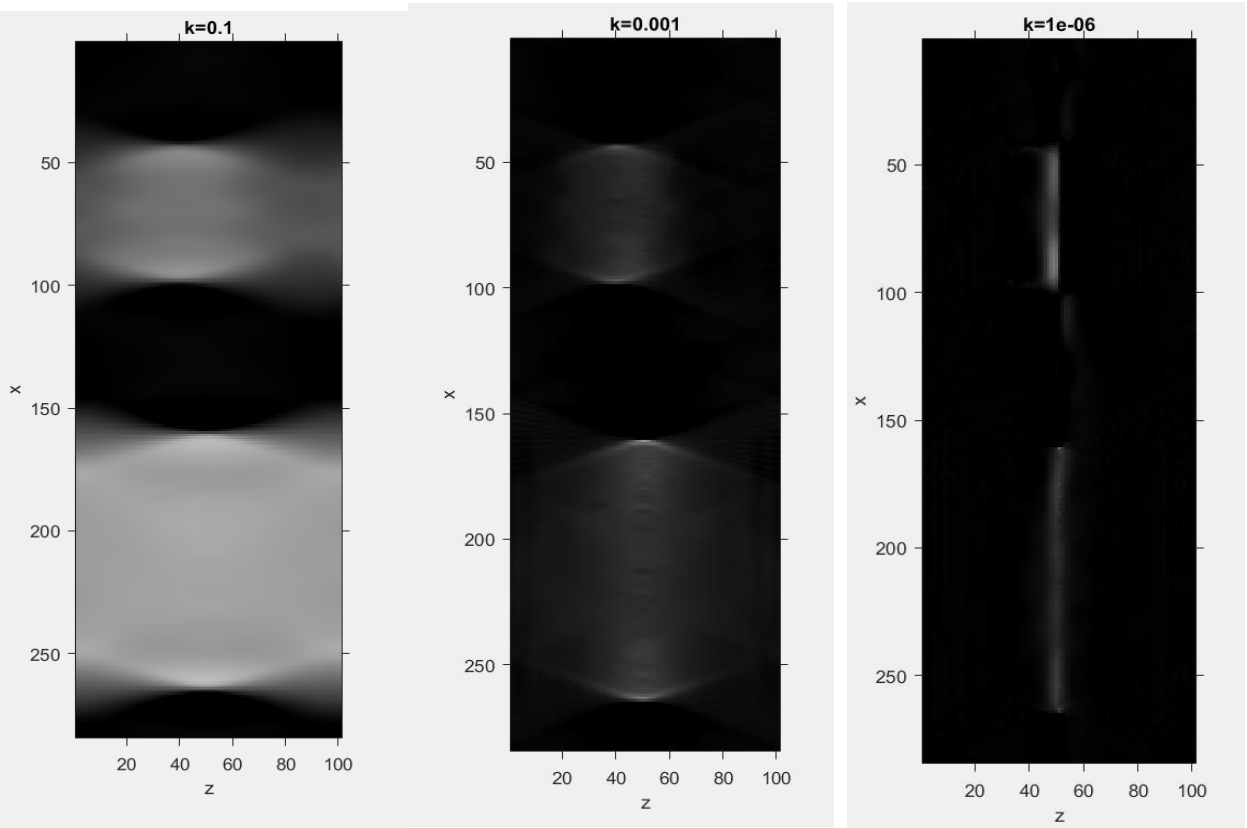

Figure 4.12: Reconstruction of real data with deconvolution method

In Figure 4.12, as the value of the regularization term decreases, the reconstruction becomes sharper. However, the algorithm cannot compensate for the mismatch of the object on the top with a small value of K. As a result the reconstruction starts getting shifted to the center as shown in the last figure in Figure 4.12. The verdict for deconvolution is the method is only useful when the following conditions are met:

1. The object surface is smooth and in the same depth.
2. The depth of the object is known.
3. The object needs to be shifted to the center of the focal stack, which can be accomplished by the plenoptic toolkit. Or alternatively modify the PSF with the depth information to match the focal stack accordingly.

4.2.2 Conjugate Gradient Method

As discussed in Section 4.12, steepest descent has been proven to be incapable of providing an accurate reconstruction within a feasible amount of time. From this point the thesis will only consider conjugate gradients.

Figure 4.13 demonstrates the process of reconstruction by conjugate gradients. The algorithm is able to remove most of the noise around 1000 iterations. However, the same problem persists; the convergence speed slows down considerably after getting near the minimum (where object edges are reconstructed). It may be acceptable for a synthesized stack because it only contains 39401 pixels, but in this case the focal stack generated from the real data contains $198\times284\times100 = 5{,}623{,}200$ pixels. In theory it will take millions of iterations to reach the function minimum while even thousands of iterations already take hours. The idea of recovering the surface from plenoptic image by an iterative method seems at this point unpromising.

However, there is still hope for an iterative algorithm. For example, it may help to find a better starting point other than zero or for the focal stack itself to bypass its current iterative "route". For example, one might use another algorithm to initially construct a crude and fuzzy surface and then rely on the iterative algorithm to do the final polishing. Also the iterative algorithm can be better conditioned by using methods such as total variation.

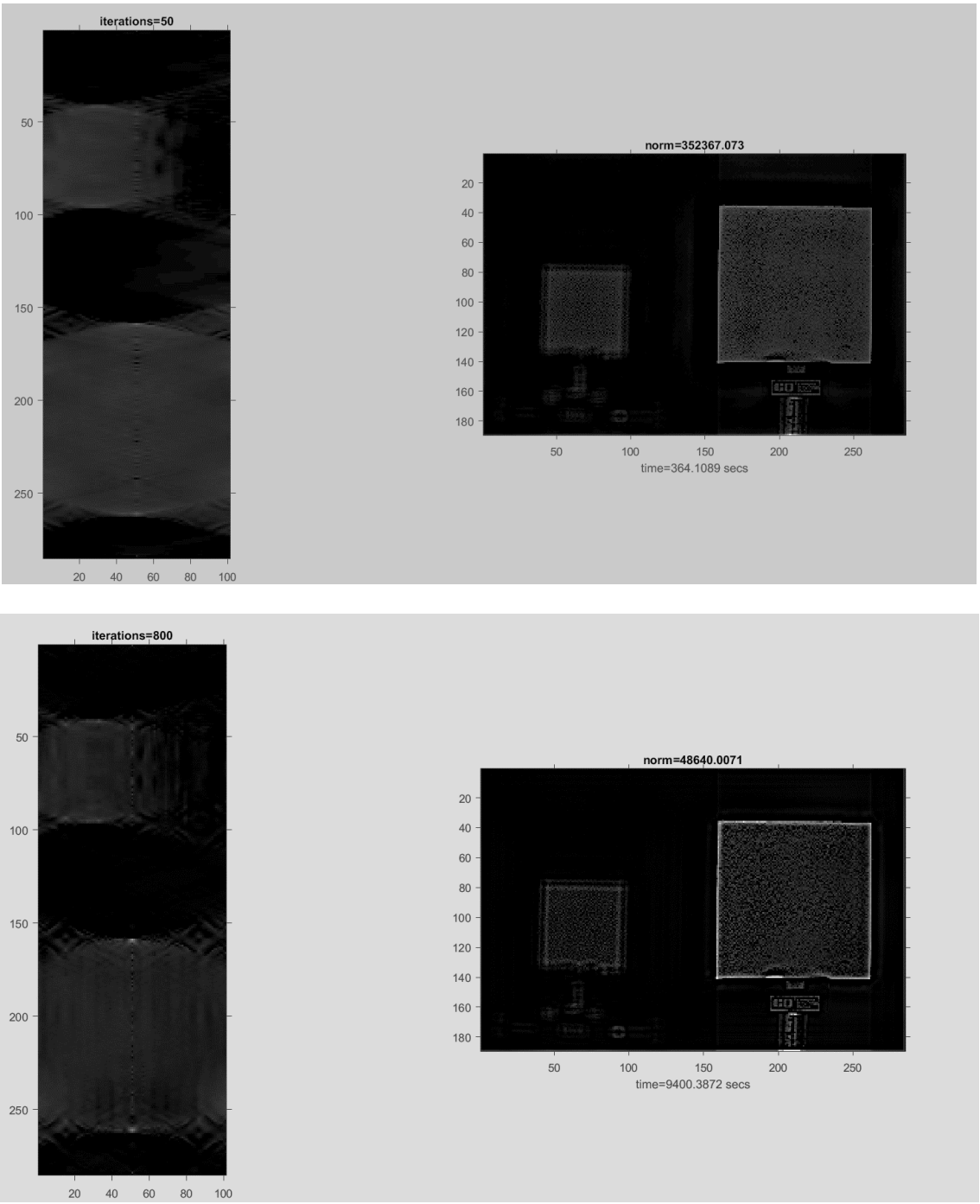

Figure 4.13: Reconstruction from conjugate gradient method as iterations increase (real data, viewed from both (x, z) and (x, y) plane at $\alpha = 1$). "norm" is the value of error term; "time" is the duration of iterations.

4.3 Improved Conjugate Gradients with Total Variation

The total variation method is usually used in iterative algorithms to produce a smoother result. By minimizing the total variation of the data it may speed up the convergence. Take Figure 4.8 as an example. All that conjugate gradients does is locate the edges of the object, then slowly recover the surface information by converging towards the center. It is expected during the process that conjugate gradients is producing a gradient difference around the edges inwards. If we could condition the algorithm to go a direction that minimizes total variation with the influence of the original line minimization, we may be able to bring up the value in the lower half of the gradient while minimizing total variation and essentially speeding up the convergence.

Figure 4.14: Surface reconstruction from conjugate gradient with insufficient iterations (left). Theoretically how total variation would modify the result with around the same amount of iterations(right).

Recall that the error function we want to minimize in the iterative algorithm is given by $||y - f(x_k)||$. To implement total variation, we modify the function by adding another error term to it: $\phi(x_k) = \left||y - f(x_k)|\right| + \alpha||D(x_k)||$. $||D(x_k)||$ is calculated as:

$$\sum_{m,n,p} \{|x(m,n,p) - x(m-1,n,p)| + |x(m,n,p) - x(m,n-1,p)| + |x(m,n,p) - x(m,n,p-1)|\}$$

4.3.1 Result from Total Variation

In this experiment the same focal stack from Section 4.1 was used, by adjusting the scale factor α accordingly (in this case $\alpha = 3200000$). As shown in Figure 4.15, the iteration required to reconstruct a surface could be considerably reduced:

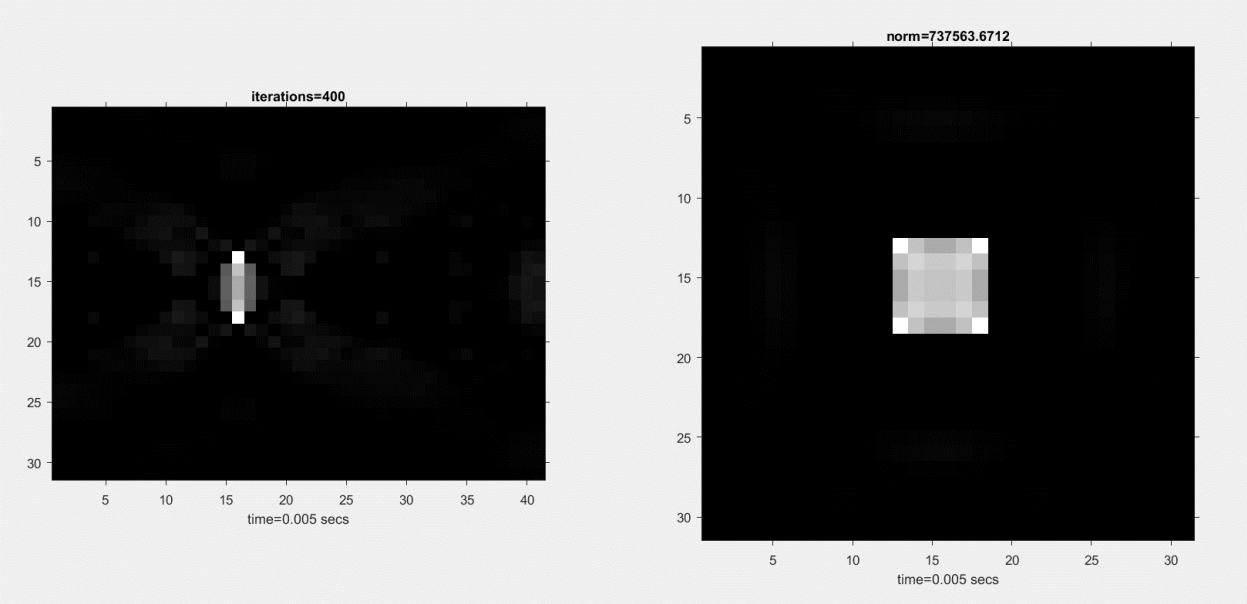

Figure 4.15: Reconstruction from total variation with synthesized focal stack

Compared to Figure 4.8, total variation closed in much faster at 400 iterations, while conjugate gradients alone would take about 2000. But this is still not satisfactory. We need a way to exponentially reduce the iterative duration. Even if we manage to reduce it to one quarter of its original duration, that is still unacceptably long for real data.

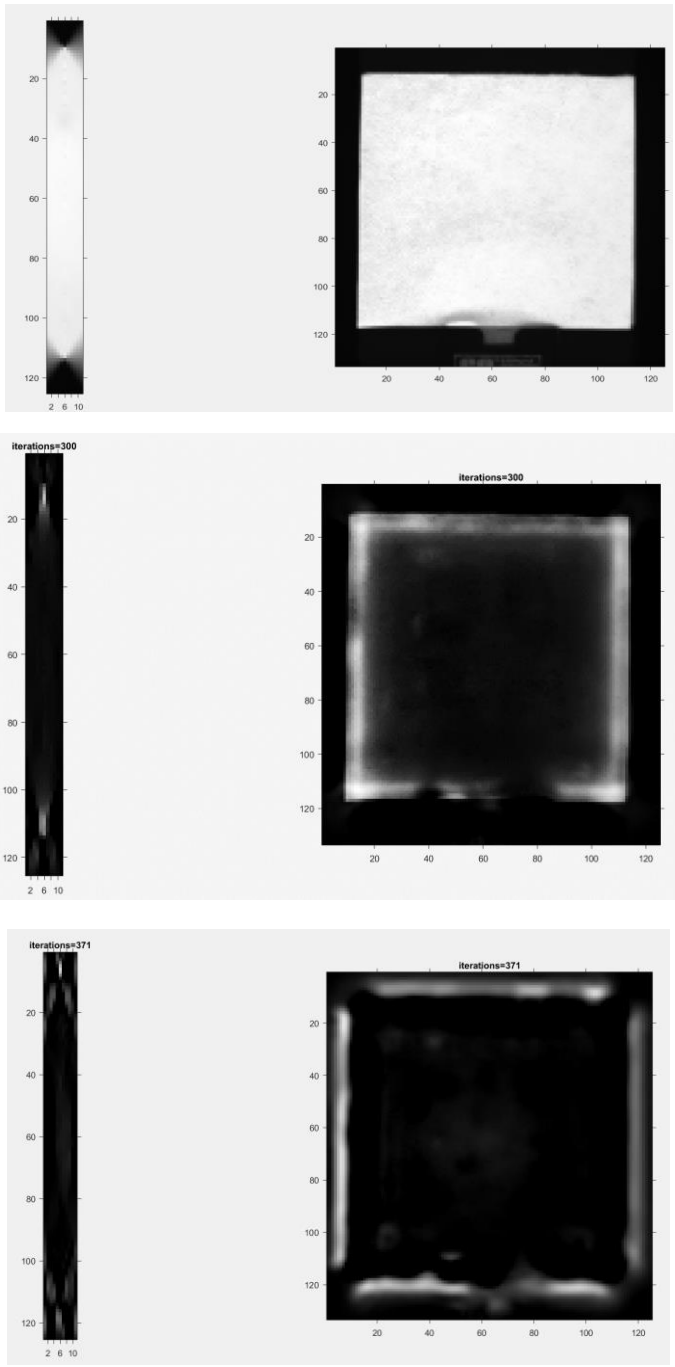

Figure 4.16: Reconstruction from total variation with real data

Original focal stack (top). Reconstruction at 300 iterations (middle). Reconstruction passes 300 iterations (bottom).

As shown in Figure 4.16, total variation does speed up the convergence to some degree. However, as total variation proceeds inwards much faster than the original line search, the latter can no longer catch up with total variation to continuously generate more gradient difference. As a result, the convergence speed slows down again after a certain amount of information was filled in. It became worse after the line search was totally overwhelmed by total variation in the late stage of iterations. The reconstruction starts going outside the edges of the object because line search has such a little influence over the data.

4.4 Unstable Modified Conjugate Gradient

A few more modified conjugate gradient algorithms like negative penalty and preconditioning were considered. Unfortunately, those algorithms hardly provide any improvement. Thus, another modification was tested which showed some promise. However, it is unstable.

Recall that in conjugate gradients vector r was calculated as the negative of the function gradient: $\vec{r}_{k+1} = \vec{r}_k - \lambda_k H^T H \vec{p}_k$. We investigated adding a term $\vec{p}_k{}^3$ to the end of the equation, making it: $\vec{r}_{k+1} = \vec{r}_k - \lambda_k H^T H \vec{p}_k + \alpha \vec{p}_k{}^3$. The iterations required to recover the surface could be greatly reduced with a proper scale factor α.

As shown in Figure 4.17, in just 1400 iterations, we have managed to recover the surface information. The reason behind it we theorized is because when $\vec{p}_k{}^3$ is added to $\vec{r}_{k+1}$ it brings $\vec{r}_{k+1}$ closer to $\vec{p}_k$. When you calculate $\vec{p}_{k+1}$ by $\vec{p}_{k+1} = \vec{r}_{k+1} + \gamma_k \vec{p}_k$.you are making $\vec{p}_{k+1}$ partially conjugate and partially orthogonal to the previous $\vec{p}$. That way you are going non-overlapping directions every time and that is why it converged much faster.

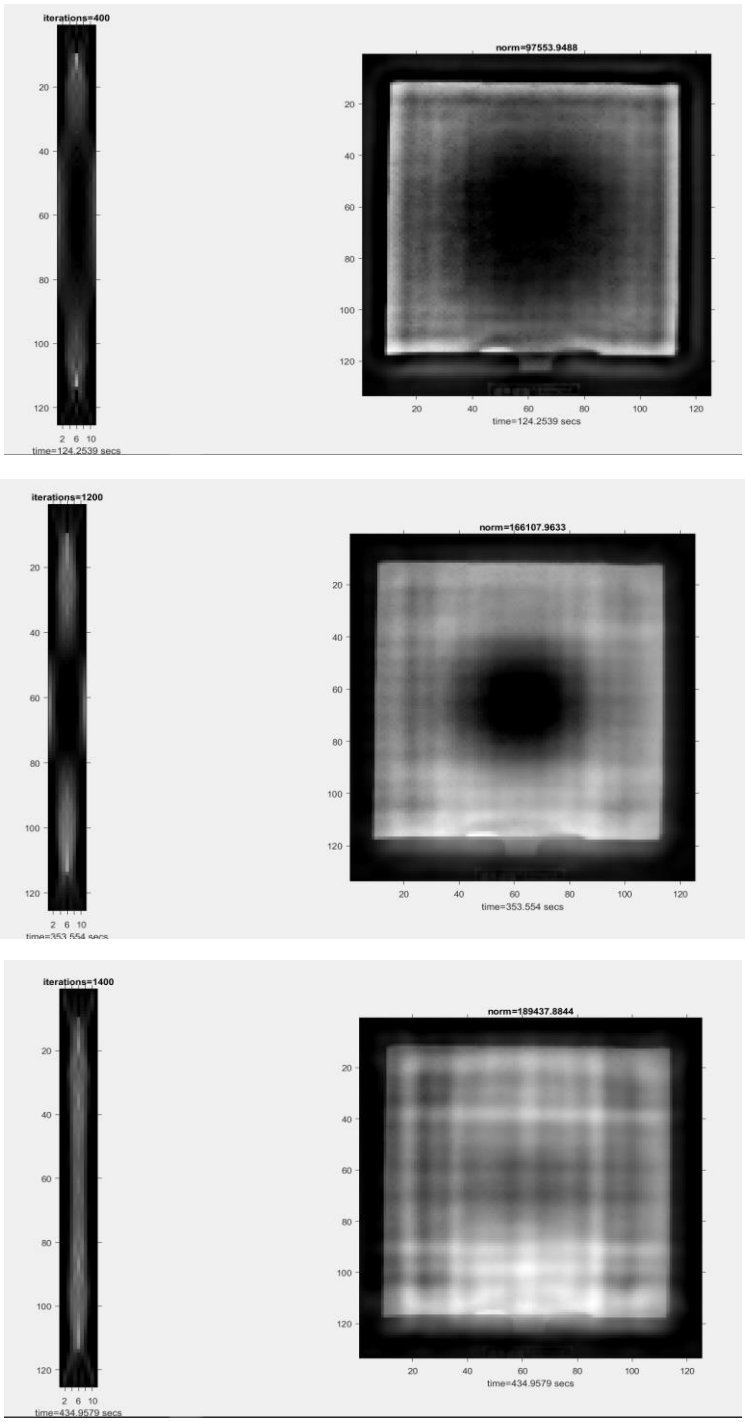

Figure 4.17: Reconstruction at 400 iterations (top). Reconstruction at 1200 iterations (middle). Reconstruction at 1400 iterations (bottom).

However, the algorithm does have one flaw. It could be very unstable depending on the scale

factor. As the size of the data gets bigger it is harder to choose a proper α. If α is too small, the modification would not be able to make any difference. If α is too big $\vec{p}$ tends to blow up to infinity at some point. Because you are constantly adding one $\vec{p}_k{}^3$ per iteration, it is likely with a big α the two vectors would go out of bounds before the reconstruction can be finished.

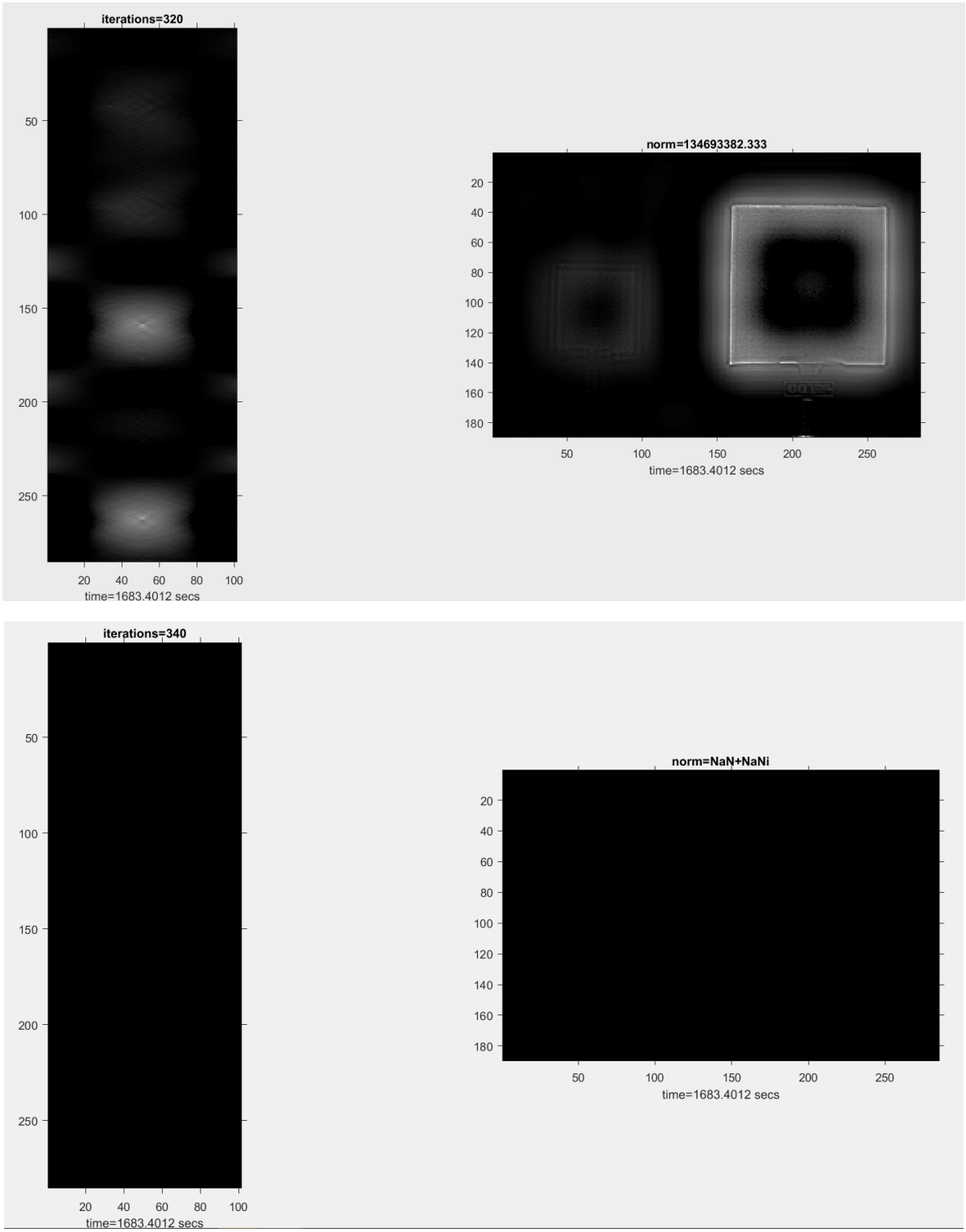

Figure 4.18: Reconstruction blows up before it can be finished

4.5 Surface Reconstruction Using a Direct Line Search Approach

The failure of the iterative method is due to the ill-conditioned line minimization. The directions the algorithm can go to reduce the error term are simply too diverse especially in a 3D space, which is the reason we chose to condition those iterative algorithms with total variation, negative penalty, etc. However, they all failed to address one of the most important issues, the opacity of the object in the slice-wise direction. The object being opaque means there is only one slice of surface in the reconstruction per (x, y) unit. The intention to constrain the reconstruction within only one slice per (x, y) unit leads to a very simple line search approach, yet it actually yields some conclusive results worth presenting.

The algorithm starts by assuming the opacity of the object by creating only one small surface in the focal stack space. Then it proceeds to "slide" this surface along the z axis while generating the corresponding focal stack using convolution with a PSF. When this surface is matched with the actual object surface, it should yield the smallest error by subtracting the full object focal stack from the surface focal stack. Therefore, we refer to this method as Stack Matching. We select a different (x, y) location to create another small surface and repeat the above processes until the entire plenoptic image is searched.

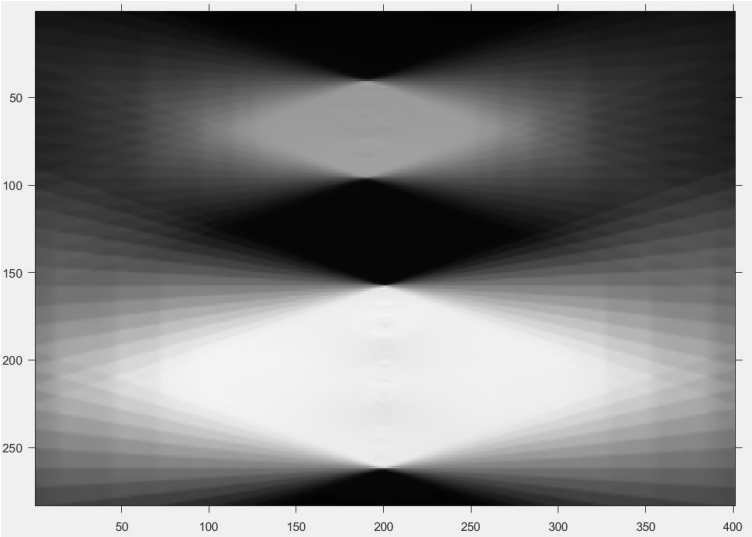

Figure 4.19: A focal stack covers a long range of relative focal length α

As Figure 4.19 suggests, the focal stack used in this method needs to cover enough range of α so that the influence of superimposed image blur can be reduced at both ends to make stack matching possible.

By making the stack-matching surface a 9 by 9 square, I was able to get a crude estimation of the surface location as shown in the figure below:

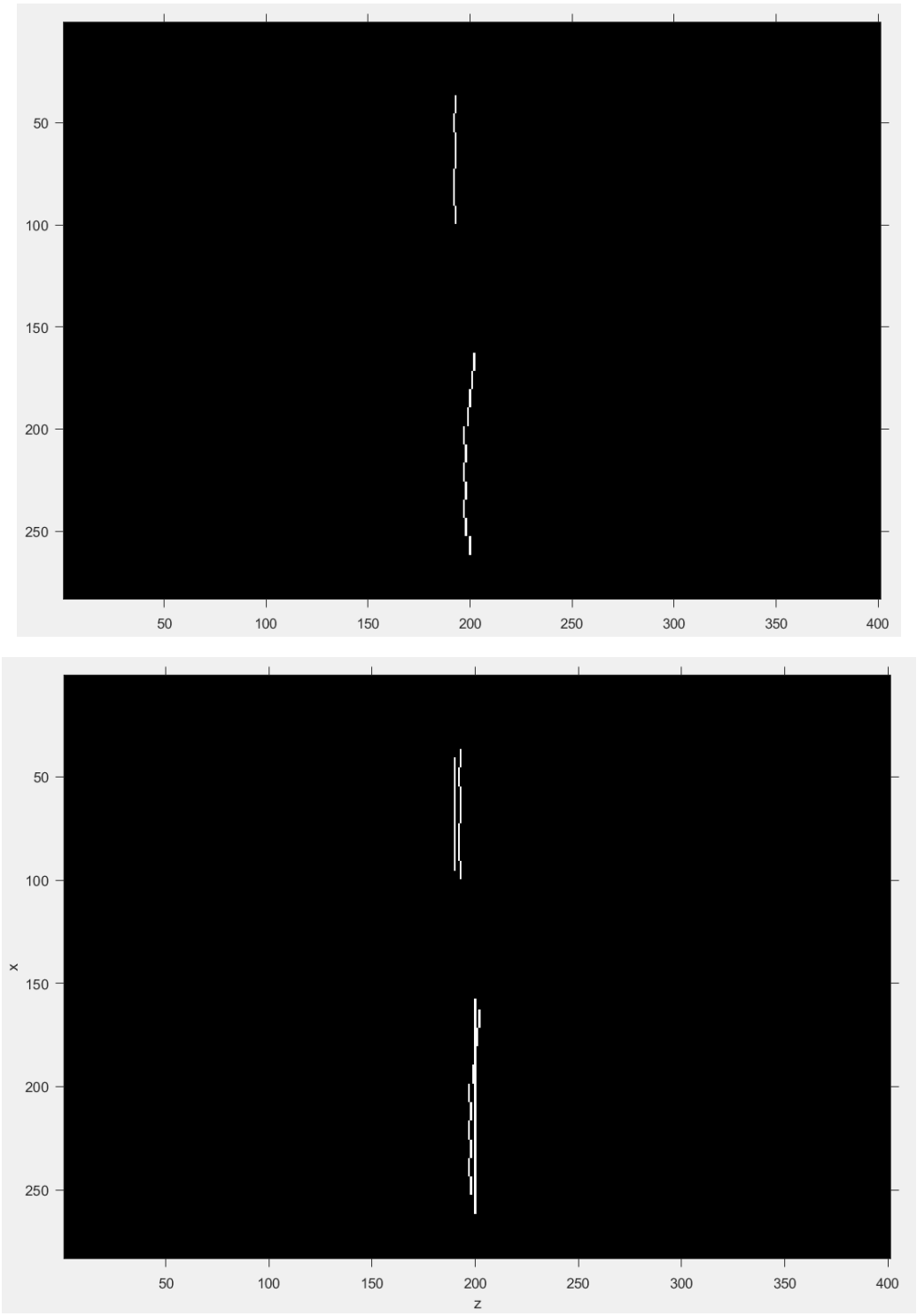

Figure 4.20: Reconstruction by Stack Matching method (top) with comparison to the actual object surface location (characterized by the two straight lines)

The dilemma in this method is that the size of the surface used in the Stack Matching cannot be so small that it makes too little difference in the error calculation due to the superposition in the focal stack. But a big surface does not have a very good precision for matching small surface details.

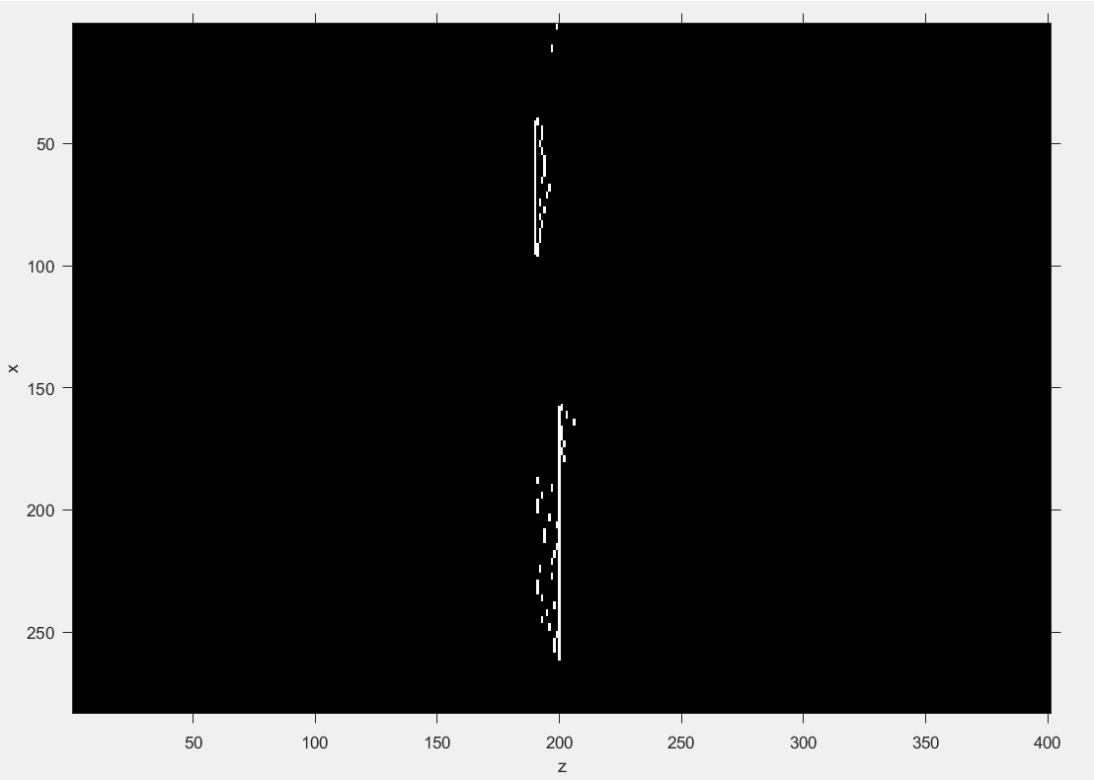

Figure 4.21: Reconstruction by Stack Matching method by 3 by 3 surfaces

Chapter 5

Conclusion and Future Work

There are various methods of estimating depth from plenoptic image. However, their practicality is very limited as they are unable to detect smooth objects, making most surface reconstruction unviable. Thus, a new method was presented called the deconvolution method. By modeling the process of image blur in a focal stack as the result of convolution with a PSF, in theory one is able to retrieve the surface information by reversing the convolution—thus deconvolution.

Again, however, deconvolution has been shown to be impractical in this thesis. Because of the shift-variant nature of the focal stack, objects not in the center of the focal stack will be truncated asymmetrically thus creating mismatch between the stack and the PSF. If the depth of the object is unknown, the convolution is irreversible by a direct form. The thesis presents a new way of reconstructing a surface by iterative algorithms. Steepest decent and conjugate gradient methods were implemented and tested. The conjugate gradient method is able to reconstruct a synthesized surface. However, it fails when trying to reconstruct surface from real data due to the sheer amount of time the iterations requires. A few modified conjugate gradient techniques were implemented. Total variation increases the convergence speed to a certain degree but not fast enough; modifying the conjugate vector accelerates convergence greatly but remains an unstable and ill-theorized algorithm. Future testing is still needed.

Future research should focus on finding a better conditioned iterative algorithm to shorten the computation time. Even better would be finding a non-iterative algorithm to reconstruct a surface at a much faster speed. Neural networks may be a good candidate to consider.

Bibliography

[1] T. G. Georgiev and A. Lumsdaine, "Resolution in Plenoptic Cameras," in Frontiers in Optics 2009/Laser Science XXV/Fall 2009 OSA Optics & Photonics Technical Digest. Optical Society of America, 2009, p. CTuB3. [Online]. Available: http://www.opticsinfobase.org/abstract.cfm?URI=COSI-2009-CTuB3

[2]——, "Digital Light Field Photography," Ph.D. dissertation, Stanford University, July2006.

[3] S. Yang N. Cohen A. Andalman K. Deisseroth M. Broxton, L. Grosenick and M Levoy. Wave optics theory and 3-d deconvolution for the light field microscope. *Opt. Express*, 21.

[4] R. Bracewell, "Strip Integration in Radio Astronomy," *Aust. J. Phys*., vol. 9, no. 2, pp. 198–217, Jan. 1956. [Online]. Available: http://www.publish.csiro.au/paper/PH560198

[5] E.H. Adelson and J.Y.A. Wang. Single lens stereo with a plenoptic camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):99106.

[6] Frdo Durand William T. Freema Anat Levin, Rob Fergus. Deconvolution using natural image priors. *Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory*, 2009.

[7] P. Favaro and S. Soatto. Learning shape from defocus. *Computer Vision ECCV Lecture Notes in Computer Science*, pages 823–8241

[8] M. Schlosser J. Jachalsky and D. Gandolph. Confidence evaluation for robust, fastconverging disparity map refinement. *IEEE International Conference on Multimedia and Expo (ICME),* page 13991404.

[9] M. Levoy and P. Hanrahan. Light field rendering. *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH 96*, page 3142, 1996.

[10] Zhang Z. Levoy, M. and I McDowell. Recording and controlling the 4d light field in a microscope using microlens arrays. *Journal of Microscopy,* 235(2):144,162.

[11] H.-Y. Lin and C.-H. Chang. Depth recovery from motion and defocus blur. *Image Analysis and Recognition,* 4142.

[12] Stefano Mattoccia. Stereo vision: Algorithms and applications. *University of Bologna*, 2013.

[13] M.levoy. Light fields and computational imaging. *Computer (Long. Beach. Calif)*, 39:44–55,

2006.1.

[14] S. J. Reeves P. Anglin and B. S. Thurow. Direct fft-based volumetric reconstruction from plenoptic data. *IEEE Transactions on Image Processing*, 2015.

[15] G. Duval M. Horowitz R. Ng, M. Levoy and P. Hanrahan. Light field photography with a hand-held plenoptic camera. *Informational***,** page 111, 2005.

[16] Ronneberger O. Nitschke R. Driever W. Temerinac-Ott, M. and H. Burkhardt. Spatially-variant lucy-richardson deconvolution for multiview fusion of microscopical 3d images. Biomedical Imaging: *From Nano to Macro, 2011 IEEE International Symposium on*, page 899 904.

[17] Filip Sroubek Xiang Zhu1 and Peyman Milanfa. Deconvolving psfs for a better motion deblurring using multiple images. ECCV, 2012.

[18] Adelson, E. H.; Wang, J. Y. A. (1992). "Single Lens Stereo with Plenoptic Camera". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (2): 99–106.

[19] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery. Numerical Recipe in C, The Art of Scientific Computing, Second Edition. Page 414. 1992.

[20] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery. Numerical Recipe in C, The Art of Scientific Computing, Second Edition. Page 421. 1992.

[21] Jan A. Snyman (2005). Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. Springer Publishing. ISBN 0-387-24348-8

[22] Runar H. Refsnaes (2009). A Brief Introduction to The Conjugate Gradient Method. Available: http://www.idi.ntnu.no/~elster/tdt24/tdt24-f09/cg.pdf

[23] Haiqiao Zhang (2015) 3D Surface Reconstruction Based On Plenoptic Image