“Project Hummingbird”

Plenoptic Imaging for Industrial Inspection Design Description Document

Wen Zhou (Project Coordinator)
Stephen Watson (Customer Relationship)
Xiaojing Huang (Document)
Weichen Yao (Scribe)

Faculty Advisor: Dr. Scott Carney

Customer: Navitar / Ian Wallhead / Russ Hudyma

Document Number 00006

Revisions Level	Date
F	05-04-2018

This is a computer-generated document. The electronic master is the official revision. Paper copies are for reference only. Paper copies may be authenticated for specifically stated purposes in the authentication block.

Authentication Block

Revision History

Rev	Description	Date	Authorization
A	Initial DDD Released	01-21-2018	All
B	Secondary Review	02-05-2018	All
C	Third Review	02-19-2018	All
D	Midterm DDD	02-26-2018	All
E	Fifth Review	04-01-2018	All
F	Final DDD	05-04-2018	All

Table of Contents

1. Introduction

There is a market for optics to be used in assembly lines and other inspection systems for quality assurance that can adapt to defocus. Navitar is designing such a system that makes use of plenoptic imaging. With an optical system design done by a previous senior team, our team is looking to characterize the limits of this system, and the methodology of image reconstruction.

Statement:

The Plenoptic Imaging for Industrial Application is a senior design driven product. As such its design inputs are derived from our interactions with our project customers, Ian Wallhead and Russ Hudyma for Navitar, and our faculty advisor, Dr. Scott Carney.

Vision:

To present a trade study of a plenoptic system with regard to microlens dimensions, sensor type, and image space f/#, and to provide the image reconstruction tools to implement Navitar's plenoptic lens system.

2. System Overview

2.1 System Block Diagram

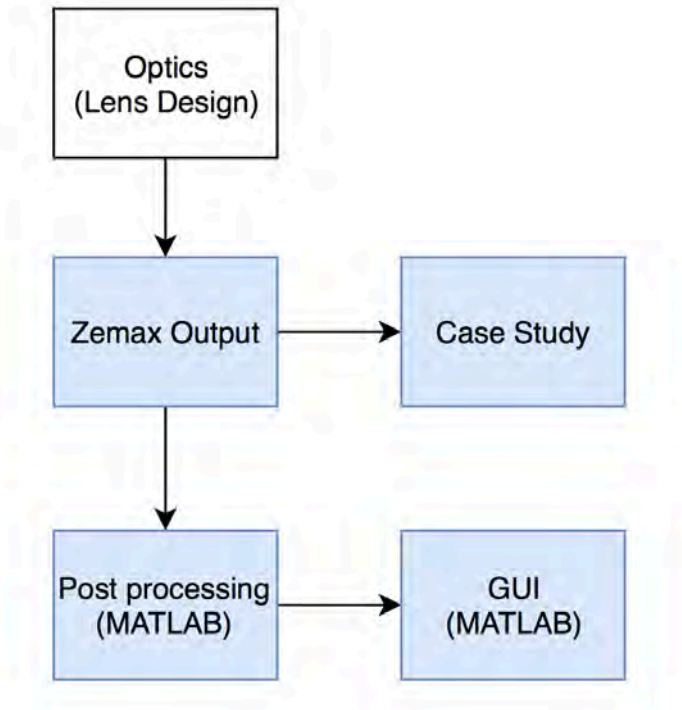


Figure 1: *System Block Diagram. Shaded Blue Boxes are our responsibilities.*

2.2 Specifications

The post-processing tools	
Programming Language	MATLAB (version later than R2016a)
Input	Pixel data file (.txt) exported from ZEMAX
Output	Reconstructed images (.jpg)

Table 2: *Post-processing Tools.*

2.3 Post Processing Algorithm Workflow

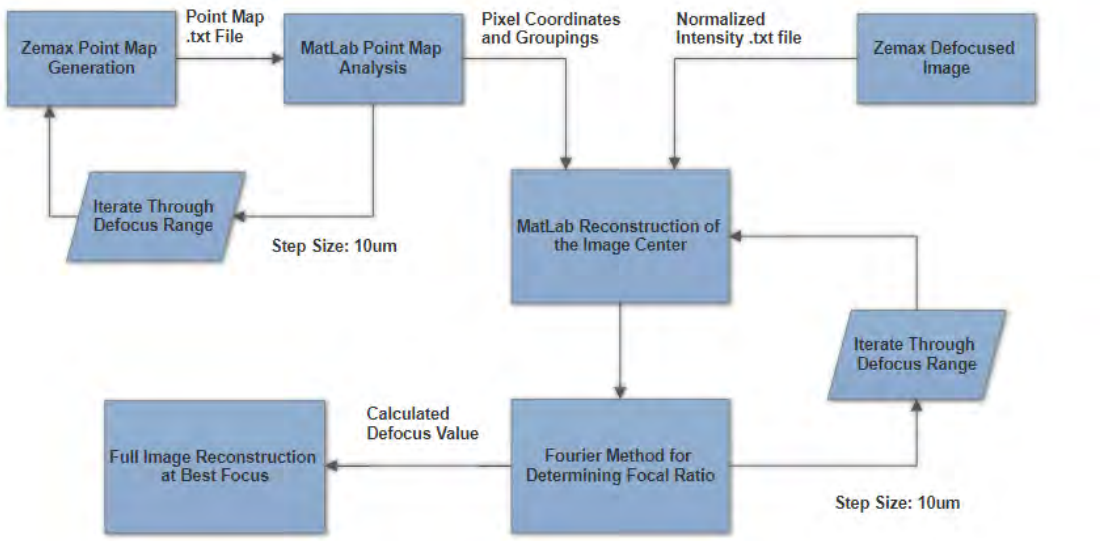


Figure 2: *The workflow for the algorithm to take an image from ZEMAX to Reconstruct it and determine the amount of defocus. The ZEMAX Point Map Generation and MATLAB Point Map Analysis only need to be done after a change to the ZEMAX system design, not for each image.*

3. Project Scope

3.1 We Are Responsible for:

- Generating un-reconstructed image formed by the plenoptic lens using ZEMAX simulation
- Studying the un-reconstructed image at different out of focus distances
- Reconstructing the image using MATLAB within an acceptable timeframe
- Developing a GUI for future use
- Developing a method to determine the best focus position

3.2 We Are Not Responsible for:

- Applying the research to alter the design of the system

4. Design Options

As the main task is to characterize the system, the main design portion of our task is the design of the reconstruction algorithm. The difficulty in the reconstruction process is determining the amount of defocus in the system so that an iterative process can accurately reconstruct the image. To determine the defocus several methods could be used, possibly a combination of tests could be used.

One possible method is a calibration test, as this could be used for an assembly line in which an in-focus picture of the object is provided, then a comparison could be used to determine the amount of defocus. According to our customer, a calibration test would be needed, but it needs to be a fallback solution to the standard due to implementation issues.

Another method is that if a Z-axis sample translation stage is built into the product, a small calibration spot could be included to compare the focus of the known distance height to the object height to determine the defocus using epipolar imaging. Our customer believes that this would be useful especially for objects with height, as epipolar images are mainly used to create a 3D map and could be used in conjunction with the third method.

The last method is to determine the amount of defocus in terms of microlens arrays. This could be done by iteratively reconstructing the image guessing at the amount of defocus, and then determining which picture is the best. This is the method pursued in our reconstruction algorithm. If other methods are included in the final design they will be chosen based on the limitations of this method, and on the design changes implemented by our customer based on our case study.

5. Design Process

5.1 Basic Idea

A plenoptic imaging set up relies on having a set of microlenses just in front of the sensor. Each set of pixels behind the microlenses makes up a microlens array. The use of this microlens array is that it provides additional information about this system, namely a directionality component. The cost of this information is a loss in resolution as the algorithm that reconstructs the image must averages out a microlens array to be one pixel. Despite the loss in resolution, having the directionality component of the light allows you to reconstruct the image to different defocus depths. Our algorithm, detailed below, was based on reconstruction algorithms from our references and altered based on our research using the current system setup. The following chapter notes our research and how we altered our design based on the research

5.2 MATLAB Reconstruction of the Center

As the reconstruction portion of the algorithm is both the most difficult bit, and the least familiar to our team, the majority of the first semester was spent researching, and analyzing potential methods of reconstruction. The below sheet gives the baseline for our code. It was designed working under the assumption that the distance between each pixel that is reconstructed back to the center pixel is an integer number of microlenses away.

5.3 Non-Integer Microlens Array Defocus Positions

At integer amounts of defocus, it is known that the distance between each pixel to be reconstructed together is an integer number of pixels and that all of the intensity at a pixel refers to one point on the image plane. We saw this behavior at 150 μm and 250 μm. We took an in-depth look at the effects of a single point being imaged at defocus locations

from 0 to 150 microns. The pictures displayed below at 10 microns defocus differences between each image.

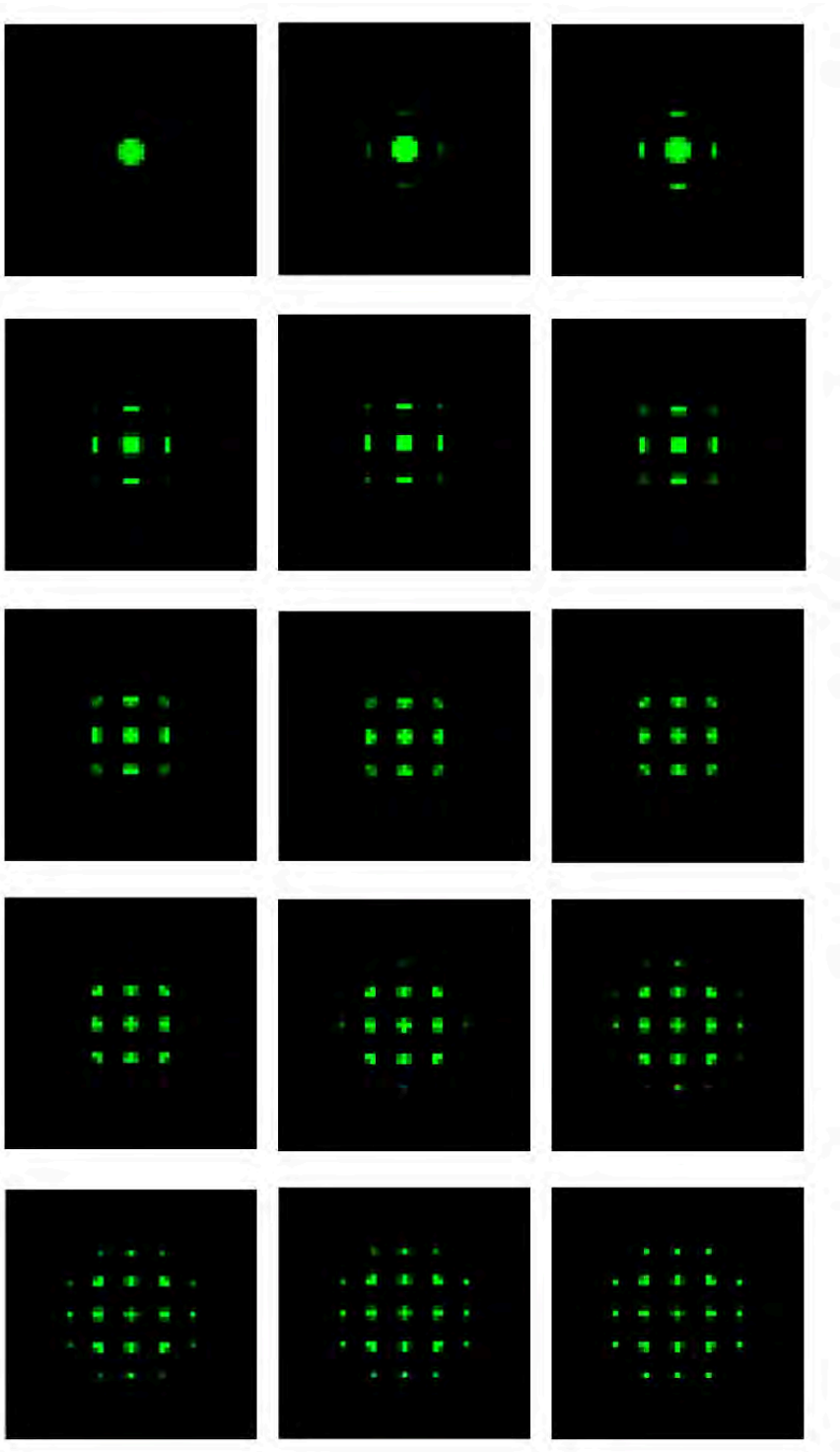


Figure 3: *The effects of a single point being imaged at different defocus positions from 0 to 150 microns*

At non-integer microlens array points of defocus you can see that rays of light will hit multiple pixels. In the majority of these pictures, you can see more than a total of 25 pixels, which means that overlapping is occurring. This means that light from multiple pixels needs to be mapped back to a single pixel. In the diagram below you can see how this occurs without the actual light rays overlapping.

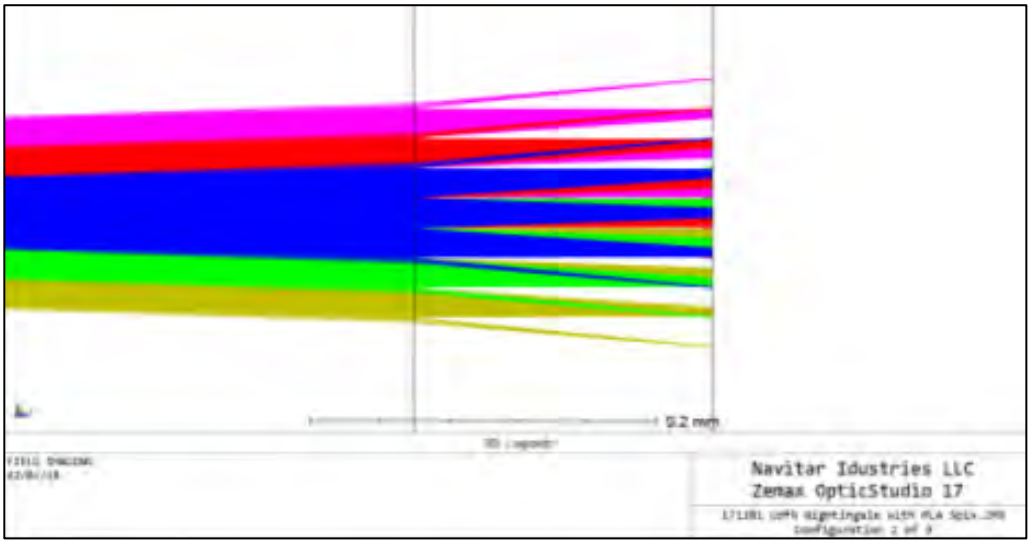


Figure 4: *Ray tracing for non-integer microlens array defocus.*

This image is taken from the Zemax file of our plenoptic system, with the object defocused by 100 microns. At no point on the detector do any of the different points overlap, which follows the principles of plenoptic imaging, but if you consider the size of a pixel it is possible that different points image to the same pixel. The main challenge of doing a fine reconstruction is determining which pixels contain overlap, which microlens they came from, and how much intensity each point contributed to the intensity seen by the pixel.

5.4 Implementing Non-Integer MicrolensArray Reconstruction

The initial reconstruction only worked at integer values of microlens defocus, which does not provide enough detail of the image if the objects are at an intermediate defocus. Using the initial reconstruction algorithm (designed to reconstruct images at 150 μm) we reconstructed images defocused from 150 μm to 100 μm. We saw that the focal ratio (the characterization of this value is detailed in 5.8) of the image dropped to that of the unreconstructed image within 40 μm. This means our reconstruction algorithm needs to take into account non-integer amounts of defocus.

To alter our current algorithm to incorporate non-integer microlens array defocus positions we will assume that each point on the object acts as the center point in terms of how it images to the sensor. To deal with the issue of overlap and different intensities we will be exploiting a quirk of plenoptic imaging, in that the center pixel of a microlens image only receives light from its point except under severe defocus. This will also provide us a limitation to our reconstruction range. By looking at which microlens images have intensity at their center pixel it gives us both a map of which microlens images need to be reconstructed, as well as an intensity comparison between microlens arrays to help deal with overlap. We believe this will also reduce our reconstruction time as this means not having to reconstruct black spots.

5.5 Boundary

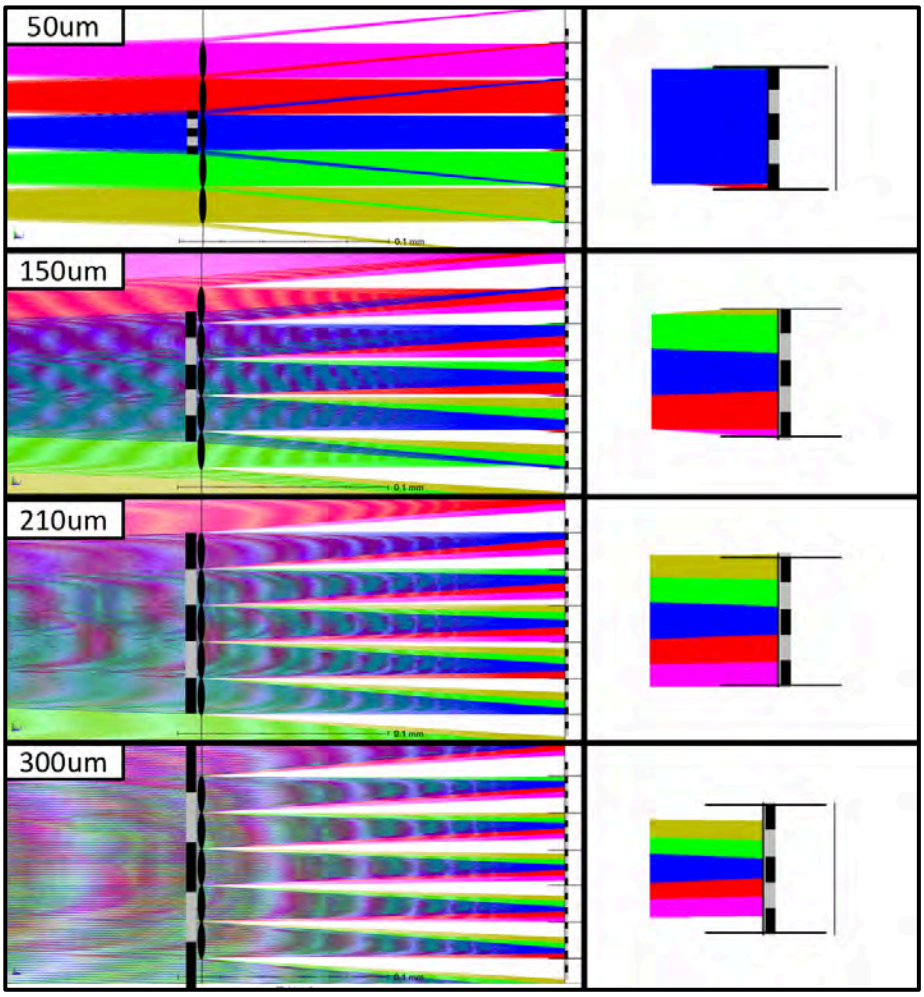


Figure 5: *Ray tracing for different out of focus distances (Left). Zoom-in pictures of center 5 by 5 pixels (Right).*

The above figure is a ray trace plot at different defocus positions (50, 150, 210 and 300 μm). Each picture in the left column is describing five rays that go through five microlens arrays and hit on the detector. The 5x5 pixels are plotted on the detector as 5 rectangles in grey and black. The right column is the zoom-in picture of the center 5x5 pixels. As we can see, at 50 μm, 150 μm, and 210 μm, the center pixel is only covered by the blue ray. However, once beyond 210 μm, the center pixel receives intensity from other points. As shown in the 300 μm ray trace picture, the center pixel has a light

intensity that comes from green, blue and red. Our algorithm currently assumes that the center pixel only has light intensity from one ray. Determining when other points map to the center pixel determines our reconstruction range. Currently, with a center at -40 μm, our boundary is set to be -290 μm to 210 μm.

5.6 100 μm Reconstruction

To alter the current reconstruction algorithm to take into account for the non-integer amounts of defocus, we tasked ourselves to write a program that reconstructs an image at 100 μm of defocus. The reason for this program is to prove a concept and to give us a better understanding of how to generalize the current algorithm. The proof of concept is necessary because we are designing the algorithm around what happens to a single point on the axis, and reconstructing assuming that this point map is shift invariant. This means that a point far off axis will still spread out in a similar enough manner as the center point for us to reconstruct an image. To begin writing this algorithm we hardcoded the reconstruction for each pixel in a microlens array. From there, we worked to generalize a formula that worked for multiple pixels. The resulting code is given below as well as the results from an input image.

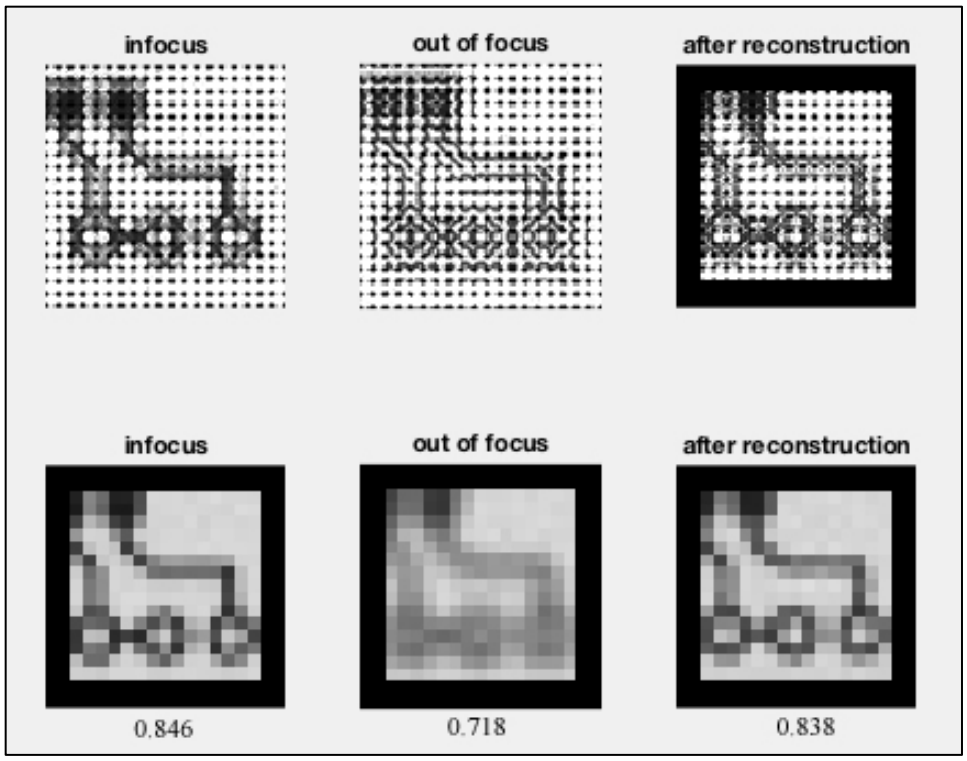


Figure 6: *Before and after reconstruction for 100 μm out of focus. Bottom row is averaged results with Fourier ratio below each picture.*

In figure 7, you can see that our code has successfully reconstructed a depiction of the in-focus image before the averaging process. The black box around the outside of the last image is intentional, as we are looking to only reconstruct the center of the image due to time constraints on our program. In the second set of images, each microlens array has been averaged out, and its focal ratio has been computed. The focal ratio of the in-focus image and the reconstructed image are very close, meaning that our reconstruction algorithm has accurately reconstructed the image.

5.7 0-100μm Out of Focus Reconstruction Evaluation

We tested the reconstruction algorithm for 100μm at defocus values from 0-100μm, with a separation of 10μm. The testing result is shown below where we plotted the Fourier ratio before and after reconstruction with the in-focus Fourier ratio of 0.848. As the

system is further away from the focus, we can see that the Fourier ratio starts to go down. However, after reconstruction, the Fourier ratios are greater than 0.828 which is about 97.6% of 0.848.

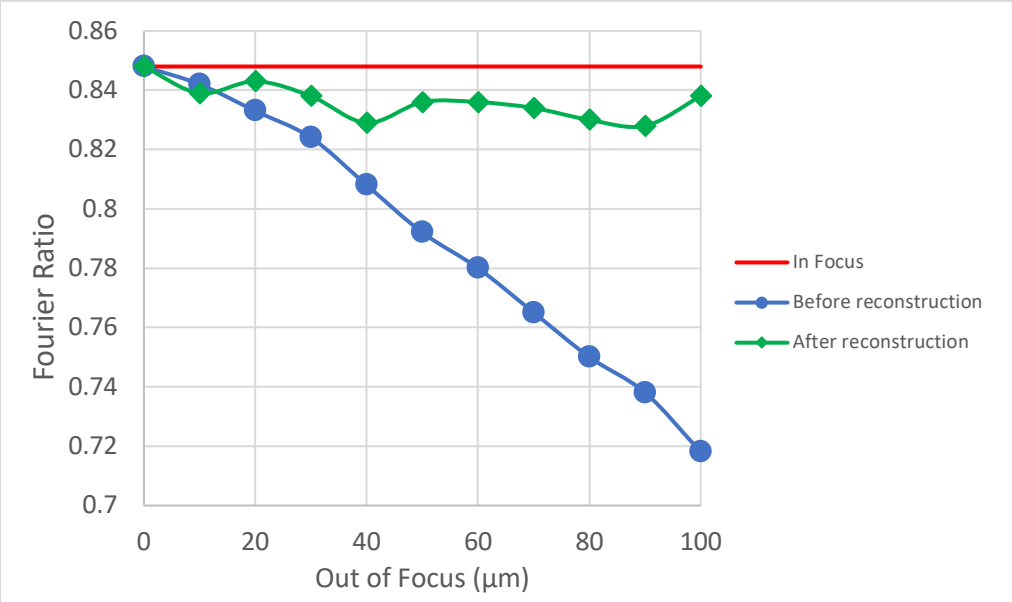


Figure 7: *Fourier ratio for in-focus (Red), before reconstruction (Blue) and after reconstruction (Green) range from 0-100 μm.*

Out of Focus (μm)	Before Reconstruction (Fourier Ratio)	After Reconstruction (Fourier Ratio)
0	0.848	0.848
10	0.842	0.839
20	0.833	0.843
30	0.824	0.838
40	0.808	0.829
50	0.792	0.836
60	0.780	0.836
70	0.765	0.834
80	0.750	0.830
90	0.738	0.828
100	0.718	0.838

Table 2: *Fourier ratio data for in-focus, before reconstruction and after reconstruction range from 0-100 μm.*

The following figure is the output from MATLAB for 100 and 50 μm reconstruction. We can see from the 50 μm averaged photo that the image become much clearer after reconstruction.

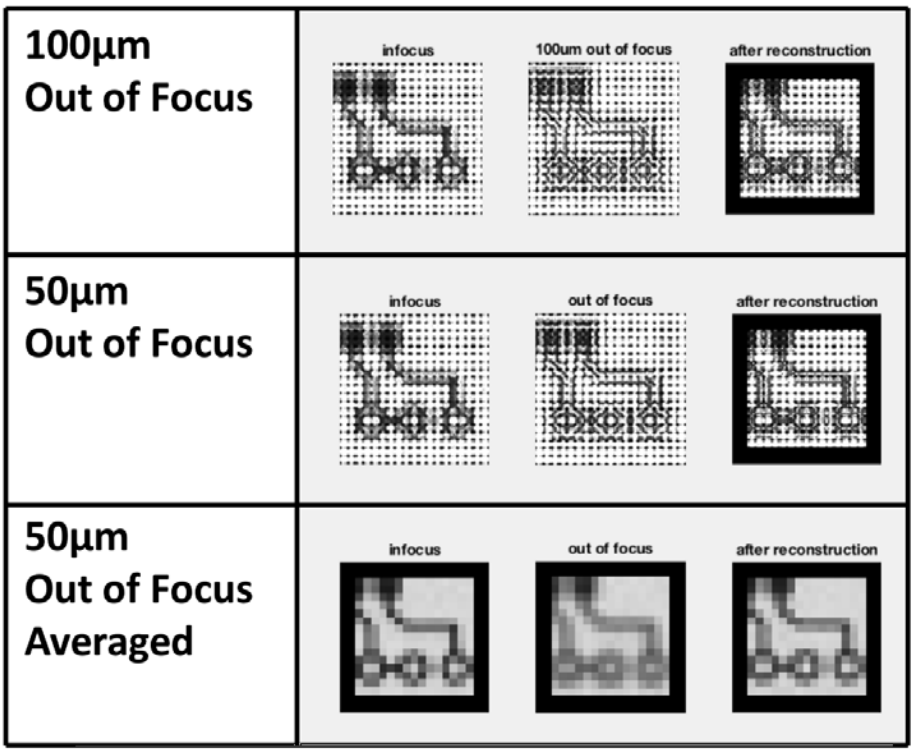


Figure 8: *100 μm reconstruction algorithm applied on a 50 μm out of focus image.*

5.8 Focus Ratio Determination of an Image Using FFT

The current method for determining the quality of focus of an image uses the Fourier image of the reconstructed image, shown on the right. By comparing the intensity of the center to the out regions, the image with a better ratio (one that displays more higher frequency points) is selected. Further adaptations for this code will be testing it on greater samples to better determine the weighting on the ratio and the determination of regions.

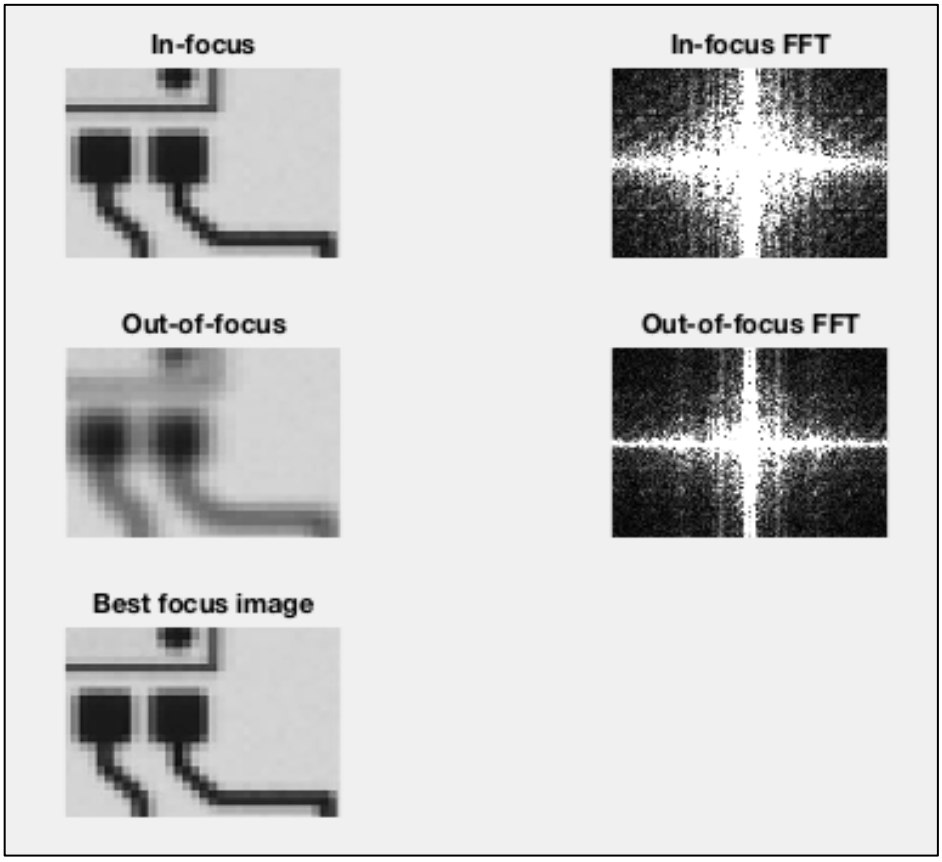


Figure 9: *Sample output for Fourier Ratio method.*

5.9 Point Spread Function Analysis

5.9.1 ZEMAX Simulation Data

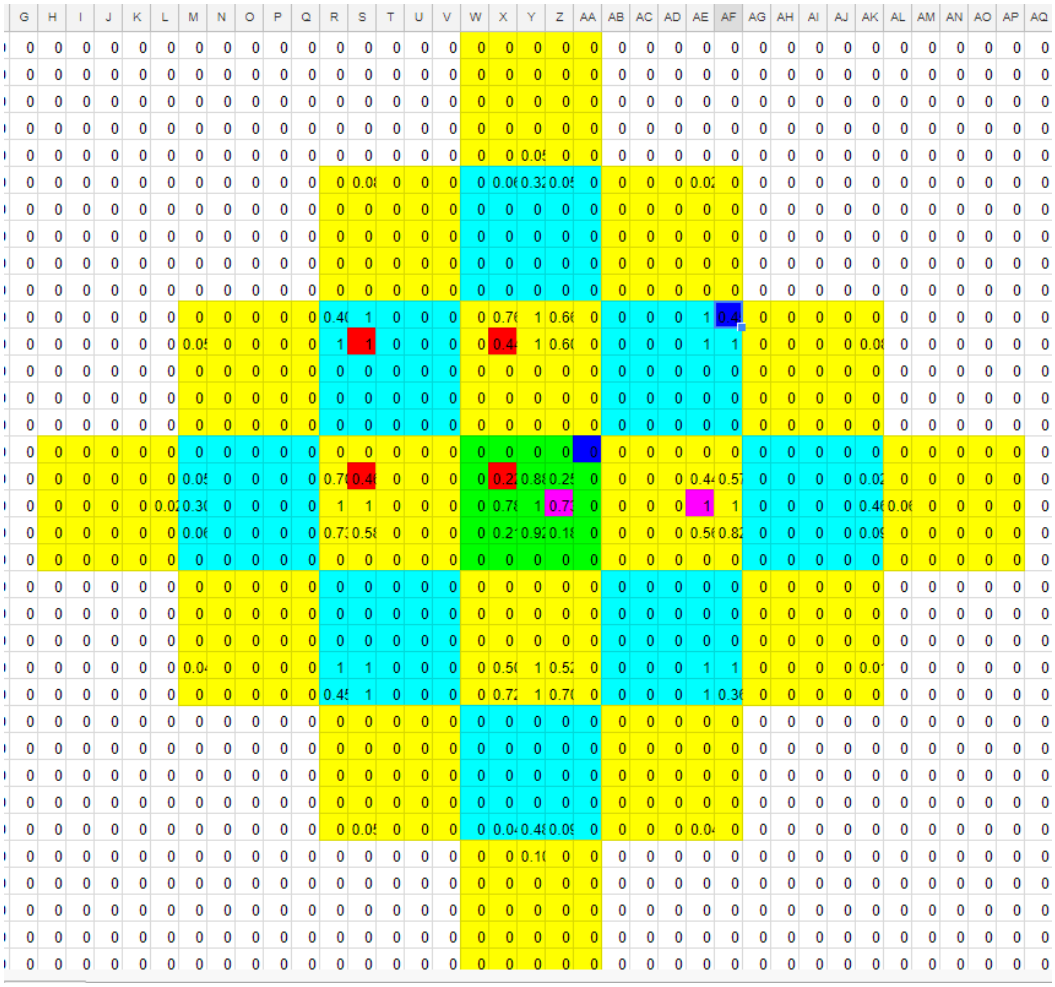


Figure 10: *Point spread data for 100 μm out of focus.*

The figure above shows the values of the 100 μm point spread. The center green area is the microlens to be reconstructed. The light that needs to be imaged back to a pixel in the center microlens array is always in the same location in a different microlens array assuming no aberrations in the system. With this knowledge, all pixels to be reconstructed can be sorted into three groups, those that can be reconstructed from 1 pixel, 2 pixels or four pixels (shown in color in the figure above). The center pixel of each microlens array is guaranteed to only map back to itself. With this, we can use the center pixel as an

intensity ratio to understand what happens at overlapping pixels. For the above figure to reconstruct the pixel one to the right from the center we would use the pink pixels. The intensity that would be mapped to this pixel would be the Intensity seen at a pink pixel time the amount of light hitting it from the center microlens array divided by the amount hitting it from the amount hitting it from the center microlens array plus the light hitting it from the microlens array to the left. This process is then repeated for the second pink pixel. The final result is an averaging formula that allows us to map back intensity at each pixel to a microlens array based on the intensities from the center pixel in each microlens array.

5.9.2 Theoretical Analysis with Excel

Working with our customer we designed an excel program that displays the theoretical point spread function of a given system at a given defocus. The purpose of this program is to manipulate it to alter the characterization of the system and use its output as the text file point spread functions to be input into our algorithm. It works by determining the size of the spot on the microlens, and which pixel in the center microlens it should go back to. From there it sees which microlens the light is currently in and assigns an intensity to the corresponding pixel in that microlens. The intensity value is determined by the ratio of the overlap of each pixel across microlenses, working under the assumption that the light is evenly distributed across each pixel. In the diagram below the black grid is the set of pixels that the light will be mapped back to, and the red grid is the set of microlenses.

System working f/#	10					
Magnification	2					
Pixels per micro lens	5					
Pixel size	0.00345	mm				
Microlens size	0.01725	mm				
MLA ref index	1.5168					
Ray angle +2 max	1.89	°				
Ray angle +2 min	1.13	°				
Ray angle +1 max	1.13	°				
Ray angle +1 min	0.38	°			Adjust	▲
Ray angle 0 max	0.38	°			Defocus	▼
Object defocus	**0.15**	**mm**				
Image defocus	0.91	mm				
Ray position +2 max	0.030	mm	1.737903	µlenses		
Ray position +2 min	0.018	mm	1.042657	µlenses		
Ray position +1 max	0.018	mm	1.042657	µlenses		
Ray position +1 min	0.006	mm	0.347538	µlenses		
Ray position 0 max	0.006	mm	0.347538	µlenses		

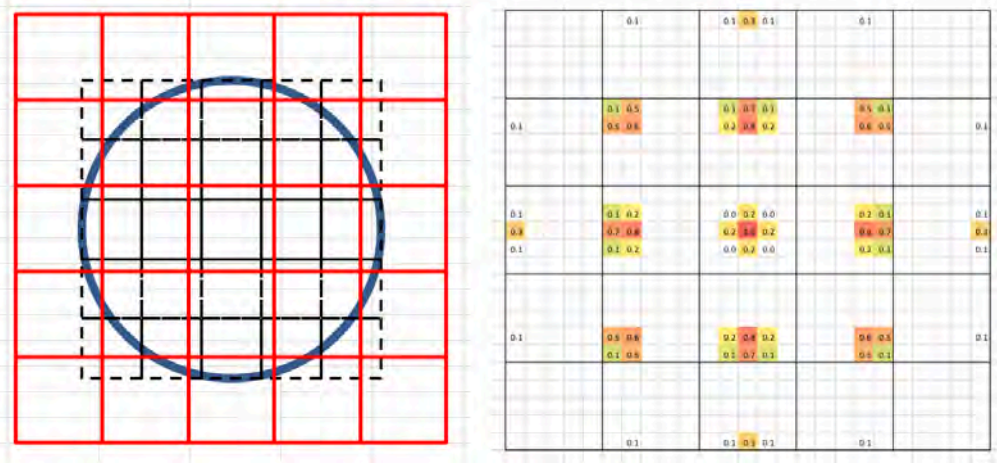


Figure 11: *Sample Excel Output at 150 μm out of focus.*

5.9.3 MATLAB Implementation

To analyze the output of the excel file we wrote a MATLAB script that iteratively searches through each microlens array for intensity values above a given threshold. When it finds a value above a given threshold it places a pixel into the array to be returned by the function. The pixel structure that the array returns is characterized by an intensity, an x component (in relation to the center pixel), and a y component (in relation to the center pixel). The output array is a k x k x 4 array where k is the size of a microlens array and the value at (x, y, z) is one of the pixels that provides intensity to the pixel x, y in the center microlens array. The output array has a maximum of 4 pixels per x, y location as four is the maximum number of locations a pixel in the center microlens array can be dependent on. If a pixel is dependent on fewer pixels then the remaining pixels are zero pixels which have a value of 0,0,0. The benefit to this analysis is that once it has been done, then the array can be stored and will not have to be recomputed until the system is changed and new point spread functions are generated.

5.10 Reconstruction with Point Spread Function

The reconstruction using the point spread function takes an analyzed PSF and the input image. The analyzed PSF gives information about which pixels influence the intensity hitting each location in the microlens array. Due to the bounds of our reconstruction method, the intensity of light hitting the center pixel in the microlens does not have light from other microlens arrays hitting it. This means that we can compare the intensities of different microlens array's center pixels to reconstruct the image. At the boundaries of images, the algorithm implements zero padding. The algorithm iterates through all microlens arrays, creating the reconstructed image. The image then calls the averaging method that averages each microlens array to be one pixel.

6. Final Results

6.1 MATLAB Reconstruction GUI

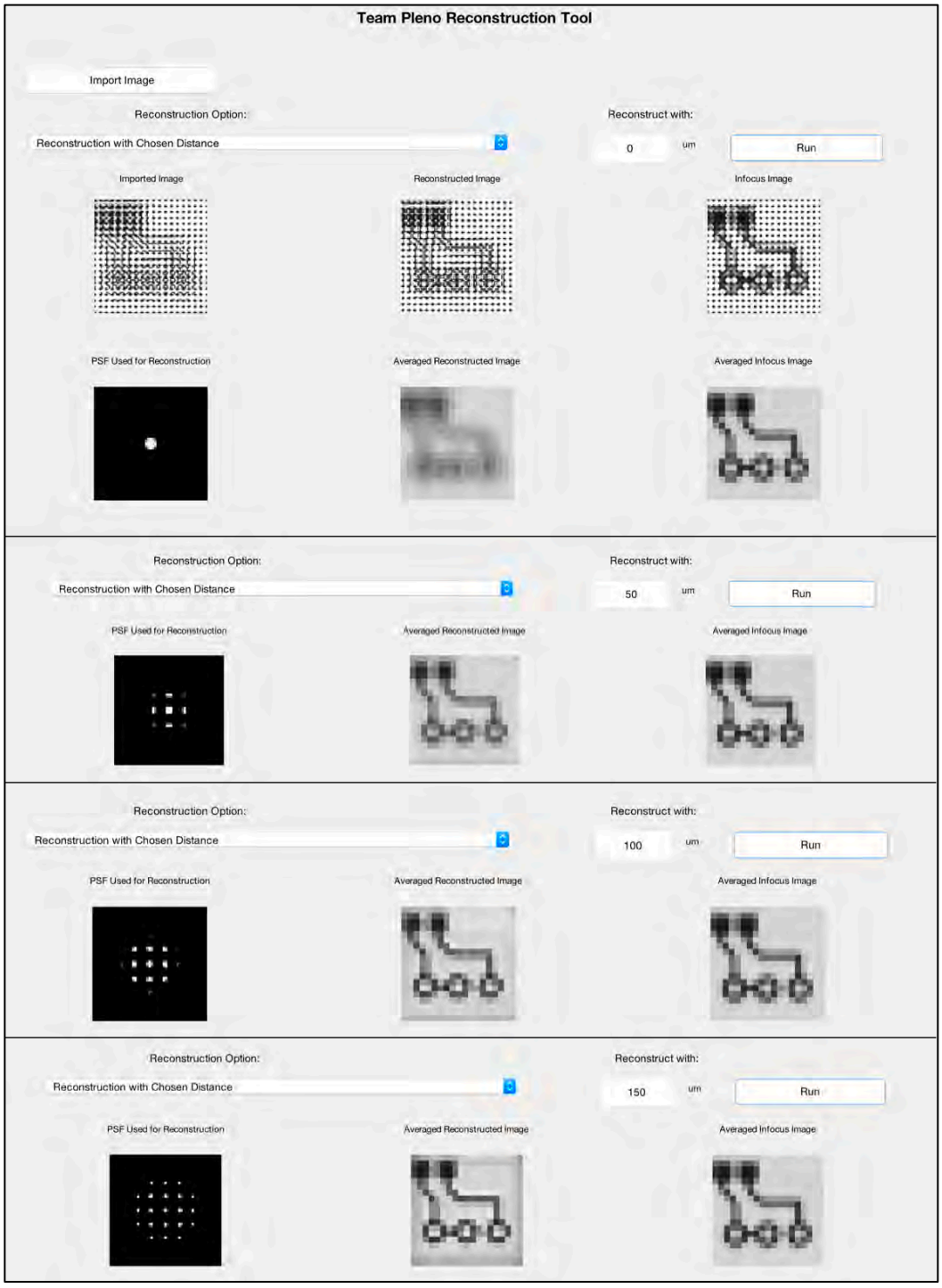


Figure 11: *150 µm defocus reconstructed with 0, 50, 100 and 150 µm PSF.*

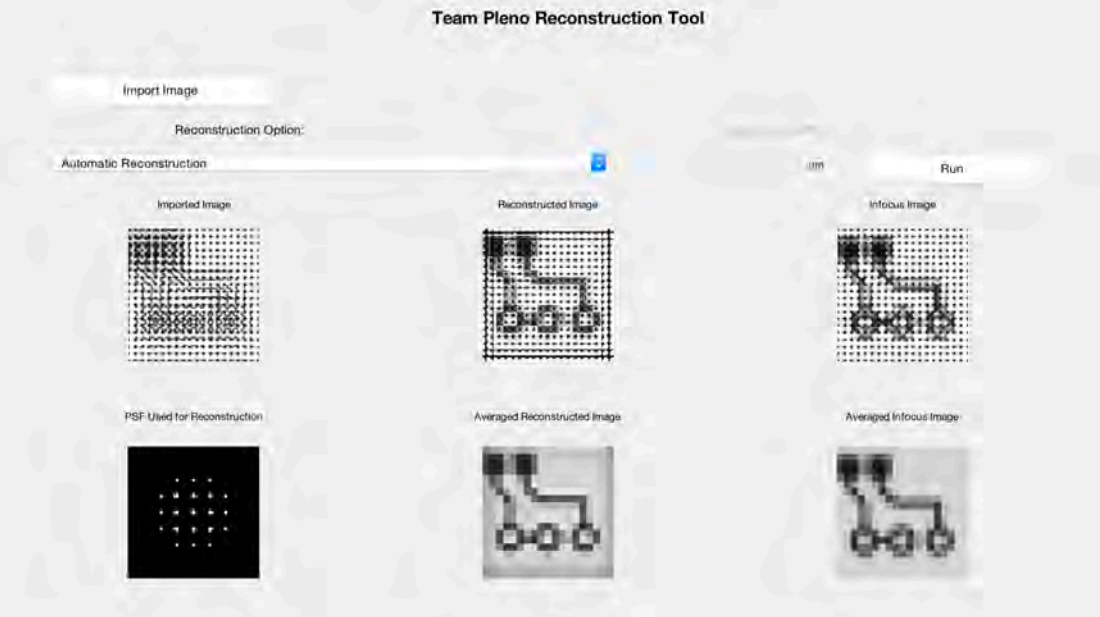


Figure 12: *150 µm defocus reconstructed automatically.*

Our GUI, as shown above, shows the process and steps of the reconstruction. It allows you to choose two reconstruction methods for an imported image at different defocus positions taken from the plenoptic system in ZEMAX. The first (as shown in Figure 12) follows the algorithm described in this paper (see Figure 2, the workflow, for a precise description) and automatically reconstructed the image at best-focus; the second (as shown in Figure 11) allows for a reconstruction at a given amount of defocus. The imported image is shown in the top left, with the used PSF below it. The center two pictures show the reconstructed image, and the averaged reconstructed image. On the right shows the in-focus image for a comparison. Even at high levels of defocus our output image closely resembles that of the in-focus image.

6.2 Design Day Demonstration

On the senior design day, we had a demo showing off the reconstruction algorithm. The setup used our standard ZEMAX file setup with a rotated object so that we could refocus it to different areas of the picture. This allowed people to see the advantages of plenoptic images, as we could alter the focus of an image after it had been taken.

7. Timeline and Future Work

March 2nd	Characterization of a system with function
March 16th	Sensitivity analysis of defocus
March 30th	Point spread function analysis
April 20th	Built the MATLAB reconstruction GUI
April 27th	Project wrap up and created a poster presentation
May 3rd	Navitar tour
May 4th	Design Day demonstration

Table 3: *Timeline of our development process.*

Now that our research is done, and we have determined a method to reconstruct the image. Given the degree of difficulty of reconstructing the image, our customer has talked about pushing back the characterization of the setup to the following year.

The future work to be done on our project is to use our algorithm to characterize the system design, but also to update the algorithm. The algorithm can be updated so that it is more efficient, as the time cost to run it on large images is still drastic. The point spread excel file generation function still needs to be more adaptable as it cannot generate the PSF for images with changing f/#, pixel size, or other inputs well. If that is too drastic a task, a ZEMAX script could be written to generate PSFs with different systems. There is also a better way to average the picture, one that does not average in the black corners which causes a loss of contrast. Lastly, script to determine the bounds of the reconstruction would be useful (determine when the center pixel of the microlens receives light from other microlens arrays), although this may be possible with an equation.

References

[1] Kim, Jonghyun, et al. "Real-time integral imaging system for light field microscopy." Optics Express 22.9 (2014): 10210-10220.

[2] Ng, Ren. Digital light field photography. California: Stanford University, 2006.

[3] Lam, Edmund Y. "Computational photography with plenoptic camera and light field capture: tutorial." JOSA A 32.11 (2015): 2021-2032.

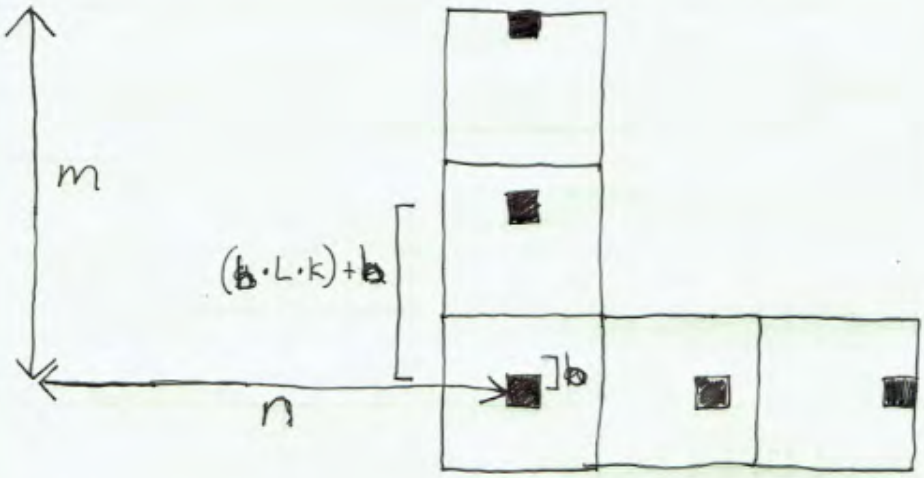
[4] Presentation from our customer, Navitar Inc.

Appendix

1. Initial Image Reconstruction Theory Crafting Pseudo-code

L = the number of pixels in a row in the micro lens image

k = the integer value for the amount of defocus given in micro lens images



Given array of normalized image values
For reconstruction of a single microlens image with coordinates (n, m) for its center pixel, obtain the value of pixels $(n-\lfloor \frac{L}{2} \rfloor, Yval)$ to $(n+\lfloor \frac{L}{2} \rfloor, Yval)$ for Y values $(m-\lfloor \frac{L}{2} \rfloor)$ to $(m+\lfloor \frac{L}{2} \rfloor)$

the value of a pixel im the micro lens image with pixel location (n+a, m+b) can be found in the image at location $(n+(a\cdot L\cdot k)+a,\ m+(b\cdot L\cdot K)+b)$

Reconstruct Image

Arguments:

L the number of pixels in a microlens image

K the defocus given in microlens images (can be negative)

I the array of doubles giving the normalized value of the pixel

Code:

Create a new double array J with width and height of I

For every center pixel in I

J = Reconstruct Micro Lens Image (N, M, L, K, I, J)

Return J

Reconstruct Micro Lens Image

Arguments:

N the X coordinate of the center pixel

M the Y coordinate of the center pixel

L the number of pixels in a microlens image

K the defocus given in microlens images (can be negative)

I the array of doubles giving the normalized value of the pixel

J the array of doubles with the reconstructed values of pixels

Code:

For each row in the microlens image (M – L/2) to (M + L/2)

For each column in the microlens image (N – L/2) to (N + L/2)

Int A = (Current X coor - N)

Int B = (Current Y coor - M)

Set J (current pixel) = Get PixelValue (A, B, N, M, L, K, I)

Return J

Get Pixel Value

Arguments:

A the X coordinate difference from current pixel to the center pixel

B the Y coordinate difference from current pixel to the center pixel

N the X coordinate of the center pixel

M the Y coordinate of the center pixel

L the number of pixels in a microlens image

K the defocus given in microlens images (can be negative)

I the array of doubles giving the normalized value of the pixel

Code:

```
Return I [ N + (A * L * K) + A] [ M + (B * L * K) + B]
```

2. Initial Reconstruction MATLAB Code

```
K = 3; %This is how many microlens array images out we look for the
first pixel from the center pixel
%There will be 2K + 1 images created
L = 5; %This is how many pixels in a row/column in a microlens image

%Taken from the original MATLAB code
PixelWide = 4096;
PixelHeight = 2160;

%change this at some point to be flexible by L
CenterRefocusWidth = 20;
CenterRefocusHeight = 20;

%The lop left center-pixel to be reconstructed
TopLeftX = 48-(CenterRefocusWidth/2);
TopLeftY = 48-(CenterRefocusHeight/2);

load book_150.txt;
I = book_150;
figure; hold on
imshow(I,'InitialMagnification',400);
title('Original image from ZEMAX (+150um)');
print('Original image from ZEMAX (+150um)', '-dpng');
hold off

step = 1;
for k = -1*K:step:K
%From -K to K reconstuct an image assuming a defocus of k
    %iterate through all the area to be reconstructed
    for n = TopLeftX:L:(TopLeftX + CenterRefocusWidth)
        for m = TopLeftY:L:(TopLeftY + CenterRefocusHeight)
          %iterateThrough the microlens array with center pixel at
(n,m)
            for a = -1 * floor(L/2):step:floor(L/2)
                for b = -1 * floor(L/2):step:floor(L/2)
                    temp1 = n + (a*L*K) + a;
                    temp2 = m + (b*L*K) + b;
                    [x,y] = size(I);
                    if (temp1 <= x && temp2 <= y)
                        J(n + a, m + b) = I(temp1, temp2);
                    end
                end
            end
        end
    end
    figure; hold on
    [jx, jy] = size(J);
    imshow(J(TopLeftX-floor(L/2) : jx, TopLeftY-floor(L/2) :
jy),'InitialMagnification',400);
    title(['k= ', num2str(k)]);
    print(['k=', num2str(k)], '-dpng');
    hold off
end
```

3. Fourier Transform Focus MATLAB Code

```
% Load in-focus (I) and out-of-focus (J) images
I=im2double(imread('PCB_in_focus.jpg'));
J=im2double(imread('PCB_out_of_focus.jpg'));

% Show in a subplot top left
subplot(3,2,1); imshow(I)
title('In-focus');

% Show in a subplot bottom left
subplot(3,2,3);imshow(J)
title('Out-of-focus');

% Do FFT of in-focus image
F=fftshift(fft2(I));

% Find magnitude of complex FFT
F=abs(F);

%Scale up values to see image better
F=F*10000/max(F(:));

%Show in a subplot top right
subplot(3,2,2);imshow(F)
title('In-focus FFT');

% Do FFT of out-of-focus image
G=fftshift(fft2(J));

% Find magnitude of complex FFT
G=abs(G);

%Scale up values to see image better
G=G*10000/max(G(:));

%Show in a subplot middel right
subplot(3,2,4);imshow(G)
title('Out-of-focus FFT');

%Allow pixel info to be read
impixelinfo

% Determine number of rows and columns in FFT image
Rows=size(G,1);
Cols=size(G,2);

% Determine sum of all pixel values
SumF=sum(sum(F));
SumG=sum(sum(G));

%Decide what low/high frequency zone we are interested in (proportion
of length and width of image)
Z=0.05;

% Search through in-focus image to sum all low frequency pixels in FFT
PixVal=0;
for R=1:Rows;
```

```
    for C=1:Cols;
        if R>Rows/2*(1-Z)
            if R<Rows/2*(1+Z)
                if C>Cols/2*(1-Z)
                    if C<Cols/2*(1+Z)
                        CurrentPixVal=F(R,C);
                        PixVal = PixVal+CurrentPixVal;
                    end
                end
            end
        end
    end
end
PixVal;
SumF;

%Ratio is proportion of FFT pixels in low F zone
RatioF=PixVal/SumF;

% Search through out-of-focus image to sum all low frequency pixels in
FFT
PixVal=0;
for R=1:Rows;
    for C=1:Cols;
       if R>Rows/2*(1-Z)
            if R<Rows/2*(1+Z)
                if C>Cols/2*(1-Z)
                    if C<Cols/2*(1+Z)
                        CurrentPixVal=G(R,C);
                        PixVal = PixVal+CurrentPixVal;
                    end
                end
            end
        end
    end
end
PixVal;
SumG;

%Ratio is proportion of FFT pixels in low F zone
RatioG=PixVal/SumG;
clc

%Show proportions in high frequency zone
fprintf('In focus high frequency proportion%6.3f\n',1-RatioF);
fprintf('Out of focus high frequency proportion%6.3f\n',1-RatioG);

%Display bottom left the image which Matlab thinks is better focused
if RatioG > RatioF
    subplot(3,2,5);imshow(I)
else
    subplot(3,2,5);imshow(J)
end
title('Best focus image');
```

```
clear;close all
load 100book.txt; %this is the image out of zemax to be reconstructed
I=X100book;

load 100point.txt;%the point distribution of a point imaged at 100um of
defocus
P=X100point;

n = 25; % the X coordinate of the center of the point function
m = 25; % the Y coordinate of the center of the point function

R = zeros(size(I));
step = 5;

for i = 13:step:83
%i is the x coordinate of the center of the current MLA
    for j = 13:step:83
    %j is the y coordinate of the center of the current MLA

        Center = I(i,j);
        if Center~=0
            %if the center pixel has no intensity skip this MLA

            %center
            R(i,j)=I(i,j);

            %corners
            R(i-2,j-2)=+I(i-7,j-7);
            R(i-2,j+2)=+I(i-7,j+7);
            R(i+2,j-2)=+I(i+7,j-7);
            R(i+2,j+2)=+I(i+7,j+7);

            %Reconstruct Pixel Up/Down of Center
            for jj = [-2,-1,1,2]
                sign = jj/abs(jj);
                shiftOne = ((abs(jj)-1)*6+1)*sign;
                shiftTwo = shiftOne + 5*sign;

                PixelOne =
I(i,j+shiftOne)*((Center*P(n,m+shiftOne))/((Center*P(n,m+shiftOne))+(I(
i,j-5*sign)*P(n,m+shiftTwo))));
                PixelTwo =
I(i,j+shiftTwo)*((Center*P(n,m+shiftTwo))/((Center*P(n,m+shiftTwo))+(I(
i,j+5*sign)*P(n,m+shiftOne))));
                R(i,j+jj) =+  PixelOne + PixelTwo;
            end

            %Reconstruct Pixel Left/Right of Center
            for ii = [-2,-1,1,2]
                sign = ii/abs(ii);
                shiftOne = ((abs(ii)-1)*6+1)*sign;
                shiftTwo = shiftOne + 5*sign;

                PixelOne =
I(i+shiftOne,j)*((Center*P(n+shiftOne,m))/((Center*P(n+shiftOne,m))+(I(
i-5*sign,j)*P(n+shiftTwo,m))));
```

4. Point Spread Analysis Code

```
function PSarray = PointSpreadAnalysis(array , k)

% array is the text file of the pointspread function from excel
converted
% into an array of doubles

%k is the number of pixels per microlens array it must be odd

%PSarray is k by k by 4 array where (x, y , n) holds the nth highest
%intensity value from the point spread function corresponding x, y
location
%in all microlens arrays

% the array is #*k by #*k in size

[m,n] = size(array);

threshold = 0.15;

if m ~= n
    disp("error the Point Spread function text file is not square");
end

if (mod(k,2) == 0)
     disp("error the given k value must be odd");
end

%PSarray = zeros(k,k,4);

%create a pixel structure
P = struct('x', {}... %the x coor
           ,'y', {}... %the y coor
           ,'i', {}...%the intensity value
           );

%fill PSarray with zero pixels P(0,0,0)

for x = 1 : 1 : k
    for y = 1 : 1 : k
        for p = 1 : 1 : 4

            %current pixel
            c = 4.*k.*(x - 1) + 4.* (y-1) + p;

            P(c).x = 0;
            P(c).y = 0;
            P(c).i = 0;

            PSarray(x,y,p) = P(c);

        end
    end
end

%iterate through all microlens arrays
for u = 0 : k : n-k- 1
```

```
    for v = 0 : k : m-k- 1

        %iterate through each microlens array
        for x = 1 : 1 : k
            for y = 1 : 1 :k


                int = array(u + x, v + y); %set the intensity
                pos = 4.*k.*(x - 1) + 4.* (y-1);
                %current point in array is given by (u + x, v + y)

                if int > threshold
                    if  PSarray(x, y, 1).i == 0 %if the first pixel has
not been set - do so

                        PSarray(x, y, 1).x = u + x;
                        PSarray(x, y, 1).y = v + y;
                        PSarray(x, y, 1).i = int;

                    else if PSarray(x , y , 2).i == 0 %if the second
pixel has not been set do so

                            PSarray(x, y, 2).x = u + x;
                            PSarray(x, y, 2).y = v + y;
                            PSarray(x, y, 2).i = int;

                        else if PSarray(pos + 3).i == 0 %if the third
pixel has not been set do so

                                PSarray(x, y, 3).x = u + x;
                                PSarray(x, y, 3).y = v + y;
                                PSarray(x, y, 3).i = int;

                            else if PSarray(pos + 4).i == 0 %if the
fourth pixel has not been set do so

                                    PSarray(x, y, 4).x = u + x;
                                    PSarray(x, y, 4).y = v + y;
                                    PSarray(x, y, 4).i = int;
                                else
                                    disp("error too many pixels taken
increase threshold");
                                end
                            end
                        end
                    end
                end
            end
        end

    end
end
```

5. Reconstruction with Point Spread Function

```
function [R,JR,FFTvalue] = ReconstructionPSF(Ip, P, k)

%R is the final reconstructed image
%Ip is the input image
%P is the point spread array
%k x k is the number of pixels per microlens array

[n,m] = size(Ip);
R = [n,m];

%  disp( "n " + n + " m " + m);


%add zero padding around the image I

padding = 50; %amount of padding per side of the image

I = zeros(n + 2.* padding, m + 2.* padding);

for a = 1 : 1 : n
    for b =  1 : 1 : m

        I(a + padding, b + padding) = Ip(a,b);

    end
end

%iterate through all the microlens arrays
for u = padding : k : padding + n - k
    for v = padding : k : padding + m - k

        %iterate through all the pixels in a microlens array
        for x = 1 : 1 : k
            for y = 1: 1 : k

                    %set the current pixel in I to equal the sum of the
                    %pixels around it determined by the PSF

                    %current location in the image is given by
                    %(u + x, v + y)

                    %current location in the microlens is given by (x,
y)

                    %the current pixel is given by p

                    if P(x, y, 1).i ~= 0 %checks to see if all the
pixels are nonzero pixels

                        I1 = P(x,y,1).i;
                        I2 = P(x,y,2).i;
                        I3 = P(x,y,3).i;
```

```
                        I4 = P(x,y,4).i;

                        X1 = P(x,y,1).x;
                        X2 = P(x,y,2).x;
                        X3 = P(x,y,3).x;
                        X4 = P(x,y,4).x;

                        Y1 = P(x,y,1).y;
                        Y2 = P(x,y,2).y;
                        Y3 = P(x,y,3).y;
                        Y4 = P(x,y,4).y;

                        %set values for the x and y coordinates of the
                        %center pixel of the microlens being
reconstructed
                        CPX = ceil(k/2) + u;
                        CPY = ceil(k/2) + v;

%                         disp( "X1 " + X1 + " Y1 " + Y1 + " CPY " +
CPY +  " CPX " + CPX);

                        %set pixel 1
                        bottom1 = ((I(CPX,CPY) .* I1) + (I(CPX + X1 -
X2, CPY + Y1 - Y2) .* I2) + (I(CPX + X1 - X3, CPY + Y1 - Y3) .* I3) +
(I(CPX + X1 - X4, CPY + Y1 - Y4) .* I4));

                        if bottom1 ~= 0%avoid divide by 0 errors
                        intensity1 = I(CPX - X1, CPY - Y1) .* I(CPX,
CPY) .* I1 ./ bottom1;
                        else
                        intensity1 = 0;
                        end

                        %set pixel 2
                        bottom2 = ((I(CPX,CPY) .* I2) + (I(CPX + X2 -
X1, CPY + Y2 - Y1) .* I1) + (I(CPX + X2 - X3, CPY + Y2 - Y3) .* I3) +
(I(CPX + X2 - X4, CPY + Y2 - Y4) .* I4));

                        if bottom2 ~= 0 %avoid divide by 0 errors
                        intensity2 = I(CPX - X2, CPY - Y2) .* I(CPX,
CPY) .* I2 ./ bottom2;
                        else
                        intensity2 = 0;
                        end

                        %set pixel 3
                        bottom3 = ((I(CPX,CPY) .* I3) + (I(CPX + X3 -
X1, CPY + Y3 - Y1) .* I1) + (I(CPX + X3 - X2, CPY + Y3 - Y2) .* I2) +
(I(CPX + X3 - X4, CPY + Y3 - Y4) .* I4));

                        if bottom3 ~= 0
                        intensity3 = I(CPX - X3, CPY - Y3) .* I(CPX,
CPY) .* I3 ./ bottom3;
                        else
                        intensity3 = 0;
                        end
```

```
                        %set pixel 4
                        bottom4 = ((I(CPX,CPY) .* I4) + (I(CPX + X4 -
X1, CPY + Y4 - Y1) .* I1) + (I(CPX + X4 - X2, CPY + Y4 - Y2) .* I2) +
(I(CPX + X4 - X3, CPY + Y4 - Y3) .* I3));

                        if bottom4 ~= 0
                        intensity4 = I(CPX - X4, CPY - Y4) .* I(CPX,
CPY) .* I4 ./ bottom4;
                        else
                        intensity4 = 0;
                        end

%                         disp("intensity1 " + intensity1 + "
intensity2 " + intensity2 + " intensity3 " + intensity3 + " intensity4
" + intensity4);

                        %set the intensity of the current pixel
                        R(u + x - padding, v + y - padding) =
intensity1 + intensity2 + intensity3 + intensity4;
                    else
                    R(u + x - padding, v + y - padding) = 0;

                    end
            end
        end
    end
end

% average
JR = zeros(size(R));
for r=1:k:n-k+1
    for c=1:k:m-k+1
        Val=0;
        for mlar=r:r+k-1
            for mlac=c:c+k-1
                Val= Val+R(mlar,mlac);
                JR(r,c)=Val/(k*k);
            end
        end
    end
end
for r=1:k:n-k+1
    for c=1:k:m-k+1
        for mlar=r:r+k-1
            for mlac=c:c+k-1
                JR(mlar,mlac)=JR(r,c);
            end
        end
    end
end

FFTvalue = FFT(JR);
end
```

6. GUI

```
function Reconstruction_GUI
k = 5;

f=figure(1);
bgcolor = f.Color;


UItitle = uicontrol('Parent',f,...
'Style','text',...
'Position',[40,394,500,23],...
'Unit','normalized',...
'String','Team Pleno Reconstruction Tool',...
'FontWeight','Bold',...
'FontSize',20,...
'BackgroundColor',bgcolor);

%% image import

Import = uicontrol('Parent',f,...
'Style','pushbutton',...
'Position',[60,350,100,23],...
'Unit','normalized',...
'String','Import Image',...
'BackgroundColor',bgcolor,...
'FontSize',14,...
'Callback',@Import_callback);

    function Import_callback(Import, eventdata, handles)
         global Img

        % import image
        [filename, pathname]=uigetfile({'*.*'},'File Selector');
        [~,~,ext]=fileparts(filename);
        filename = strcat(pathname,filename);

        if strcmp(ext,'.xls')==1
            Img=xlsread(filename);
        elseif strcmp(ext,'.xlsx')==1
            Img=xlsread(filename);
        elseif strcmp(ext,'.txt')==1
            Img=textread(filename);
        end


    end

%% static components
rec_type = uicontrol('Parent',f,...
'Style','text',...
'Position',[70,320,150,23],...
'String','Reconstruction Option:',...
'Unit','normalized',...
'FontSize',14,...
```

```
'BackgroundColor',bgcolor);

rec_distance = uicontrol('Parent',f,...
'Style','text',...
'Position',[330,320,100,23],...
'Unit','normalized',...
'String','Reconstruct with:',...
'BackgroundColor',bgcolor,...
'FontSize',14,...
'enable','off');

% unit text box
unittext = uicontrol('Parent',f,...
'Style','text',...
'Position',[375,305,50,15],...
'Unit','normalized',...
'String','um',...
'FontSize',12,...
'BackgroundColor',bgcolor);

% subtitle text boxes
text1 = uicontrol('Parent',f,...
'Style','text',...
'Position',[50,280,150,15],...
'Unit','normalized',...
'String','Imported Image',...
'FontSize',12,...
'BackgroundColor',bgcolor);

text2 = uicontrol('Parent',f,...
'Style','text',...
'Position',[206,280,150,15],...
'Unit','normalized',...
'String','Reconstructed Image',...
'FontSize',12,...
'BackgroundColor',bgcolor);

text3 = uicontrol('Parent',f,...
'Style','text',...
'Position',[363,280,150,15],...
'Unit','normalized',...
'String','Infocus Image',...
'FontSize',12,...
'BackgroundColor',bgcolor);

text4 = uicontrol('Parent',f,...
'Style','text',...
'Position',[50,145,150,15],...
'Unit','normalized',...
'String','PSF Used for Reconstruction',...
'FontSize',12,...
'BackgroundColor',bgcolor);

text5 = uicontrol('Parent',f,...
'Style','text',...
'Position',[206,145,150,15],...
```

```
'Unit','normalized',...
'String','Averaged Reconstructed Image',...
'FontSize',12,...
'BackgroundColor',bgcolor);

text6 = uicontrol('Parent',f,...
'Style','text',...
'Position',[363,145,150,15],...
'Unit','normalized',...
'String','Averaged Infocus Image',...
'FontSize',12,...
'BackgroundColor',bgcolor);

% sets of axes for plotting
ax1 = axes('Parent',f,...
'Position',[0.12 0.45 0.2 0.2]);
ax2 = axes('Parent',f,...
'Position',[0.12 0.12 0.2 0.2]);
ax3 = axes('Parent',f,...
'Position',[0.4 0.45 0.2 0.2]);
ax4 = axes('Parent',f,...
'Position',[0.4 0.12 0.2 0.2]);
ax5 = axes('Parent',f,...
'Position',[0.68 0.45 0.2 0.2]);
ax6 = axes('Parent',f,...
'Position',[0.68 0.12 0.2 0.2]);

%% editable text boxes

% input reconstruction distance
distedit = uicontrol('Parent',f,...
'Style','edit',...
'Position',[350,300,40,23],...
'Unit','normalized',...
'string','',...
'BackgroundColor',bgcolor,...
'FontSize',14,...
'enable','off');

%% pop-up menu

% start a popup menu
popup = uicontrol('Parent',f,...
'Style','popup',...
'Position',[60,300,250,23],...
'Unit','normalized',...
'FontSize',14,...
'String',{'Automatic Reconstruction','Reconstruction with Chosen
Distance'},...
'BackgroundColor',bgcolor,...
'Callback', @popupcallback);

function popupcallback(popup, eventdata, handles)

switch get(popup,'Value')
    case 1
```

```
        set(distedit,'Enable','off')
        set(rec_distance,'Enable','off')
    case 2
        set(rec_distance,'Enable','on')
        set(distedit,'Enable','on')
end
end

%% show the plot

runpush = uicontrol('Parent',f,...
'Style','pushbutton',...
'Position',[420,300,80,23],...
'String','Run',...
'BackgroundColor',bgcolor,...
'FontSize',14,...
'Unit','normalized',...
'Callback',@runpushcallback);

    function runpushcallback(runpush, eventdata, handles)
        % clear variables

        % get data
         global Img
        v=get(popup,'Value') ;

        if v==1 % Fourier Guess

            % loop through the 8 possible out-of-focus distances to
find the best focus
            FFTs = zeros(1,15);
            for i = 1:1:15
                d1=strcat(int2str(i),'0point.txt');
                d1 = strcat('data/',d1);
                p=load(d1);
                [~,~,FFTvalue] = ReconstructionPSF(Img,
PointSpreadAnalysis(p , k), k);
                FFTs(1,i) = FFTvalue;
            end

            [~, idx] = max(FFTs);

            disp(idx);
            d1=strcat(int2str(idx),'0point.txt');
            d1 = strcat('data/',d1);
            P=load(d1);
            [R,JR,FFTvalue] =
ReconstructionPSF(Img,PointSpreadAnalysis(P , k),k);
            imshow(Img, 'parent',ax1);
            imshow(P, 'parent',ax2);
            imshow(R, 'parent',ax3);
            imshow(JR, 'parent',ax4);

            disp(FFTvalue);
```

```
        elseif v==2 % Fixed Distance
            d=get(distedit,'String');
            d1=strcat(d,'point.txt');
            d1 = strcat('data/',d1);

            P=load(d1);
            [R,JR,FFTvalue] =
ReconstructionPSF(Img,PointSpreadAnalysis(P , k),k);

            imshow(Img, 'parent',ax1);
            imshow(P, 'parent',ax2);
            imshow(R, 'parent',ax3);
            imshow(JR, 'parent',ax4);

            disp(FFTvalue);

        end

        % plot in focus images
        infocus = load('data/0book.txt');
        imshow(infocus, 'parent',ax5);
        infocus_ave = imread('data/infocus.png');
        imshow(infocus_ave, 'parent',ax6);


    end

end
```