Report-

Experiment of Correcting High Order Aberration

Zhao Fu

Advisor: Brian Barsky

1 Overview on Optical System

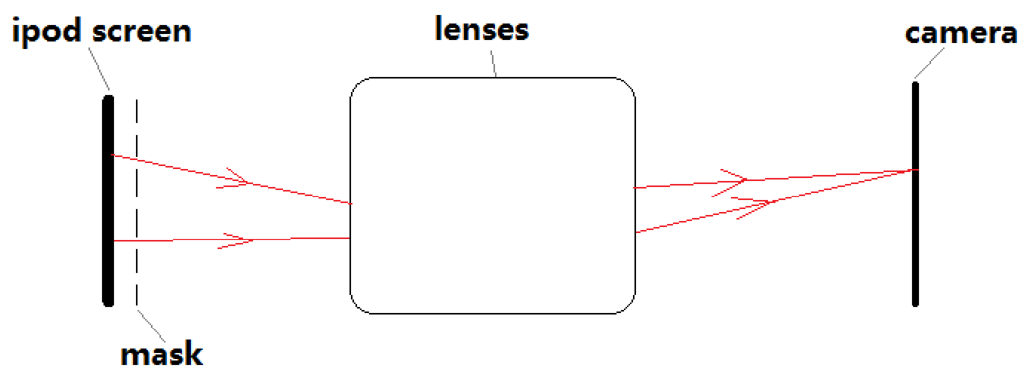


figure 1.1

We use an optical system to simulate our eye. The camera is like our retina and the lenses are to simulate the lens in our eye, as showed in figure 1.1. The mask is an opaque board with little pin holes aligned in rectangular matrix on it and the distance between two adjacent pin holes are carefully measured. When put the mask on the iPod screen, rays are emitted from the iPod pixels through specific pin holes on the mask, which means that these rays are fixed in specific directions so that all the rays emitted from the iPod are calculable. After going through the lenses, each ray will reach a specific pixel on the camera. From the view of the camera side, one pixel on the camera can receive several rays emitted from different pixels on the iPod, which means the color of this pixel on the camera is the sum of the colors of those pixels on the iPod if lightness is ignored, as shown in the following equation

$$\mathrm{color}_{\mathrm{camera}_i} = \sum \mathrm{color}_{\mathrm{ipod}_j} = \mathrm{A}_i\boldsymbol{x},$$

Where $\boldsymbol{x}$ is a vector containing the values of all the pixels on the iPod and A_i is a coefficient vector containing 0s and 1s. Thus we can combine all these equations together to calculate the values of each pixel on the iPod by solving the combined equations

$$\mathrm{A}\boldsymbol{x} = \mathrm{B},$$

Where

$$\mathrm{A} = \mathrm{vector}(\mathrm{A}_i)^{\mathrm{T}}$$

And

$$B = \mathrm{vector}\left(\mathrm{color}_{\mathrm{camera}_i}\right)^{T}.$$

In this way, we can get the proper image on the iPod, which is called the compensated image.

2 Algorithm and Parameters

2.1 Composition of coefficient matrix A

The algorithm of the previous version of the code is to calculate the back propagation of each ray from the camera side to the iPod side. It samples more than 6000 rays going through the aperture emitted by each pixel on the camera. When each ray reaches the mask, this algorithm ignores the size of the holes on the mask and just supposes that the size of the holes are large enough to let all the rays go through. In this way, the distribution of the rays from one pixel on the camera reaching the ipod screen is estimated by the calculation. So the coefficient matrix A is calculated.

This previous method is time consuming because the number of samples is too large and the complexity of the algorithm depends on the number of pixels calculated on the camera, also called the resolution of the image on the camera. Besides, when using the calculated matrix A to simulate the system to reconstruct the original image, the result is not very similar to the reality, which means the simulation is not good enough when using the previous algorithm.

I improve the algorithm just by reversing the direction of calculation of the rays. I calculate the rays emitted from each hole on the mask to the camera. This method has to sample the angles of rays emitted from one hole. Since the size of the aperture is known, the maximum angle of the rays can be calculated. So it just takes into consideration the pixels on the ipod screen that are inside the maximum angle. In this way, the complexity of the algorithm just depends on the resolution of the mask and the maximum angle of the rays that can go through the aperture. So we can increase the resolution of the image calculated on the camera to make the simulation more similar to the reality.

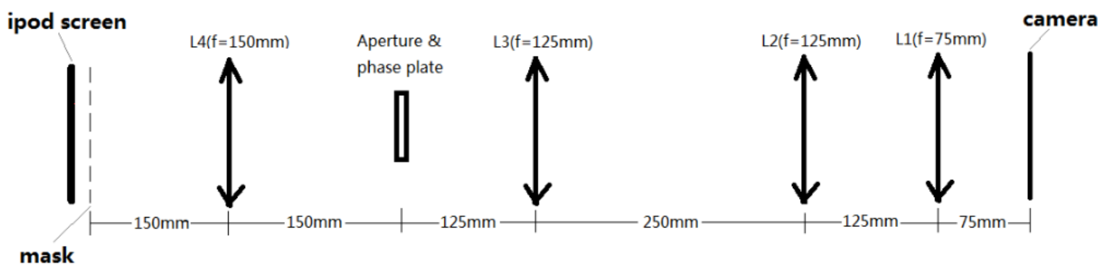


figure 2.1

Then comes the details of the designation. The detailed optical system is shown in figure 2.1. Since we use derivatives of Zernike polynomials to express the wavefront of the parallel rays after going through the phase plate, the calculation seems easier if we make the rays in front of the phase plate become parallel rays. This is why L4 is put between the ipod screen and the phase plate. I put the mask right at the focal length of L4 so that the rays in front of the phase plate become parallel. Then the way to simulate an abnormal eye is to combine a normal convex lens with a coma phase plate. L3 and L2 are two same lenses whose function is to copy the light rays after the phase plate to the rays in front of L1, making L1 and phase plate together have the same effect as one combined lens. It's because our apparatus doesn't allow us to make the phase plate and L1 close enough that we have to use this designation.

As for the parameters, when there is no phase plate and the other apparatuses are all set according to figure 2.1, the mask will be well focused and there will be a sharp image on the camera. When put the phase plate on and without other changes, the aberration will be too small to correct. To magnify the effect of the phase plate, I add some low order aberration in it. So I set the distance between L1 and the camera to 70mm so that the mask is inside focus. Next question is how to get the coefficient of the derivatives of Zernike polynomials. The coefficient of coma comes from an excel given by Ram (the postdoc) while the coefficient of defocus is calculated by the formula

$$c_4 = -\frac{\text{aperture}}{4\sqrt{3}f},$$

Where f is the focal length of L1 which is 75mm and aperture is the diameter of the aperture which is 6mm. We can easily derive this formula from the lens image formation rule.

After setting all these parameters properly, we can run our code to get the coefficient matrix A.

2.2 Solve the equations

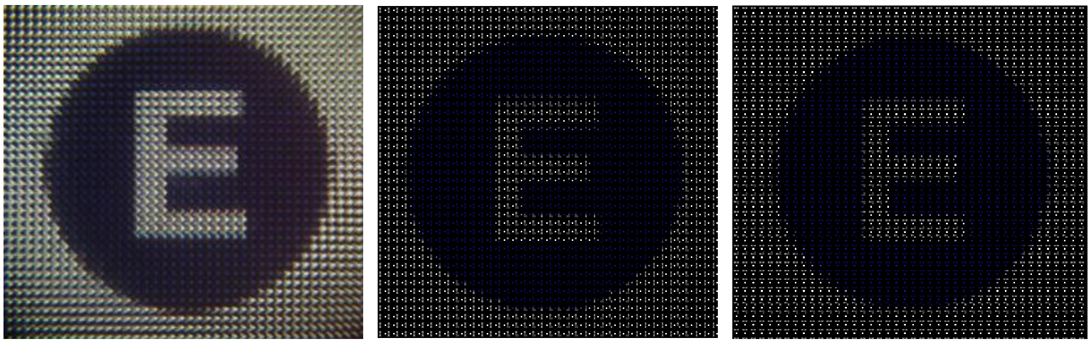
After we get the equations $Ax = B$, we can find that the number of rows of matrix A is smaller than the number of columns of it, which means that the number of equations is fewer than the number of unknown numbers. So we have to add constraints to it and use optimization method to solve it. It is obvious that there are many pixels on the iPod cannot emit rays reach the camera, so the colors of these pixels don't matter. Since we don't want these pixels to influence the experiment, we expect the colors of such pixels to be black, which means the values of these pixels to be zeros. So we add $\boldsymbol{x} = 0$ to previous equations, making A become $\binom{A}{I}$. In this way, the number of equations becomes more than the number of unknown numbers. The previous version of the code used LBFGSB lib to iteratively solve the equations, but the speed is too slow. In order to solve the equations in real time, I use linear regression method to solve them. First, I transform the equations to

$$A^T A\boldsymbol{x} = A^T B.$$

Then we can easily calculate $(A^T A)^{-1}$ in preprocess so that we could use $\boldsymbol{x} = (A^T A)^{-1} A^T B$ to calculate $\boldsymbol{x}$ in real time.

2.3 Verification of the algorithm

The verification method is very important when debugging the code. When I put a sharp image on the iPod with phase plate and without defocus, I will see the real image on the camera like shown in Figure 2.2(a), from which you can see the coma effect is horizontal. But in figure 2.2(b), you can see that the coma effect is vertical, which means the x-y coordinate is wrong. After changing the coordinate, the simulated image is like figure 2.2(c), from which you can see the coma effect is just similar to that in figure 2.2(a). After simulation, I can make sure that the code has no critical bugs.



(a) (b) (c)

figure 2.2

3 Experiment

3.1 Apparatus

L1: $F = 75\text{mm}$;

L2: $F = 125\text{mm}$;

L3: $F = 125\text{mm}$;

L4: $F = 150\text{mm}$;

Aperture: $\text{Diameter} = 6\text{mm}$;

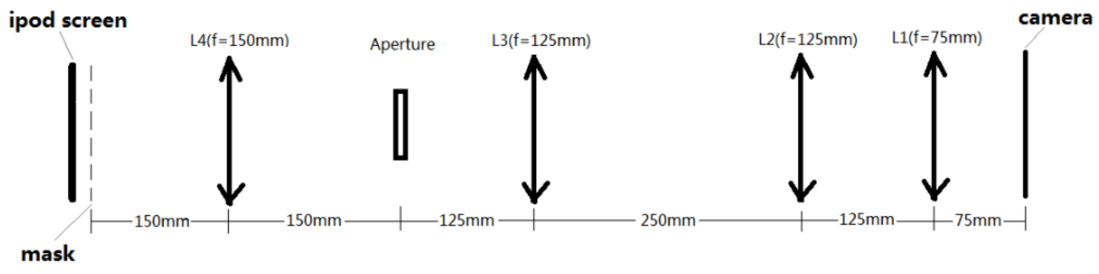
Phase plate: $C_1 = -0.643199,\ C_7 = 0.629738$;

Camera;

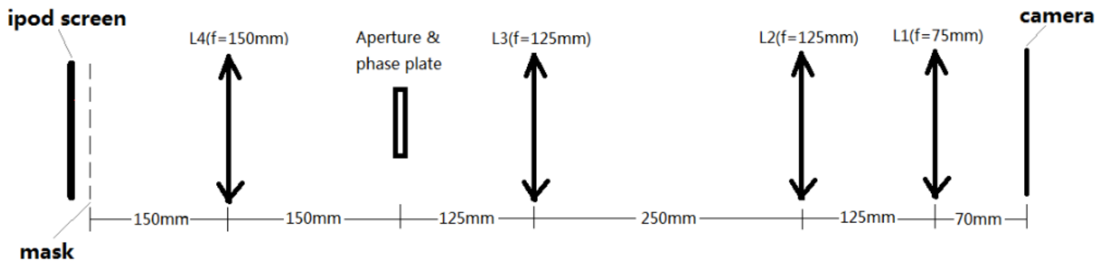
iPod: $\text{resolution} = 640 \times 640$;

mask: $\text{resolution} = 128 \times 128$.

3.2 Step



(a)



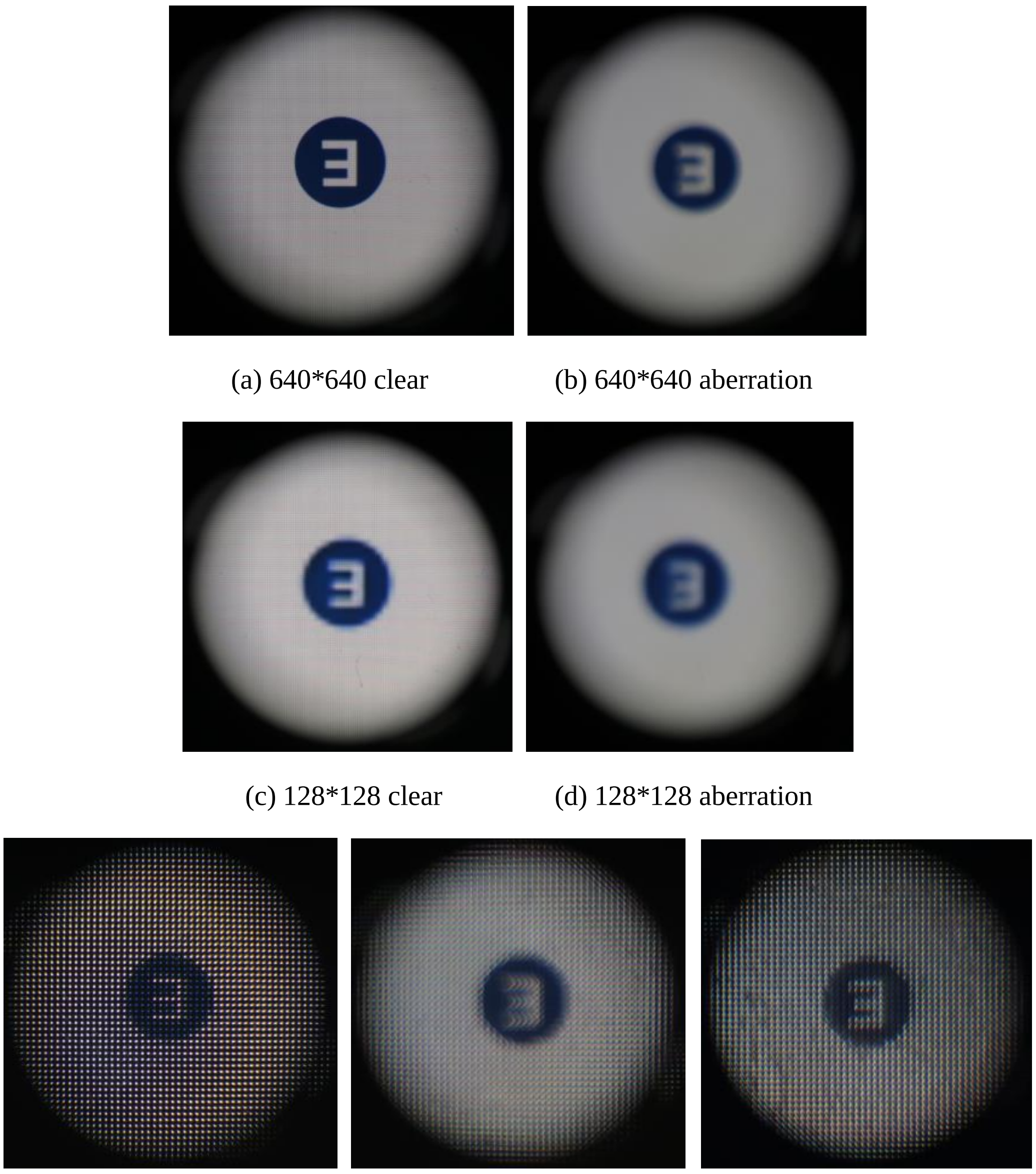
(b)

figure 3

First, we should put the camera at the focal length in front of L1 and put the original image on the iPod with mask on it. Then we adjust the distance between iPod and L4 until the image on the camera is clear enough in order to ensure that the distance is the focal length of L4. When doing this, we should take off the phase plate, as shown in figure 3(a). To make it convenient to observe the result, we connect the camera to the laptop to make the image appear on the laptop using the software named EOS Utility. After that, we put the camera at 70mm in front of L1 to add some defocus effect to the system and put on the phase plate to add some coma effect to the system and take a photo of the blurred image, as shown in figure 3(b). Finally we put the compensated image on the iPod and adjust the iPod at the same distance but with different angles to make the image on the camera clear enough and then take a photo of the corrected image.

There is some explanation of why adding some defocus effect to the system. The coma effect of the phase plate is too little to make the image blur when put a mask with resolution of 128*128 on the iPod. In other word, only if the resolution of mask is large enough can the coma effect make the adjacent pixels overlap and make the image blur. As a result, we try to add some defocus effect to it to make the original image become blur so that we can correct it with our method. Now we are still trying to prove that our method is correct not only with defocus (low order) aberration but also with coma (high order) aberration.

4 Result



(a) 640*640 clear (b) 640*640 aberration

(c) 128*128 clear (d) 128*128 aberration

(e) mask clear (f) mask aberration (g) compensate aberration

figure 4.1

Figure 4.1 shows the real photos taken by our camera. First we define that ‘with aberration’ means there is both coma and defocus effect on the image. (a) is the original image of 640*640 resolution without aberration. (b) is the original image of 640*640 with aberration. (c) is the original image of 128*128 without aberration. (d) is the original image of 128*128 with aberration. (e) is the original image of 640*640 with mask of 128*128 on it and without aberration. (f) is the original image of 640*640 with mask of 128*128 on it and with aberration. (g) is the compensated image of 640*640 with mask of 128*128 on it and with aberration.

From these photos, you can see the compensated image is as clear as the original image of 128*128 without aberration, much clearer than the ones with aberration.

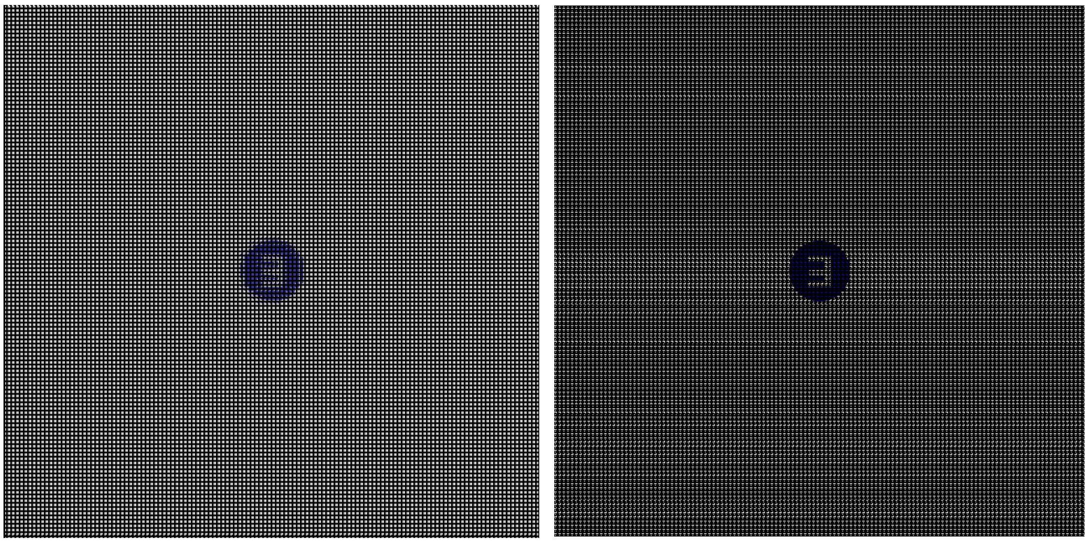


figure 4.2

Figure 4.2 left shows the compensated image calculated by our program and right shows the simulated result when put the compensated image on. We can see that the simulated result is similar to our real result which in figure 4.1 (g).