Algorithms and Applications of the Vision Correcting Display

by Charles Ding

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Brian A. Barsky
Research Advisor

(Date)

* * * * * * *

Professor Laura Waller
Second Reader

(Date)

Algorithms and Applications of the Vision Correcting Display

University of California, Berkeley

Charles Ding

12 May 2017

Abstract

Human vision problems such as nearsightedness and farsightedness are a result of optical aberrations in the human eye. The common method of treating eye aberrations is wearing corrective lenses, such as glasses or contact lenses, or undergoing laser eye surgery. The vision correcting display team under Professor Brian Barsky aims to develop a light field display using hardware and software that lets the user see a digital screen clearly without glasses or contact lenses. In addition, the team writes software simulations of the vision correcting display to evaluate the performance of the display for different eye aberrations and different digital devices.

In this paper, we present a new software approach to the display, the backward method, and compare it with the original light field algorithm [8] and the forward method [22]. We also explore two new applications of the vision correcting display: the effect of different pixel arrangements from different devices on the performance of the display and the effects of diffraction in choosing an optimal pinhole size. Finally, we present a new approach to the software simulation that combines both ray and wave optics.

Acknowledgements

I want to thank my advisor Professor Brian Barsky for all his help and support during my two years as a member of his research group. Also, I want to thank all the members of the vision correcting display team, especially Michael Chen, Dingyi Zheng, Linghan Zheng, Tongyu Chen, Luxin Yang, Vivek Claver, Haonan Jing, Hung Vu, and Yirong Zhen for all their help and contributions. Finally, I want to thank Professor Laura Waller for helping me look over my thesis and teaching the computational imaging class that I took.

Contents

Chapter 1

Introduction

Humans depend on their vision for every waking hour of the day. Out of the five main senses, vision is arguably the most important in acquring information about the surrounding environment. A person cannot walk in the streets, drive a car, or read if he or she cannot see. However, genetic inheritance, disease, and age will naturally lead to the loss of vision. According to the Vision Council, roughly 75 percent of all adults in the United States require some form of vision correction [5].

The most common method for correcting eye aberrations are with glasses and contact lenses. Both seek to divert the light rays right before they reach the cornea, or front layer of the eye, so that the image displayed on the retina, or sensor of the eye, is clear. However, glasses and contact lenses may present an inconvenience. For example, a person who is far-sighted might only need his or her glasses when reading a screen close to him or her and would prefer to not always put on and take off his or her glasses. But what if instead of fixing the eye, the image is modified on the screen so that it appears clear without eyewear? Professor Brian A. Barsky has been focusing on this field for over a decade and introduces the concept of the vision correcting display in his paper *An Overview of Vision Realistic Rendering and Vision Correcting Displays* [3]:

> The concept of a vision correcting display involves digitally modifying the content of a display using measurements of the optical aberrations of the viewer's eye so that the display can be seen in sharp focus by the user without requiring the use of eyeglasses or contact lenses. Given the measurements of the optical aberrations of a users eye, a vision correcting display will present a transformed image that when viewed by this individual will appear in sharp focus. Vision correction could be provided in some cases where spectacles are ineffective.

Fu-Chung Huang et. al proposed two different solutions to this problem. In his first solution, he uses a multilayer display [9], and in the second solution, he used light field technology, specifically a pinhole mask, to generate a sharp image out of the display plane [7]. Wu presented one new software algorithm (forward method), one major hardware improvement (lens array display), and a simulation program [22].

This paper builds on the work from Wu's report. In chapter 4, we present the backward method, which aims to fix some of the issues with the issues with Huang's algorithm. In

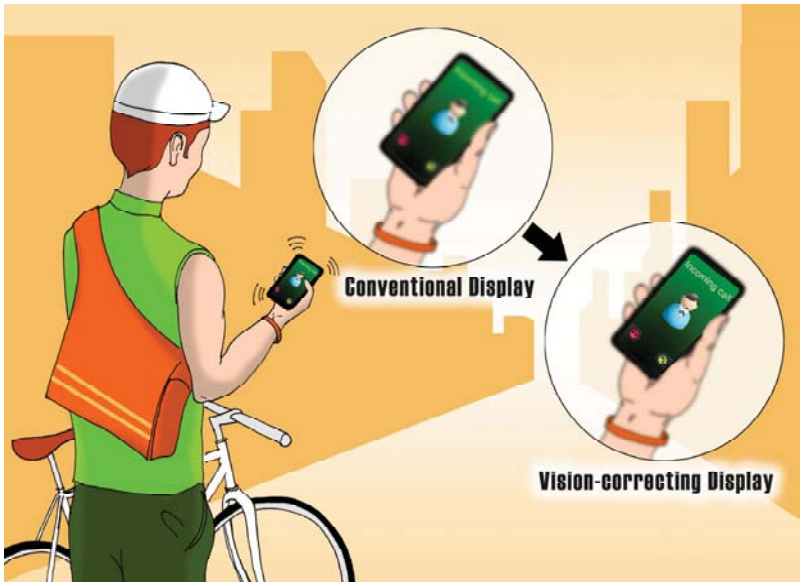


Figure 1.1: Application Scenario of Vision-Correcting Display [7]

chapter 7, we test the forward method on different RGB pixel arrangements for different phone screens. In chapter 8, we apply a combined ray and wave optics model in the simulation to account for diffraction and find the optimal size of a pinhole. Finally, in chapter 9, we compare the performance of Huang's algorithm, the forward method, and the backward method.

Chapter 2

Aberrations of the Eye

Most of the report will be devoted to correcting the first two types of aberrations, myopia, or nearsightedness, and hyperopia, or farsightedness. These two types of aberrations are examples of defocus, where the user focuses at a distance different from the that of the object, and are easiest to illustrate via a Gaussian ray tracing model. They are by far the most common aberrations and are the easiest to correct using a vision correcting display. We then go on to describe other types of aberrations such as astigmatism, coma, and trefoil, which are modeled using Zernike polynomials. Although these aberrations are much harder to correct, they represent a major subject of interest for future work.

2.1 Gaussian Ray Tracing

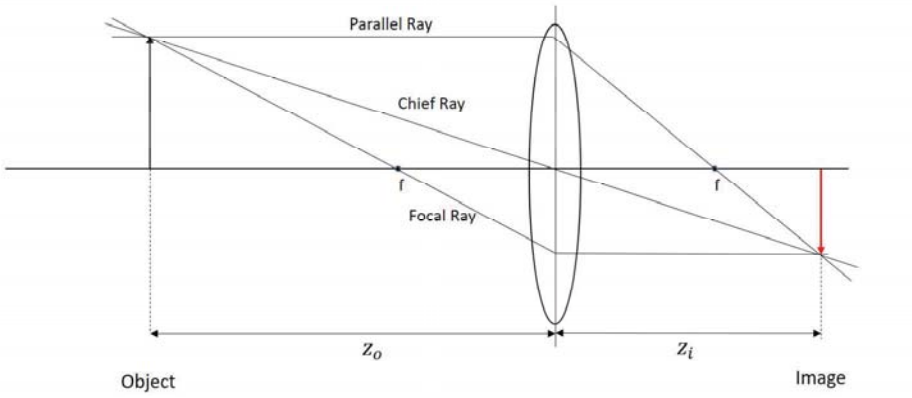


Figure 2.1: Gaussian Ray Tracing

The Gaussian ray tracing model illustrates the path of light rays passing through an ideal thin lens. The lens contains a focal point f where all of the light rays are focused, and the eye focuses at a focus plane. Three light rays are drawn: The parallel ray travels parallel to the viewing axis and bends to the focal point behind the lens. The chief ray travels straight through the center of the lens without bending. The focal ray travels to the first focal point

in front of the lens and bends parallel to the viewing axis after hitting the lens. An image is formed where the three rays intersect. In a perfect or ideal eye, the object is located at the focus plane, and the light rays traveling through the pupil converge on the sensor (retina on the eye). The distance from the object to the lens (z_o) and the distance from the image to the lens (z_i) are related by the following equation, known as the thin lens equation:

$$\frac{1}{f} = \frac{1}{z_i} + \frac{1}{z_o}$$

2.2 Myopia

In a myopic or nearsighted eye, the focus plane is in front of the object, and light rays converge in front of the retina. A myopic viewer does not have trouble seeing objects up close but cannot see objects far away.

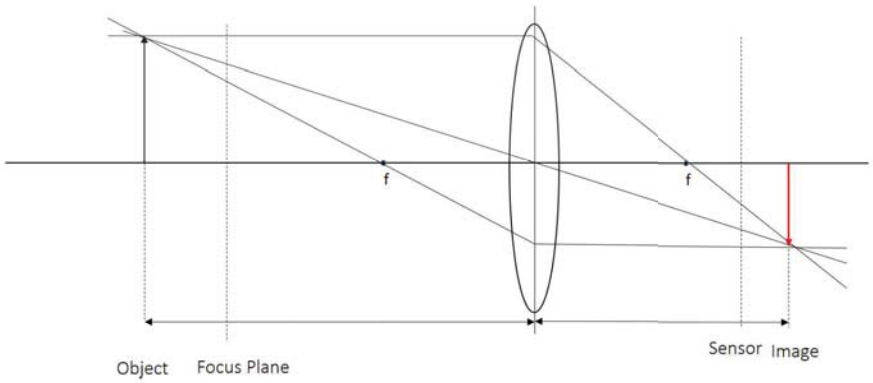


Figure 2.2: Myopia

Myopia affects about 30 to 40 percent among adults in Europe and the United States, and up to 80 percent or higher in the Asian population, especially in China [12]. In addition, incidence of myopia is increasing, from about 25.0% percent of Americans in 1971-1972 to 41.6% in 1999-2004 [20].

2.3 Hyperopia

In a hyperopic or farsighted eye, light rays converge behind the retina. The viewer has a easier time from seeing far away, but an image that is close looks blurred.

Hyperopia is a common condition that affects about five to ten percent of the United States population [16]. It can affect both children and adults, and people whose parents have hyperopia may also be more likely to get the condition.

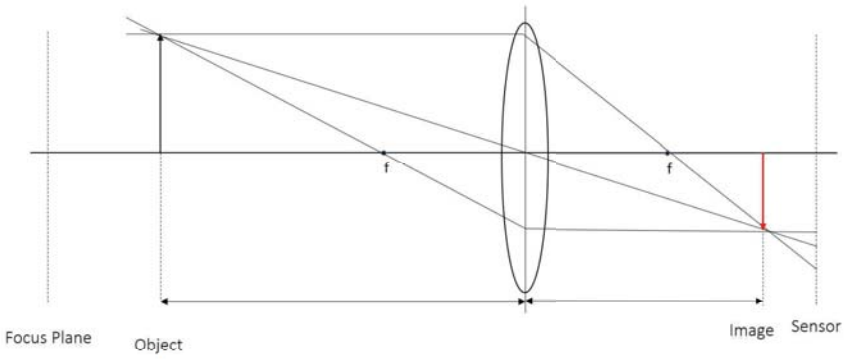


Figure 2.3: Hyperopia

2.4 Zernike Polynomials and Higher Order Aberrations

The pupil in the eye captures a wavefront, which is a surface in which the waves have a constant phase. The light rays are all perpendicular to the wavefront, and an aberrated wavefront is one in which not all of the rays traveling out of the wavefront surface travel through the same focal point. We fit the wave aberrations with the Zernike polynomials, which are a set of shapes orthogonal to the unit disk used to measure the eye's wavefront [21]. The Zernike shapes are very similar to the usual aberrations that are found in a human eye. If we consider $W(x, y)$ as the wavefront, then here is the equation that fits the Zernike polynomial:

$$W(x, y) = \sum_i c_i Z_i(x, y)$$

i represents the index of the coefficient c, and $Z(x, y)$ is the associated polynomial equation. Here are the first six terms of the Zernike polynomial:

$$\begin{aligned} W(x, y) = {} & c_0 \times 1 + c_1 \times 2 \times \rho \times sin(\theta) + c_2 \times 2 \times \rho \times cos(\theta) \\ & + c_3 \times \sqrt{6}\rho^2 \times sin(2\theta) + c_4 \times \sqrt{3} \times (2 \times \rho^2 - 1) \\ & + c_5 \times \sqrt{6} \times \rho^2 \times cos(2\theta) + ... \end{aligned}$$

The first six Zernike polynomial terms measure lower order aberrations and the next sixty terms measure higher order aberrations. The names of lower order aberrations are tilt (terms one and two), astigmatism (terms three and five), and defocus (term four). Some examples of higher order aberrations are coma and trefoil (terms six through nine) and quatrefoil, secondary astigmatism, and spherical aberration (terms ten through fifteen). Lower order aberrations account for 90 percent of all decline in the quality of retinal images [4], while higher order aberrations account for the remaining 10 percent. Lower order aberrations are frequently treated with eyeglasses, which divert the direction of light rays so that they land squarely on the retina. However, higher order aberrations are much harder to treat with eyeglasses, but with the vision correction display, a user can input the Zernike polynomial coefficients to manipulate the image on the screen.

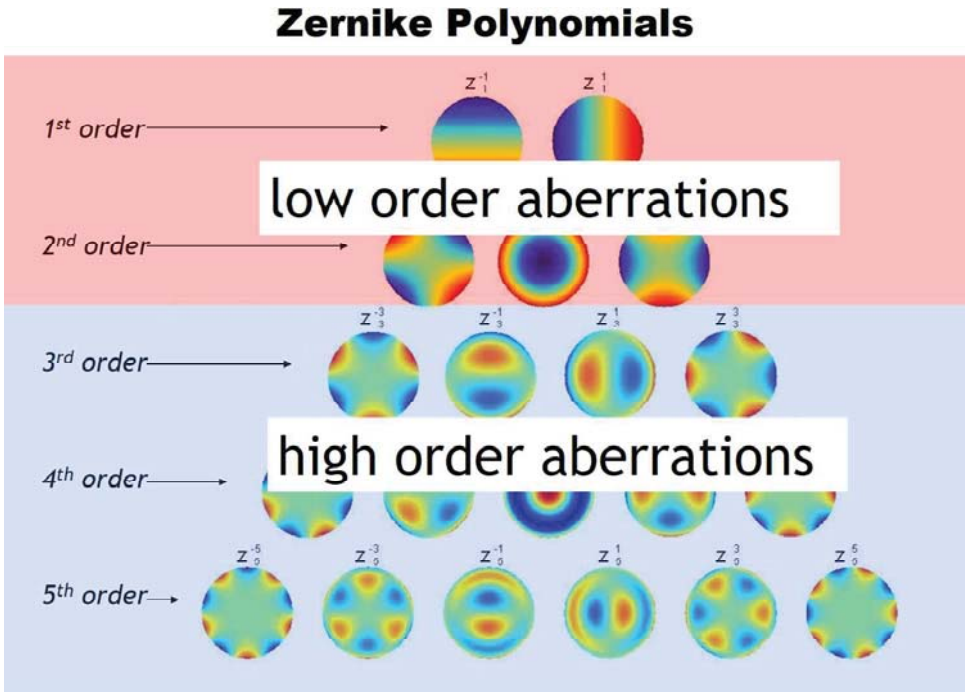


Figure 2.4: Zernike Polynomials [18]

Chapter 3

Hardware of the Vision Correction Display

We explore two hardware devices used for the vision-correction display: the pinhole mask and the lenslet array. Both devices are examples of light field displays.

3.1 Light Field Display

A light field is a vector function that describes the amount of light flowing in every direction through every point in space. The direction of each ray is given by the 5D plenoptic function

$$L(x, y, z, \theta, \phi)$$

, where (x, y, z) represents the position of the ray and (θ, ϕ) represents the direction. The magnitude of the light ray is equivalent to its radiance, denoted by L and measured in watts (W) per steradian (sr) per meter squared (m^2). The steradian is a measure of solid angle, or the subtended surface area on the unit sphere, and meters squared are used here to measure cross-sectional area.

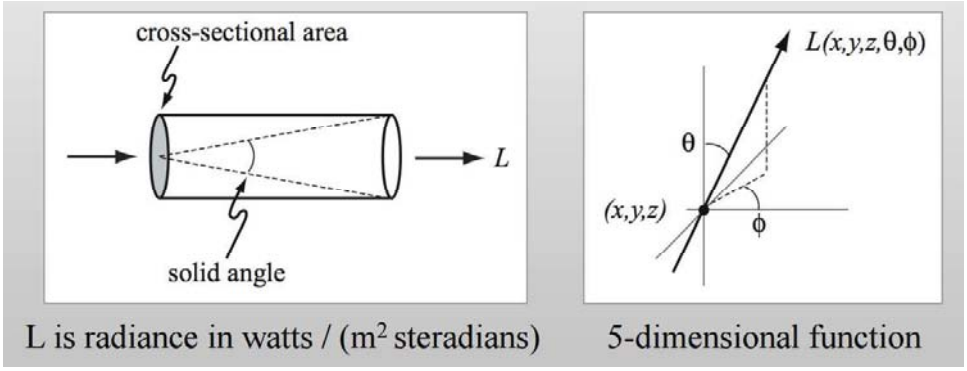


Figure 3.1: Radiance (left) and 5D plenoptic function (right) [13]

In the case of the vision correcting display, a 4D light field is considered. A 4D light field has constant radiance along the ray, and one dimension is ignored (z). Each light ray

starts at a (x, y) coordinate on the screen and in the direction (θ, ϕ). The light field display is placed on top of the device and used to capture and/or manipulate the light fields from the surface.

3.2 Pinhole Mask Display

The pinhole mask display is a light field display that is composed of a sheet of black film taped to a layer of acrylic 6 millimeters thick. The black film is composed of a 2D array of pinholes in the shape shown in the figure below.

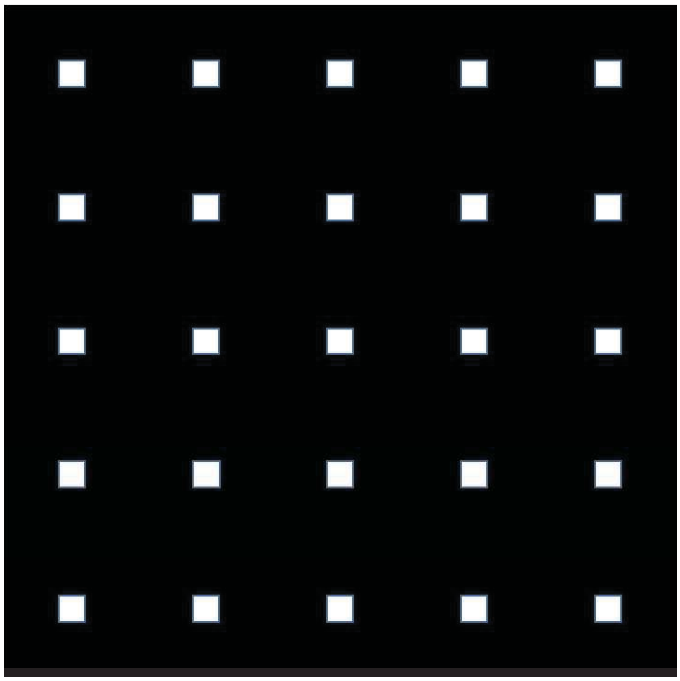


Figure 3.2: Area blocked and covered by pinhole.

The pinhole mask is an example of a parallax barrier. The parallax barrier attenuates most of the light rays in the area covered by the pinhole. Therefore, less light rays will interfere, and light from each pixel will land on a smaller area on the retina. The blur as a result of defocus is reduced and contrast is improved.

Previously, Huang et al. 2014 [7] developed a similar pinhole mask display. Both Huang's display and our display had pinholes of size 75 micrometers that are 390 micrometers apart. Huang's display is used for the Apple iPod touch 4th generation display, which has a pixel pitch of 78 micrometers and resolution of 960×640 pixels, while our display is used for the iPhone 6, which has a resolution of 1334×750 pixels. Our pinhole mask has a greater depth (6 mm instead of 4 mm), and this increased depth allowed the outer pixels in the $5x5$ superpixel covered by the pinhole to become more visible. The light rays from the outer pixels travel at an angle through the pinhole mask and will miss the pupil or aperture if the angle was too large.

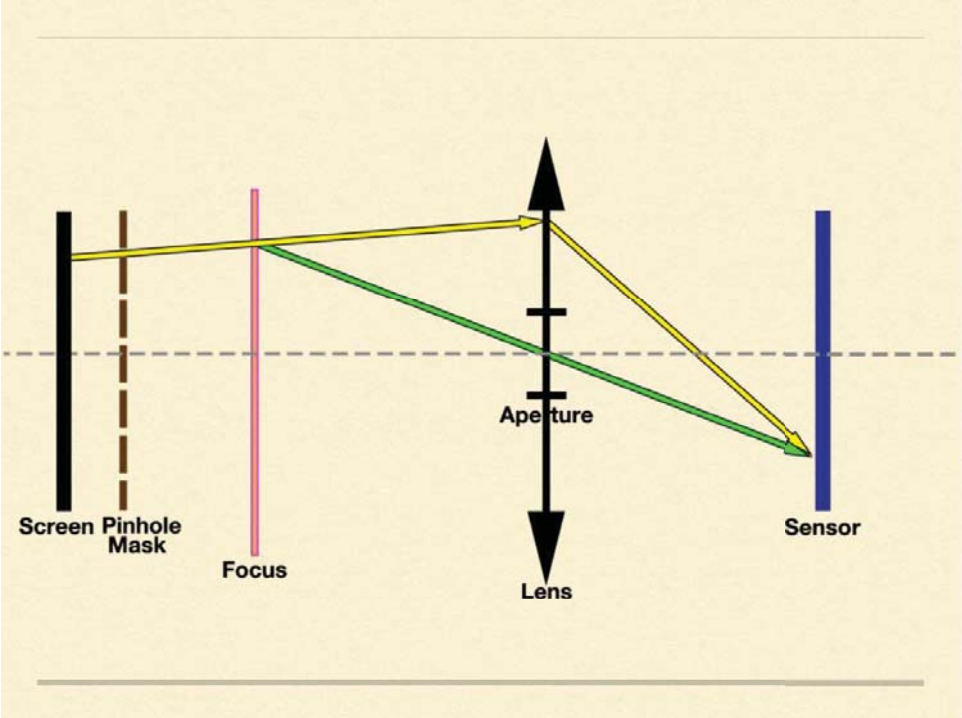


Figure 3.3: Light rays traveling through a pinhole [22]

3.3 Lens Array Display

The lens array display is a 2D array of microlenses etched on a layer semiconductor material. A microlens is a small lens, generally with a diameter less than a millimeter (mm) and often as small as 10 micrometers (μm). The depth of the microlens is equal to the its focal length so that all of the light rays emanating from a screen pixel travel outwards in parallel. The size of the microlens is equal the length of its covered superpixel. For example, in a 5x5 superpixel on the iPhone6 (pixel size $0.078mm$), the diameter of the microlens would be $5 \times 0.078mm = 0.3895mm$.

One advantage of the lens array is that no light is lost and no information is lost. In the pinhole mask, light from many screen pixels are blocked, and only about 4 percent of all light under a 5x5 superpixel pass through the pinhole. However, in a microlens array, light from every screen pixel is allowed to pass through. One disadvantage of the microlens array is that it costs significantly more to manufacture than the pinhole mask display.

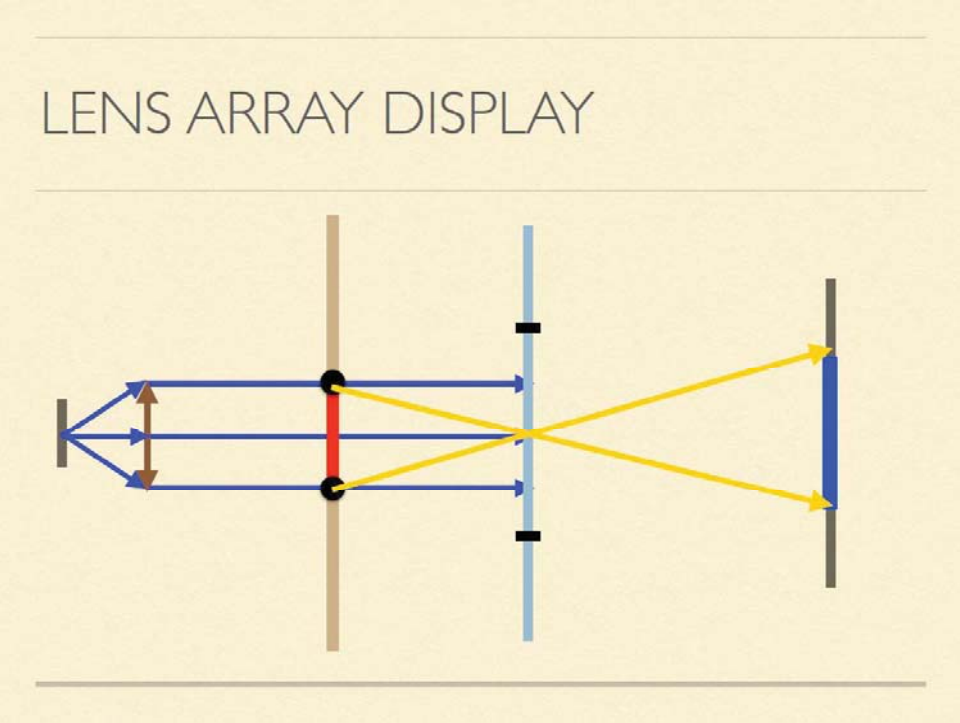


Figure 3.4: Light rays traveling through a microlens [22]

Chapter 4

Software of the Vision Correction Display

4.1 Introduction to Algorithms

Fu-Chun Huang et al. applied light field technology to develop an algorithm that created a sharp image on the display plane [8]. Peter Wu introduced the forward method and its optimizations [22]. Another new algorithm we will introduce is the backward method, which processes an image under the assumption that most light rays that travel through the center of the pinhole or lenslet. The method aims to improve run time compared to Huang's algorithm while taking into consideration more light rays than the forward method. We will use the same definitions, symbols, and content on Huang's algorithm and the optimized forward method in chapter 4 of Wu's paper [22] for consistency.

4.1.1 Projection and Prefiltering

Each algorithm in this paper is divided up into two components: the projection relationship and prefiltering. The projection relationship is the mapping relationship from a point on the display [1] to a point on the sensor or retina. The projection relationship depends on the experimental settings such as focal point, object focus, and object distance. Prefiltering is the step in which the content on the display is modified. Prefiltering takes in the projection relationship and the input image and produces a transformed output image.

4.1.2 Symbols

The size of the input image is s_w by s_h pixels. The input image is the unmodified image a normal person sees on the display. We define two functions f_s and f_d, both of which contain the same parameters:

- p and t stand for arbitrary points in 3-D space. p and t specify a light ray.

[1] To avoid confusion, display in this chapter means the phone screen and not the light field display. In later chapters, we will use the word screen instead of display.

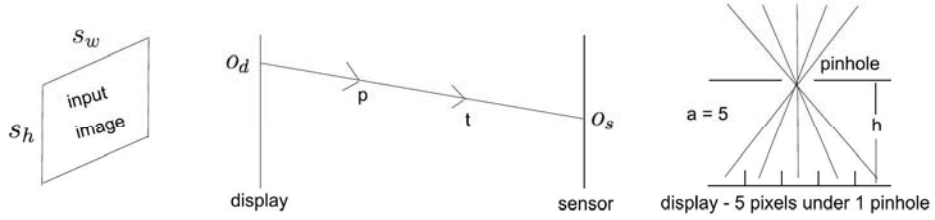


Figure 4.1: The figure shows the setup for the prefiltering algorithm. (left) The input image has dimensions s_w by s_h pixels. (middle) p and t are points on the light ray. (right) There are 5 pixels under each pinhole, and h is the distance between the pinhole and the display. This image is from [22].

- e is the experimental settings. If we only consider defocus (myopia or hyperopia), the only parameter to consider is the distance from the the screen to the lens.
- c is the condition of the camera or human eye. If we only consider defocus, then this parameter is the distance from the sensor to the lens.

f_s takes in the parameters and computes the position o_s, the position where the light ray (p, t) hits the sensor. Similarly, f_d takes in the same parameters and computes the position o_d, the position where the light ray (p, t) hits the display.

$$f_s(p, t, e, c) \rightarrow o_s$$

$$f_d(p, t, e, c) \rightarrow o_d$$

The functions m_s and m_d transforms an arbitrary position into a discrete two-dimensional index. Both take a parameter p which is any point on either the display plane or sensor place and transform p into a integer two-dimensional index.

$$m_s(p) \leftarrow (a_s, b_s)$$

$$m_d(p) \rightarrow (a_d, b_d)$$

The angular resolution a is the ratio of the screen resolution to that of the pinhole mask or lens array. If a 5x5 pixels are covered under the light field display, then $a = 5$ for the device. The depth h is the distance from the surface of the light field display to the plane of the phone display.

4.2 Huang's Light Field Algorithm

The projection relation is represented as a matrix in Huang's algorithm. He assumes that the sensor has the same resolution as that of the pinhole mask, which is $L_s = s_w/a$ by $L_d = s_h/a$. These display pixels have resolution $s_w \times s_h$. Next, the algorithm builds a projection matrix. First, an all-zero matrix P with dimensions L_d by L_s is created. Second, for every pixel on the screen, n points $\{o_n\}$ are sampled on the aperture, and a light ray

$p_s - o_i$ is constructed for each sample point. Third, the algorithm applies the function f_d to get the position p_d on the display. The function f_d in this case uses Zernike polynomials and can consider higher order aberrations. Then, the function m_d is called to convert p_d into a 2-D index I_d. Finally, the algorithm adds index I_d of the matrix by $1/n$.

After the projection matrix is formed, the following linear equation needs to be solved, where P is the project matrix and b is the one-dimensional representation of the input image:

$$P \cdot x = b$$

P is not a square matrix, but we can multiply both sides by P^T, and $P^T P$ is square.

$$(P^T \cdot P) \cdot x = p^T \cdot b$$

Define $H = P^T \cdot P$. There is a high chance that H is singular and there is no solution to the linear system. To solve the linear system, a very small positive value λ is defined and added to the the system where I is the identity matrix:

$$H' \leftarrow H + \lambda \cdot I$$

λ introduces a negligible amount of error to the system, and H is not singular anymore.

Instead of solving the system directly, Huang's algorithm changes it into an optimization problem:

$$\textbf{Minimize:} (H'x - P^T b)^T \cdot (H'x - P^T b)$$

The L-BGFS method [14] was used to solve this optimization problem. The final result x may have very large positive or very small negative values, but Huang cut off all values in the range (0, 255). The algorithm was applied for the R, G, and B (red, green, and blue) channels separately. The projection matrix was the same for computing the display matrix for each channel.

4.2.1 Analysis

Huang's algorithm is the most thorough algorithm in that it considers light rays for all possible positions on the aperture and therefore almost all possible positions between the display and sensor. However, it is subject to many flaws. The algorithm can lead to inconsistency if the number of random aperture samples is not large enough. It also has a slow runtime: the algorithm takes 45 seconds in MATLAB assuming the projection matrix for a viewing angle (angle between normal of phone display and sensor plane) of 0 was precomputed and roughly three hours if all viewing angles from -5 degrees to 5 degrees are considered. The time complexity of the algorithm is $\mathcal{O}(m \cdot n^2)$, where m is the number of samples on the aperture and n is the length of a square sensor. The algorithm leads to loss of light because it to divides by n while most of the n light rays do not reach the display. The assumption that the sensor resolution is $1/a$ that of the original display is misleading because a human eye often has resolution even better than that of the display. The algorithm ignores the pixel arrangements and the fact that each point on the display can only correspond to one color.

Finally, the L-BGFS method makes the prefiltering speed greatly dependent on experimental settings and the user's eye condition and adds significant complexity to the algorithm.

4.3 Forward Method

In the forward method, we assume that the sensor has the same resolution as the display or screen, so if the display or phone screen has size 640×640 pixels, then so does the sensor. In addition, the light rays travel from the display to the sensor instead of from the sensor to the display.

In the projection step of the algorithm, we compute all possible relationships from each point on the display to any point on the sensor. Following the function f_s, the light ray travels in a continuous straight line from the display pixel p_d to the center of the closest pinhole o_c, and if not blocked by the aperture, to a point on the sensor o_s. We keep track of the sensor position for each display position, and apply this process for the R, G, and B channels separately. Since the three channels occupy different positions on the display pixel[2], the list of sensor positions for each channel is different. Unlike Huang's algorithm, there are no matrices involved in the optimized forward method.

In prefiltering, the first step in is to the set the sensor image equal to the original perfect image. Then, for each display position o_d, we set its R, G, and B equal to pixel values at the precomputed sensor positions $o_s(R)$, $o_s(G)$, $o_s(B)$, respectively. The display color is set to 0 if the sensor position is invalid.

The two steps can be merged together in that for each display pixel, the corresponding R, G, and B sensor pixel positions are computed and the display pixel colors is set to the sensor pixel colors.

Listing 4.1: Pseudocode For Forward Method [3]

```
int[][] sensor_image = origin_image;
int[][] prefiltered_image = new int[sensor_image.height][sensor_image.width];
// Assume sensor_image is a square
int screen_size = sensor_image.width;
int aperture_radius = 0.003; // 3 mm aperture radius

// Loop through the display image
for (int y_index = 0; y_index < screen_size; y_index++) {
  for (int x_index = 0; x_index < screen_size; x_index++) {
    float2 screen_pos = compute_screen_pos(x_index, y_index);
    float2 pinhole_pos = find_nearest_pinhole(screen_pos);
    float2 aperture_pos = compute_aperture_pos(screen_pos, pinhole_pos);
    if (aperture_pos.norm < aperture_radius) {
      int2 sensor_pos = ray_trace(screen_pos, pinhole_pos, sensor_image);
      prefiltered_image[screen_pos] = sensor_image[sensor_pos];
    }
  }
}
```

[2]More on this will be covered in chapter 7. A pixel is composed of a red, green, and blue area.

[3]The R, G, and B channels are ignored for simplicity.

4.3.1 Analysis

The mains benefits of the forward method are that it is fast and simple. The method has a run time of less that one second for large images and a time complexity of $\mathcal{O}(n^2)$, a major improvement over Huang's algorithm. Huang's algorithm samples many rays that are blocked by the pinhole mask, while the forward method ignores such samples. One issue with the forward method is that only light rays that travel through the center of the pinhole mask are considered, which could lead to loss of information. In addition, by starting from the display, the amount of light a sensor pixel receives is hard to measure.

4.4 Backward Method

In the backward method, rays are traced from the sensor to the screen, like in Huang's algorithm. We also assume that the resolution of the display and the sensor are the same.

In the projection step of the algorithm, we compute all possible points on the display for each point on the sensor. The function f_d applies Gaussian ray tracing, where the sensor pixel p_s corresponds to a point on the focus plane o_f, and we check whether a ray traveling from o_f in the direction of o_c is blocked by the aperture or pupil. If not blocked, the intersection of the ray and display plane o_d is added to the list of display points for the sensor pixel. f_d is applied for every pinhole. The RGB channels are not taken into consideration to make sure that the number of red, green, and blue hits are equal. Like the forward method, there are no matrices involved, although it is possible for generate the projection in matrix form.

In prefiltering, the first step is to the set the sensor image equal to the original perfect image, the display image equal to a $s_w \times s_h$ array of zeros, and a counter matrix of size s_w by s_h for normalization. For each sensor pixel p_s, we check the set of display pixels from projection $\{p_d\}^n$ and increment the color of the display pixel p_d is by the the color on the sensor pixel p_s and the temporary matrix at p_d by 1. We then do a matrix pointwise division between the display image and the counter matrix.

The two steps can be merged together in that for each sensor pixel, the corresponding display positions for each pinhole are computed and the display pixel colors are set to the sensor pixel colors.

Listing 4.2: Pseudocode For Backward Method [4]

```
int[][] sensor_image = origin_image;
int[][] prefiltered_image = new int[sensor_image.height][sensor_image.width];
int[][] prefiltered_image_norm = new int[sensor_image.height][sensor_image.width];
// Assume sensor_image is a square
int sensor_size = sensor_image.width;
int angular_resolution = 5;
int pinhole_size = sensor_size // angular_resolution
int aperture_radius = 0.003; // 3 mm aperture

// Loop through the sensor image
for (int y_index = 0; y_index < sensor_size; y_index++) {
  for (int x_index = 0; x_index < sensor_size; x_index++) {
  int2 screen_pos = (x_index, y_index)
    // Check each pinhole
```

```
    for (int py = 0; py < pinhole_size; py++) {
      for (int px = 0; px < pinhole_size; px++) {
        float2 pinhole_pos = compute_pinhole_pos(px, py);
        float2 aperture_pos = compute_aperture_pos(pinhole_pos, screen_pos);
        if (aperture_pos.norm < aperture_radius) {
          float2 display_pos = compute_display_pos(pinhole_pos, aperture_pos);
          prefilter_image[display_pos] += sensor_image[screen_pos];
          prefilter_image_norm[display_pos] += 1;
        }
      }
    }
  }
}

for (int y_index = 0; y_index < sensor_size; y_index++) {
  for (int x_index = 0; x_index < sensor_size; x_index++) {
    int2 screen_pos = (x_index, y_index);
    prefilter_image[screen_pos] /= prefilter_image_norm[screen_pos];
  }
}
```

4.4.1 Analysis

The backward method seeks to fix some issues of both Huang's algorithm and the forward method, but also has issues similar to both. It is simpler than Huang's algorithm but more complex than the forward method. The method takes around 2.5 minutes to run, which is better that Huang's algorithm at multiple angles and slower than both the one angle version of Huang's algorithm and the forward method. The time complexity is $\mathcal{O}(n^4)$, which is worse that Huang's algorithm for large screen sizes. There is a lot of unnecessary sampling as light from many pinholes will not reach a specific point on the sensor. Like the forward method, only light rays that travel through the center of the pinhole mask are considered, which could lead to loss of information. One way to reduce this loss of information is to take many samples points on the sensor pixel, but this makes the run time even worse. The main benefits of the backward method are that it is most similar to the software simulation algorithm and that light passing through every pinhole is considered for each sensor pixel.

[4]The R, G, and B channels are ignored for simplicity.

Chapter 5

Physical Experiment

The simulation software in Wu's paper demonstrates that Huang's algorithm and the forward method do a lot to correct for defocus, but does the physical setup yield similar results [22]? One can think of the simulation software as tests or a rough draft of a final paper, and the physical demonstration as the final product or copy. The camera in the physical demonstration mimics a human eye with defocus in the real world, and the images captured by the camera are similar to the images perceived by the retina. We set up the camera as a defocused eye and compare the performance of Huang's algorithm and the forward method for effectiveness in vision correction.

5.1 Physical Setup

The camera used in the experiment was a Nikon D70s digital camera with a Nikon 18-70mm f/3.5-4.5 AF-S DX lens. We set the f-stop of the camera to f/8 (closest to human pupil size of 6 mm) and set the zoom to 70 mm. The distance between the sensor and the flange focal distance, or distance from the flange (the rear end of the lens) to the film plane, is 46.5 mm. The camera was focused at 380 millimeters and the distance between the camera and the phone was 250 millimeters. Since the camera focuses farther than the object, our setup simulates mild hyperopia. We chose such distances to mimic a hyperopic person trying to read from a screen up close. This setup closely resembles a +7.69 D defocus, and a +4.5 D defocus represents a risk factor for adults [11]. For Huang's algorithm, the depth of the pinhole mask was 4 millimeters, and for the forward method, the depth was 6 millimeters. Other parameters remained the same for both algorithms.

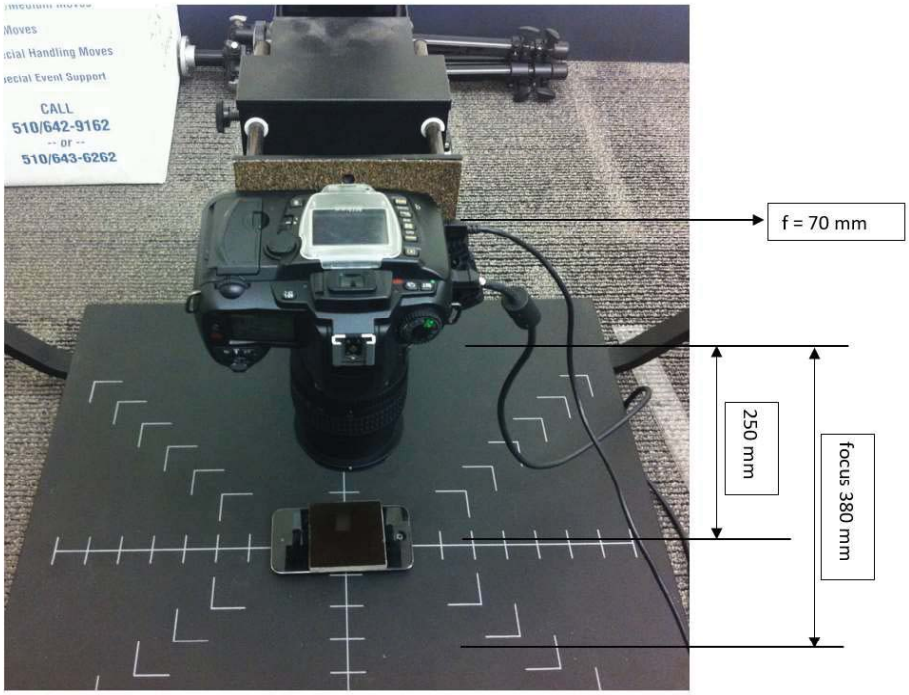


Figure 5.1: Experimental Setup

5.2 Results

Table 5.1: Original Images for Reference

Bunny Image	Word Image
	Hello Hello World Hello World Hello World Wish you happy everyday How are you today?

Table 5.2: Huang's Algorithm, Bunny Image

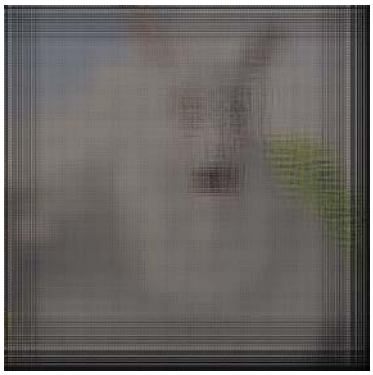
Blurred/Defocused Image	Prefiltered Image	Pinhole-Corrected Image
		

Table 5.3: Forward Method, Bunny Image

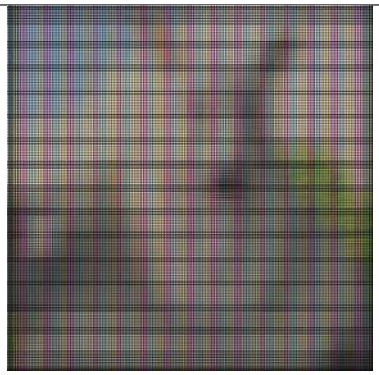
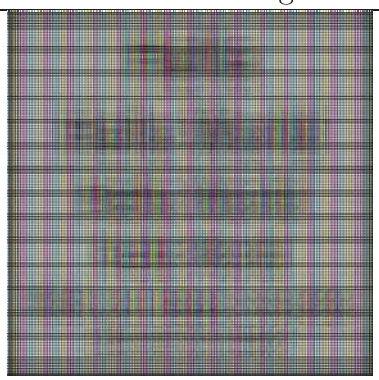
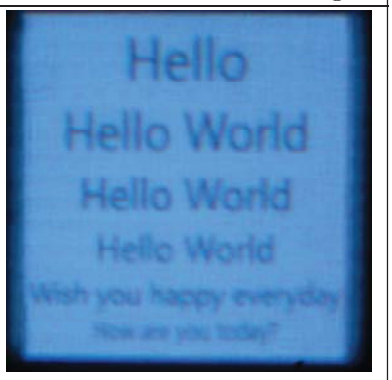
Blurred/Defocused Image	Prefiltered Image	Pinhole-Corrected Image
		

Table 5.4: Forward Method, Text Image

Blurred/Defocused Image	Prefiltered Image	Pinhole-Corrected Image
		

5.3 Analysis

For Huang's algorithm, the corrected image shows better resolution but worse contrast. The bunny is far more clear in the corrected image than the defocused image, but the sky becomes

saturated. Nevertheless, an average user would likely prefer the corrected image.

For the forward method, the bunny image shows moderately better resolution with little to no change in contrast. The forward method yields slightly less improvement in resolution than Huang's algorithm, but the limited loss of contrast makes the forward method more appealing depending on the viewer.

The forward method performs best on the text image in terms of resolution and contrast. In the defocused image, the last three lines are nearly impossible to read, and all words are badly blurred. However, in the corrected image, the first 3 lines are very clear, the next two lines are readable, and only the final line is blurry.

We adjusted the image brightness on the cell phone and the exposure time of the camera that yielded the best possible results, so we did not keep the brightness factor constant. Nevertheless, exposure time was kept between 1.5 seconds and 2 seconds, so the differences in brightness between images was not significant. In the real world, a user has the ability to adjust the brightness on their cell phone, and a human eye is capable of adjusting to different light conditions.

5.4 Conclusion and Future Work

For this specific case – an object placed 250 millimeters away and focused at 380 mm – both Huang's algorithm and the forward method yield improvements in quality of image. In the bunny image, Huang's algorithm yields slightly better resolution than the forward method but at the cost of losing contrast. The improvement in text image in the forward method is especially appealing because users usually spend more time reading text on an iPhone than viewing images. In the future, we want to run experiments on other object distances, focus distances, and depths. For example, we may want to try to simulate myopia (nearsightedness) – focus at 380 mm but place the object 500 mm away. Moreover, we want to do physical experiments with higher-order aberrations, which is much more challenging.

Chapter 6

Software Simulation

We develop a software simulation to model the effectiveness of the two light field displays. There are multiple motivations for the software simulation. One, a software simulation allows experiment parameters, such as focus distance and physical distance to be adjusted more easily. In addition, a pinhole mask or lens array display is costly, and the software simulation is used to fine tune the parameters such as depth and lenslet or pinhole size. Finally, higher order aberrations are difficult to simulate with ordinary lenses, but Zernike polynomials are easy to incorporate into a software simulation.

6.1 Ray Optics Model

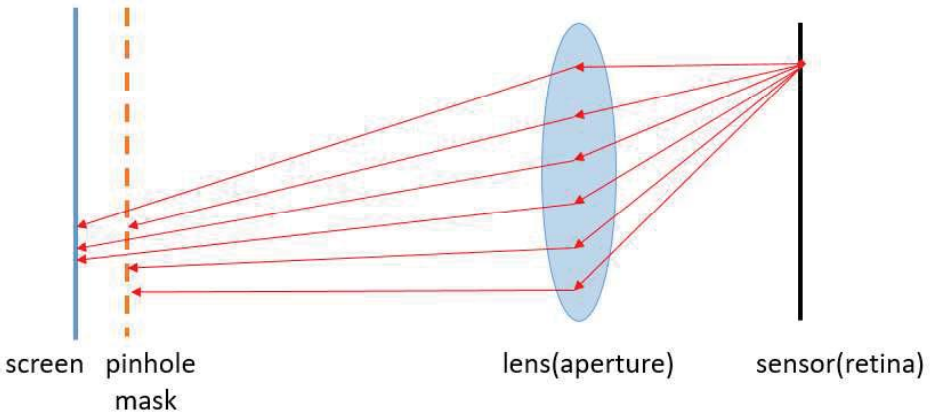


Figure 6.1: Ray Optics Simulation

One approach we take to simulate the physical experiment is a purely ray optics model. We use a backward ray tracing algorithm, in which light travels from the sensor plane to the display plane. For every pixel on the sensor, the simulation samples multiple points on the aperture (human eye pupil of diameter 6 millimeters). The ray travels straight from the sensor pixel to the aperture point and to the screen. Some of the light rays may get blocked by the pinhole mask or entirely miss the screen. Each screen pixel is composed of a blue,

green, and red area, and the color of the ray depends on which position it lands on the screen pixel. Each light ray that reaches the screen contributes to either the red, green, or blue intensities of the sensor pixel.

Listing 6.1: Pseudocode For Ray Optics Simulation

```
int[][][] sensor = new int[sensor_size][sensor_size][3];
// Iterate over sensor pixels
for (int iy = 0; iy < sensor_size; iy++) {
  for (int ix = 0; ix < sensor_size; ix++) {
    int color[3] = {0, 0, 0};
    int hits = 0;
    // Iterate through random aperture points
    for (int i = 0; i < aperture_sample_time; i++) {
      sensor_pos = (ix, iy);
      aperture_pos = (aperture_samples[2*i], aperture_samples[2*i + 1]);
      type, color = getRayColor(sensor_pos, aperture_pos);
      if (value >= 0) {
        hits++;
        color[type] += value;
      }
    }
  }
  sensor[ix][iy] = color * 3 / hits;
}
```

In the physical world, each pixel of light travels straight through free space, through the pupil, and hits a point (rod or cone cell) the retina. Each point on the retina may receive light rays from multiple pixels, so the backward ray tracing algorithm does a good job simulating a human eye.

6.2 Advantages and Disadvantages

Another advantage of this approach is that it is simple and easy to parallelize. With OpenCL, a simulation of an image of size 640×640 pixels for roughly 1100 aperture samples runs in less than one minute. This approach can also take into consideration the quantity of light traveling from the screen to the sensor. If the pinhole size is too small, then too many light rays are blocked by the pinhole mask, and the image on the sensor becomes dim due to the lack of light. To fix this issue, we divided the sum of the light rays by the number of hits.

One disadvantage of the ray optics model is that it does not take into account wave properties like diffraction and interference. Even though small pinhole sizes filter out more light and improve contrast, they also create larger diffraction effects, which this model does not capture.

Chapter 7

Case Study: RGB Pixel Arrangements

A pixel is composed of miniature red, green, and blue LEDs. When modeled as a square, there is a red, green, and blue component as well as dark areas. Different phones have different RGB pixel arrangements, and we want to test whether the prefiltering algorithm works regardless of pixel arrangement. We ran the software simulation with the forward method on different pixel arrangements. We consider the scenario with no pixel arrangements, which is the same as a RGB order on an OpenCV image, and a base case of a pixel with simple red, green, and blue bars, in that order. The different devices that we considered are the iPhone [1] and three generations of Samsung [2] [3] [4]. We chose a focus at 400 mm, distance of 350 mm, and a screen size of 640x640 pixels. In addition, for the simulation, we compared normalizing by total number of hits and normalizing by the number of hits per color.

7.1 Results

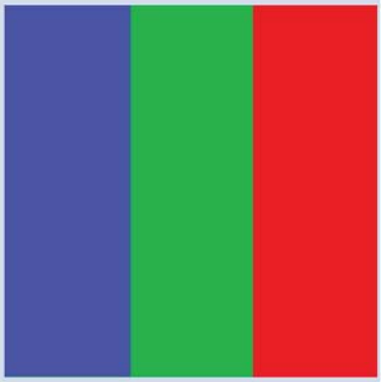
[1] Link to iPhone pixel: https://www.androidheadlines.com/2013/04/samsung-shows-off-the-new-diamond-pixel-layout-behind-the-s4s-amoled-display.html [23]

[2] Link to Samsung Galaxy S2 pixel: http://www.gsmarena.com/samsung_galaxy_note_ii-review-811p3.php [19]

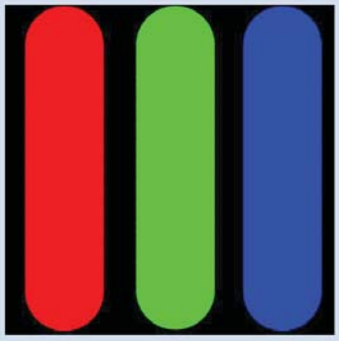
[3] Link to Samsung Galaxy S3 pixel: https://www.androidheadlines.com/2013/04/samsung-shows-off-the-new-diamond-pixel-layout-behind-the-s4s-amoled-display.html [23]

[4] Link to Samsung Galaxy S4 pixel: https://www.chipworks.com/ko/node/126 [10]

No Pixel Arrangements

Pixel Arrangement	Normalize By Total Hits	Normalize By Hits Per Color
		

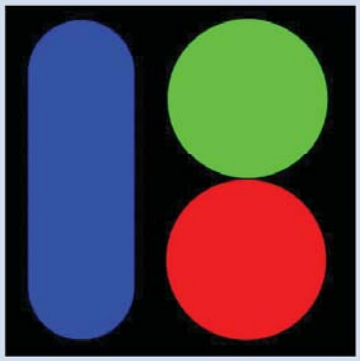
Base Case: RGB Bar

Pixel Arrangement	Normalize By Total Hits	Normalize By Hits Per Color
		

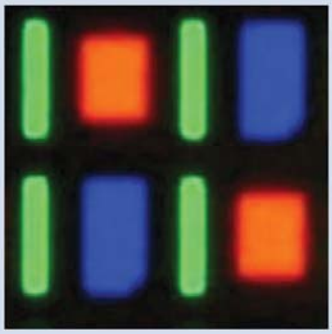
iPhone

Pixel Arrangement	Normalize By Total Hits	Normalize By Hits Per Color
		

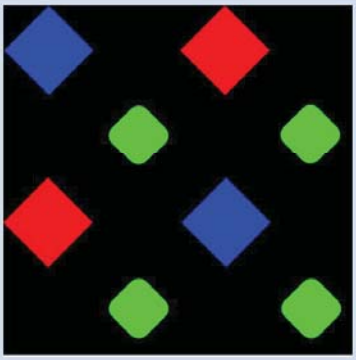
Samsung Galaxy S2

Pixel Arrangement	Normalize By Total Hits	Normalize By Hits Per Color
		

Samsung Galaxy S3

Pixel Arrangement	Normalize By Total Hits	Normalize By Hits Per Color
		

Samsung Galaxy S4

Pixel Arrangement	Normalize By Total Hits	Normalize By Hits Per Color
		

7.2 Analysis

The default simulation (OpenCV RGB bars) yields the best results, which is expected. The RGB Bar and the Samsung Galaxy S2 have images that are bright with a blue hue when normalization by the total number of hits is considered. This is expected for the Samsung Galaxy S2 since the the blue LED covers a larger area than the green or red LED, but very unusual for the RGB bar, which has equal red, green, and blue. The iPhone does not have any unique shade, which is expected. The Samsung Galaxy S3 and especially the Samsung Galaxy S4 have a green shade when normalized by the total number of hits, possibly due to a disproportionate amount of green rays as a result of a larger green area on the pixel. All images look very clear when normalized by hits per color and yield a similar grid pattern. The grid pattern is more severe for pixel arrangements with a larger black area as a percentage of total area, which makes sense because more light rays will miss the screen and deviate from the no pixel arrangement case.

As expected, the pixel arrangement creates a major impact when we normalize by the total hits because there is a disproportionate amount of red, green, and blue hits when the red, green, and blue components are not equal in area or on different positions of the pixel. However, when we normalize by hits per color, this issue is resolved. The grided pattern may be a result of some pixels on the sensor that have rays traced to unexpected areas on the screen, but it does not interfere much with clear recognition of the bunny.

Chapter 8

Case Study: Diffraction through the Pinhole

One important question to ask about the vision correcting display is: "What is the optimal pinhole size for the pinhole mask?" The regular simulation applies ray optics, which does not take into account properties of light, such as diffraction and interference. We want a new simulation that models the effects of diffraction through a pinhole.

8.1 Software Simulation - Wave Optics

We utilize another approach to the software simulation that considers both light traveling as a ray and the diffraction effects of the pinhole.

8.1.1 Fraunhofer Diffraction through Rectangular Aperture

We assume that a light source emits a spherical wavefront that creates a constant electric field on the pinhole. We consider each screen pixel as a point source, and since the depth of the pinhole mask (6 millimeters) is much larger than the size of the screen pixel (78 micrometers) or the pinhole (75 micrometers), the value of the electric field will not vary significantly between the edges of the pinhole and center of the pinhole. The distance between the pinhole plane and the sensor plane is about 27 centimeters, which is much larger than the size of the sensor, so there is not a large variation in the length of different light rays traveling from the mask to the sensor. The pinholes are squares, and we meet the conditions for Fraunhofer diffraction through a rectangular aperture [1]:

$$\frac{W^2}{L \times \lambda} \ll 1$$

Here, W is the length of the aperture, L is the distance of the sensor from the aperture, and λ is the wavelength. W is at most 125 micrometers, L is roughly 27 centimeters, and λ is about 500 nanometers.

Let ϵ_A represent the electric field strength per unit area, which is assumed to be constant over the entire aperture (derivation below from Hecht 1987 [6]). The electric field contributed

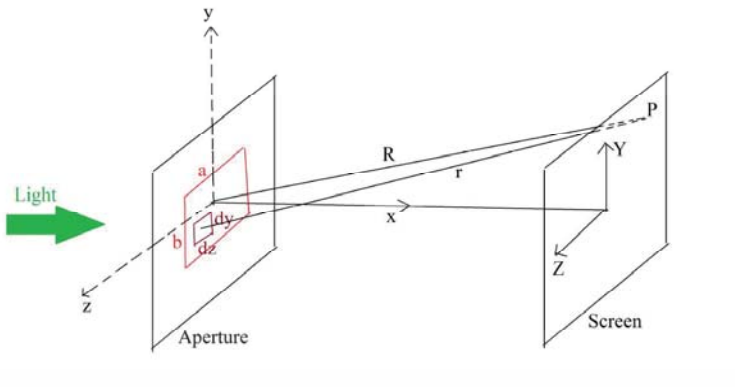


Figure 8.1: Fraunhofer Diffraction through Rectangular Aperture Diagram [17]

by the small section of the aperture $dzdy$ at point P on the screen is

$$dE = \left(\frac{\epsilon_A}{R}\right) e^{i(\omega t - kr)} dzdy$$

where $r = [X^2 + (Y-y)^2 + (Z-z)^2]^{\frac{1}{2}}$ and $k = \frac{2\pi}{\lambda}$, the wavenumber. We can use the far field approximation and set $r = R$ for the amplitude term and $r = R[1 - (Yy + Zz)^2/R^2]$ in the phase term. The total electric field at point P on the screen is

$$E = \left(\frac{\epsilon_A}{R}\right) e^{i(\omega t - kR)} \int_{-b/2}^{b/2} e^{ikYy/R} dy \int_{-a/2}^{a/2} e^{ikZz/R} dz$$

Solving the integral gives

$$E = \left(\frac{ab\epsilon_A}{R}\right) e^{i(\omega t - kR)} sinc\left(\frac{kaZ}{2R}\right) sinc\left(\frac{kbY}{2R}\right)$$

Intensity is equal to the square of the amplitude of the electric field intensity, so

$$I(Y,Z) = Re\left[\left(\frac{ab\epsilon_A}{R}\right) e^{i(\omega t - kR)} sinc\left(\frac{kaZ}{2R}\right) sinc\left(\frac{kbY}{2R}\right)\right]^2$$

$$I(Y,Z) = I_0 sinc^2\left(\frac{kaZ}{2R}\right) sinc^2\left(\frac{kbY}{2R}\right)$$

Intensity of light is proportional to two *sinc* squared functions multiplied in both the Y and Z directions.

8.1.2 Diffraction through a Lens

The above diffraction equation creates a very large diffraction pattern on the sensor that leads to far too much loss of contrast. To fix this,we need to consider the effects of the lens and the fact that the display is not placed at the focus distance of 380 millimeters.

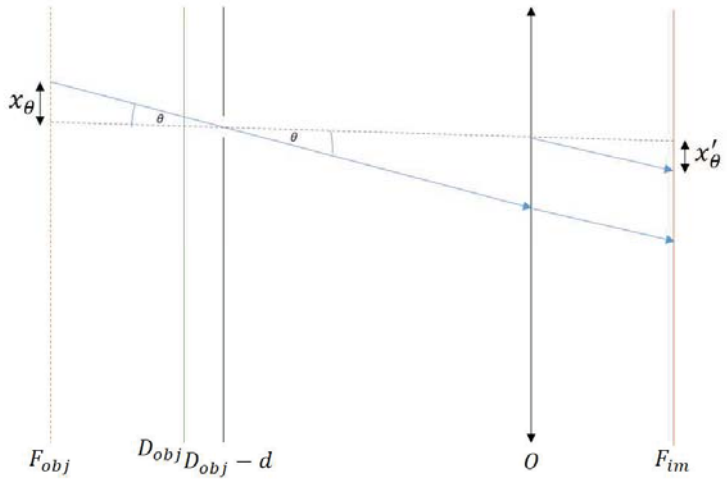


Figure 8.2: Fraunhofer diffraction through lens. F_{obj} is the focus distance, D_{obj} is the display distance, d is the depth of the pinhole mask, O is the plane of the lens, and F_{im} is equal to position of the sensor at focus.

We know that

$$I(Y,Z) = I_0 sinc^2\left(\frac{kaZ}{2R}\right) sinc^2\left(\frac{kbY}{2R}\right)$$

We substitute in $sin(\theta) \sim \frac{Z}{R}$ and $sin(\phi) \sim \frac{Y}{R}$:

$$I(\theta,\phi) = I_0 sinc^2\left(\frac{\pi a sin\theta}{\lambda}\right) sinc^2\left(\frac{\pi b sin\phi}{\lambda}\right)$$

We want to find the shape of the diffraction pattern as a function of sensor coordinates, x'_θ and x'_ϕ. In other words, we want to determine $I(x'_\theta, x'_\phi)$, where x'_θ is the image created by x_θ and x'_ϕ is the image created by x_ϕ.

We consider the angle that light travels through the pinhole, and imagine that the light ray is traced back to the plane at the focus distance. We have:

$$tan(\theta) = \frac{x_\theta}{F_{obj} - D_{obj} + d}$$

which gets rearranged to

$$x_\theta = (F_{obj} - D_{obj} + d)tan\theta$$

Then, based on the displacement x_θ on the focus object plane, we can compute the displacement on the sensor x'_θ through the magnification equation.

$$x'_\theta = x_\theta \frac{F_{im}}{F_{obj}} = \frac{F_{im}}{F_{obj}}(F_{obj} - D_{obj} + d)tan\theta$$

Likewise,

$$x'_\phi = x_\phi \frac{F_{im}}{F_{obj}} = \frac{F_{im}}{F_{obj}}(F_{obj} - D_{obj} + d)tan\phi$$

We want to incorporate the expressions for x'_θ and x'_ϕ into the original intensity equation. We use the conditions $\theta \ll 1$ (and $\phi \ll 1$) to make the approximations $sin\theta \approx tan\theta \approx \theta$ and $sin\phi \approx tan\phi \approx \phi$. Therefore,

$$I(\theta, \phi) \approx I_0 sinc^2\left(\frac{\pi a\theta}{\lambda}\right) sinc^2\left(\frac{\pi b\phi}{\lambda}\right)$$

We rearrange

$$x'_\theta = \frac{F_{im}}{F_{obj}}(F_{obj} - D_{obj} + d)\theta$$

to get

$$\theta = \frac{F_{obj}}{(F_{obj} - D_{obj} + d)F_{im}} x'_\theta$$

Likewise,

$$\phi = \frac{F_{obj}}{(F_{obj} - D_{obj} + d)F_{im}} x'_\phi$$

We substitute these results into the intensity equation to get

$$I(x'_\theta, x'_\phi) \approx$$

$$I_0 sinc^2\left(\frac{\pi a F_{obj} x'_\theta}{F_{im}(F_{obj} - D_{obj} + d)\lambda}\right) sinc^2\left(\frac{\pi b F_{obj} x'_\phi}{F_{im}(F_{obj} - D_{obj} + d)\lambda}\right)$$

8.1.3 Integration of Ray and Wave Optics

In the integrated ray and wave optics model, ray optics determines the direction of light rays, and wave optics determines the size of the diffraction pattern on the sensor. We continue to do backwards ray tracing on multiple aperture samples, but each ray creates a diffraction pattern on multiple sensor pixels. The combined model is equivalent to blurring the result of the ray optics image by applying a low pass filter, and the blur matrix is a 2D array of intensities from the Gaussian diffraction pattern. We apply a different blur matrix for each color because different colors have different wavelengths and therefore diffraction patterns of varied sizes.

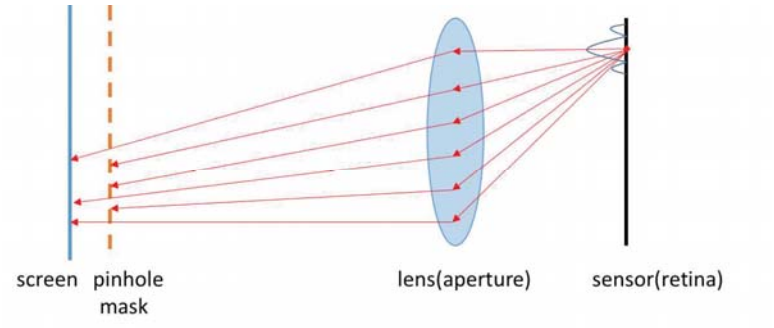


Figure 8.3: Wave Optics Simulation

Listing 8.1: Pseudocode For Wave Optics Simulation [1]

```
int[][][] temp_sensor = new int[sensor_size][sensor_size][3];
int[][][] sensor = new int[sensor_size][sensor_size][3];
// Iterate over sensor pixels
for (int iy = 0; iy < sensor_size; iy++) {
  for (int ix = 0; ix < sensor_size; ix++) {
    int color[3] = {0, 0, 0}; // R, G, and B
    int hits = 0;
    // Iterate through random aperture points
    for (int i = 0; i < aperture_sample_time; i++) {
      float2 sensor_pos = (ix, iy);
      float2 aperture_pos = (aperture_samples[2*i], aperture_samples[2*i + 1]);
      type, value = getRayColor(sensor_pos, aperture_pos);
      if (value >= 0) {
        hits++;
            color[type] += value;
      }
    }
    temp_sensor[ix][iy] = color;
  }
}

// Iterator over sensor pixels
for (int iy = 0; iy < sensor_size; iy++) {
  for (int ix = 0; ix < sensor_size; ix++) {
    // Apply blur over each sensor pixel
    for (int x = 0; x < diffractMatrix.height; x++) {
      for (int y = 0; y < diffractMatrix.width; y++) {
        sensor[ix + x - diffractMatrix.height / 2][iy + y -
        diffractMatrix.width / 2] += diffractMatrix[x][y] * color * 3 / hits;
      }
    }
  }
}
```

8.2 Results

We compare the images produced by the physical experiment, ray optics simulation, and wave optics simulation. The parameters for all three cases are the same as those described earlier in the physical experiment section (380 millimeters focus distance, 250 millimeters display distance, 6 millimeters depth of pinhole mask, 6 millimeters diameter of human eye or f/8 stop, and 20 millimeters focal length). In addition, to see how well the ray and wave optics model account for diffraction, we examine the image quality of multiple pinhole sizes.

8.2.1 Metrics

The two metrics used to measure image quality are DRIM (dynamic range imagery) and RMSE (root mean square error).

[1] Note that the diffraction matrix varies by color. This is not shown in the pseudocode.

DRIM

Aydin et. al 2008 [2] describes DRIM as a metric that enables comparison of images with different dynamic ranges. The paper describes three measurements of image distortion: loss of visible contrast, amplification of invisible contrast, and reversal of visible contrast. We focus on loss of visible contrast, which occurs when contrast that was visible in the reference image becomes invisible in the test image. The algorithm in the paper applies a loss of visible contrast predictor and then a high-pass or low-pass filter. Figure 8.4 shows the distortion maps for the wave optics model for a 75 micrometer pinhole:

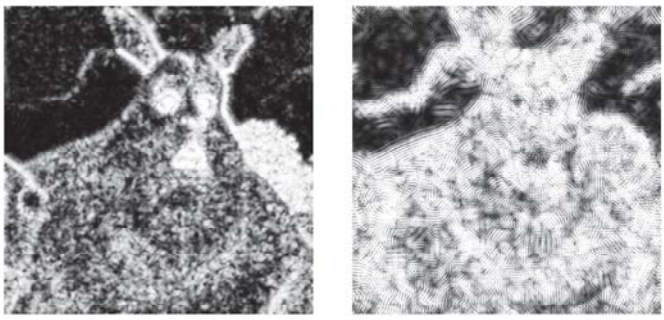


Figure 8.4: Contrast Loss High Pass (left) and Contrast Loss Low Pass (right)

The bright pixels represent areas of contrast loss. We compute the contrast loss as the pixel sum of the low-pass distortion image plus the pixel sum of the high-pass distortion image divided by two.

RMSE

RMSE is a measurement used to measure how close the pixel values of the test image are to the reference image. I_r in the equation below represents the RGB (red, green, and blue) intensities of the reference image, and I_s represents the RGB intensities of the simulated image.

$$RMSE = \sqrt{\sum_{y=0}^{\text{image length}} \sum_{x=0}^{\text{image width}} \sum_{c \in \{r,g,b\}} (I_r(x,y,c) - I_s(x,y,c))^2}$$

8.2.2 Comparison of Ray and Wave Optics

For the ray optics model, the larger the pinhole size, the worse the contrast. The pinhole size that gives the least loss of contrast is 25 micrometers. A pinhole of size 120 micrometers (contrast loss of about 600,000) generates twice the amount of contrast loss as a pinhole of size 25 micrometers (contrast loss of about 300,000). These results are expected because smaller pinholes block more light rays, which leads to sharper image contrast. For the combined ray and wave optics model, the contrast loss from different-sized pinholes is roughly the same (between 640,000 and 770,000). We find that the benefits of a small aperture in filtering light are counterbalanced by the larger diffraction pattern that is created. The smallest pinhole

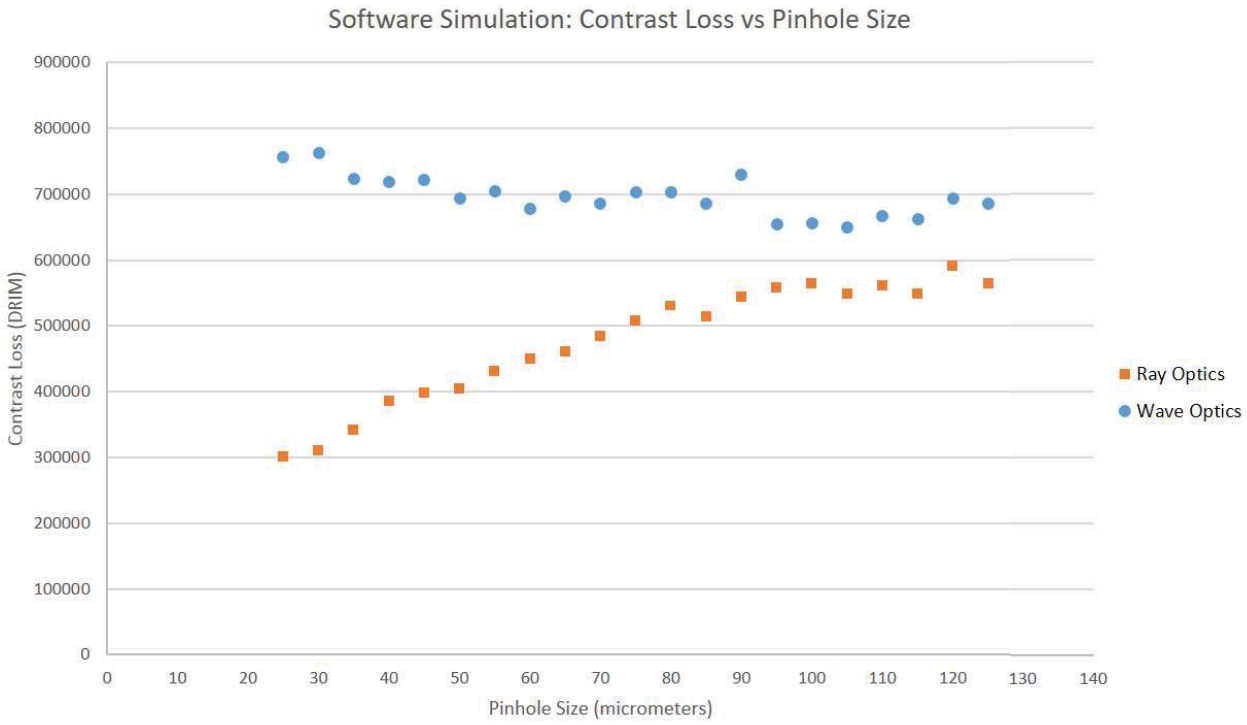


Figure 8.5: DRIM Graph

sizes create the largest diffraction patterns and therefore have the most severe contrast loss. From 25 micrometers to 105 micrometers, there is a weak trend of improving contrast due to the effects of diffraction outweighing the sharper focus of light through the pinhole. From 105 micrometers to 125 micrometers, there is a weak trend of declining contrast because at that point, the diffraction effects are very minor, and the larger pinhole sizes reduce the depth of field and do not focus the light source well. A pinhole size of 105 microns generates an image with the most contrast.

For the RMSE metric, the wave optics model features improvement over the ray optics model, despite the fact that the wave optics model produces a blurred version of the ray optics model. For both models, small pinhole sizes create an amplification of contrast, which leads to a larger RMSE. In addition, since small pinhole sizes allow significantly fewer light rays to reach the screen, there is a greater possibility of having too many rays of one color (red, green, or blue) and getting a noisy image (see figure 8.7). Because blurring reduces this amplification of contrast, larger pinhole sizes have smaller RMSE values for the ray optics model, and the wave optics model consistently produces images with smaller RMSE than the ray optics model. The wave optics model shows a trend of increasing RMSE for pinhole sizes 75 micrometers and above, because at that point, there is no more amplification of contrast, and blurring only causes the modeled image to deviate further from the clean image. Although a smaller RMSE is expected to indicate a image with pixel values close to the original image, it actually has a high chance of indicating more blur. Therefore, RMSE is a poor indicator of contrast and mediocre indicator of image quality.

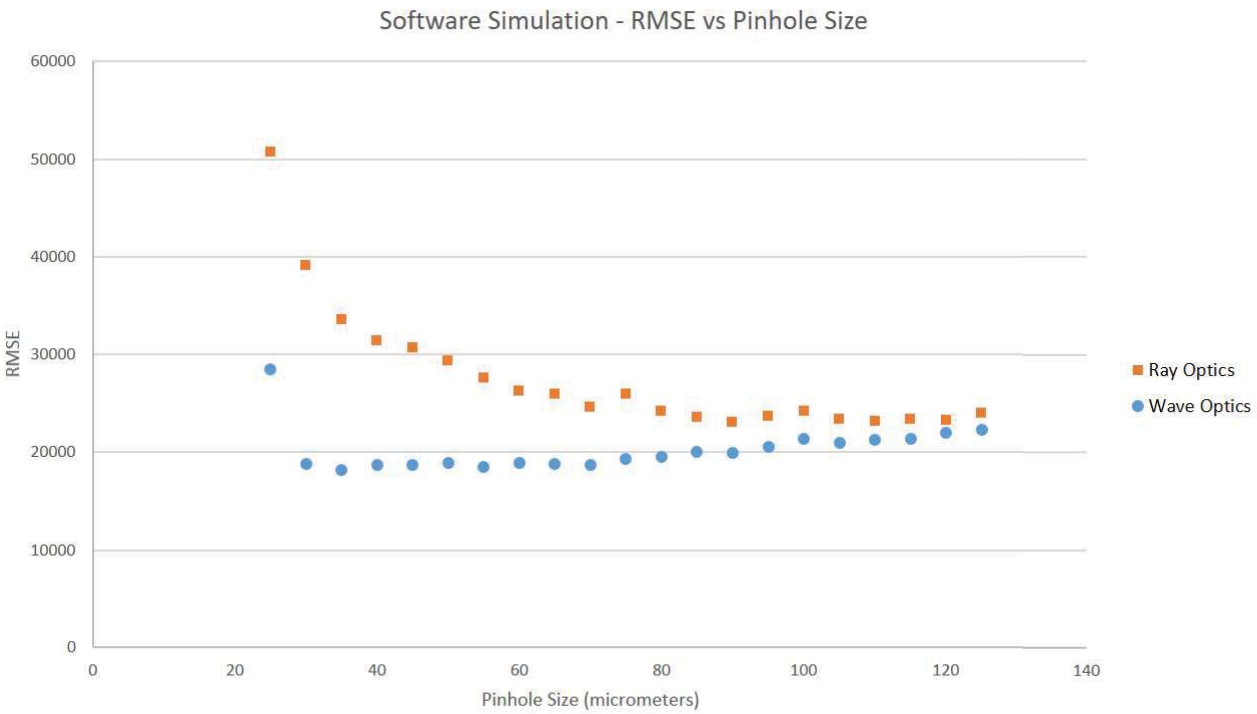


Figure 8.6: RMSE Graph

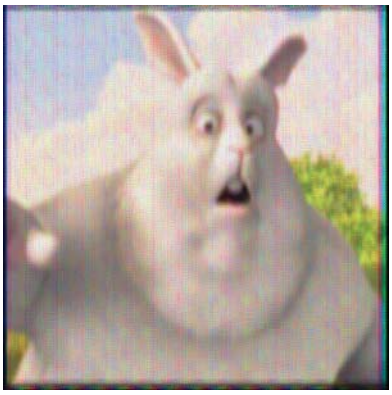


Figure 8.7: Ray Optics Simulation for 25 micrometer pinhole mask

8.2.3 Comparison of Physical Experiment and Software Simulation

We compare the images produced by the physical experiment, ray optics simulation, and wave optics simulation for a 75 micrometer pinhole. The wave optics simulation is slightly more blurred than the ray optics simulation, which is expected. Interestingly enough, the physical simulation generates the clearest image, but the exposure had to be inflated in order for the physical simulation to work. If one looks closely at the physical experiment image and the ray simulation image, one will find that a grid pattern is visible on the physical

Figure 8.8: Left: Physical Experiment, Center: Ray Optics Simulation, Right: Wave Optics Simulation

Figure 8.9: Zoomed Version of Figure 8.8

experiment image and especially the ray optics image, but is removed from the wave optics image due to diffractional blur. This shows that perhaps the wave optics simulation created a blur pattern that was too large for the 75 micrometer pinhole.

8.3 Conclusion and Future Work

The physical experiment and the software simulation give mixed results. The prefiltering algorithm combined with the pinhole mask display yields a significant improvement to the defocused image. The ray optics model yields an image that has slightly less contrast than the one in the physical experiment and recommends using the smallest pinhole size to achieve the best contrast as long as RMSE is not too high. The wave optics model gives even more blurred images than the ray optics model and recommends using a larger pinhole size of approximately 105 micrometers to achieve an image with the best contrast. There exists a slight contrast gap between the physical experiment and software simulations that needs to be resolved.

In the future, we will try to find more ways to improve on the ray and wave optics model to look more like the physical experiment. One area to expand on in the wave optics model is to consider the light ray as a Gaussian beam rather than a point source. In addition, we would like to test the ray and wave optics models for astigmatism, another form of lower

order aberrations, and higher order aberrations like coma and trefoil. We would also be interested in running physical experiments with pinhole masks for different sizes to confirm the results of the ray and wave optics models.

Chapter 9

Comparison of Different Algorithms

We compare the performance of the three different algorithms used in this report by running the ray optics simulation. For Huang's algorithm, we consider both a viewing angle of 0 degrees and multiple viewing angles from -5 to 5 degrees in 1 degree increments. We also test the performance of the forward method and backward method on different angular resolutions (3x3 pinhole/lenlet vs 5x5 pinhole/lenslet). Potential advantages of the 3x3 pinhole or lenslet include improved resolution. However, the 3x3 pinhole filters out less light than the 5x5 pinhole, which can lead to more blurring.

For all simulations, we use a screen size of 640x640 pixels, an object distance of 250 mm, and a focus distance of 380 mm. We use a depth of 6 mm for all algorithms except for the backward method with the pinhole mask (still use depth 6 mm for the backward method with the lens array) which uses a depth of 9 mm. The backwards method for the pinhole mask yields poor results at a depth of 6 mm because the prefiltering algorithm ignores too many pixels around the center pixel of the 3x3 or 5x5 superpixel. Thus, the prefiltered image is a bit dark, and the simulation is distorted by grided shapes. A larger depth will enable more screen pixels to pass through the pinhole because the angle of the ray traveling from the pixel to center of the closest pinhole is smaller. The metrics used to evaluate are contrast loss and RMSE from the chapter 8 as well as a new metric, peak signal-to-noise ratio.

9.1 PSNR

PSNR, or peak signal-to-noise ratio, measured in decibels, is used as a image quality measurement between an original and a compressed image. It is a ratio between the maximum power of a signal and the corrupting noise. The higher the value of PSNR, the better the quality of the compressed or reconstructed image.

We first define the mean square error (MSE), which is the cumulative squared error between the original image and the reconstructed image. Let I_1 and I_2 be the two images and M and N be the dimensions of I_1 and I_2, respectively.

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N}$$

Let R be the maximum fluctuation in the input image data, or difference between the

largest possible value and smallest possible value. For example, in the RGB image where each channel is between 0 and 255, R is 255. PSNR is computed by the following equation [15].

$$PSNR = 10log_{10}\left(\frac{R^2}{MSE}\right)$$

9.2 Results

Below are a copy of the original image and a blurred image, for reference.

Table 9.1: Original Images for Reference

Bunny Image	Word Image
	

The results are displayed in the next three pages.

9.3 Analysis

Huang's algorithm at one angle yields a very dark prefiltered image. Many pixels appear to be blocked by the pinhole, and the non-black pixels have very small values, likely due to normalization by n (see chapter 4). Huang's algorithm at multiple angles produces somewhat brighter prefiltered image, but still nowhere near that of the forward and backward methods. The pinhole mask simulation and the lens array simulation look very similar for Huang's algorithm at one angle and multiple angles. The darkness in the simulation leads to high RMSE, high mean squared error, and thus low PSNR. The bunny would be clear for Huang's algorithm at one angle if the grid was not blocking. Huang's algorithm for multiple angles produces the clearest bunny in terms of resolution.

The forward method yields moderately clear images with the pinhole mask display and very clear images with the lens array display with the exception of the RGB stripes. A 3x3 superpixel leads to more contrast loss than a 5x5 superpixel for both the pinhole mask and the lens array. In terms of both image quality or PSNR and image accuracy or RMSE, an

Table 9.2: Huang's Algorithm and Forward Method

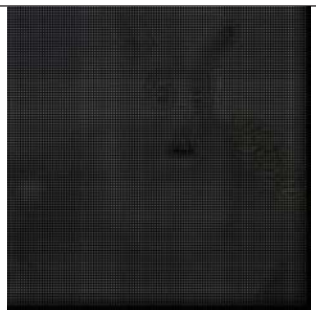
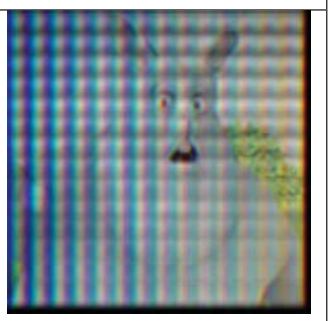
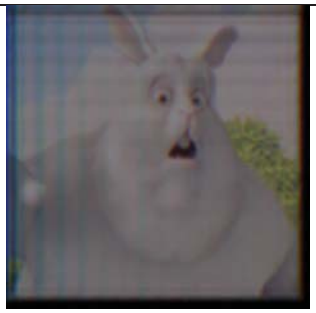
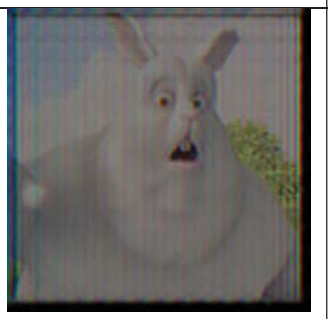
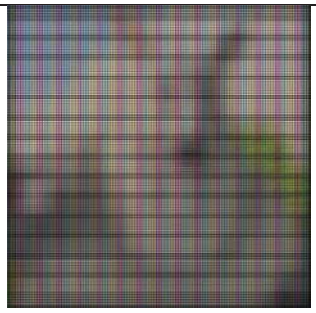
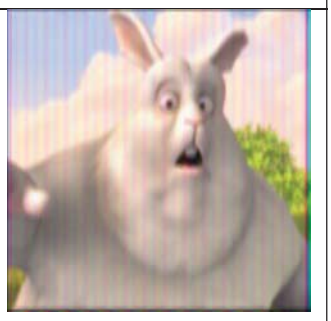
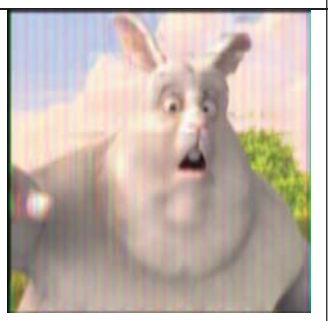
Algorithm	Prefiltered Image	Pinhole Mask Simulation	Lens Array Simulation
Huang's Algorithm at One Angle			
Huang's Algorithm at Multiple Angles			
Forward Method with 5x5 superpixel			
Forward Method with 3x3 superpixel			

Table 9.3: Backward Method

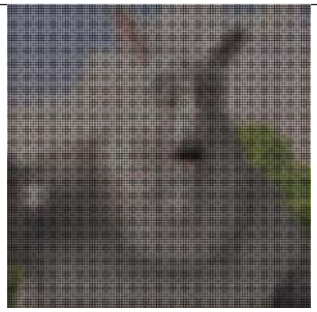
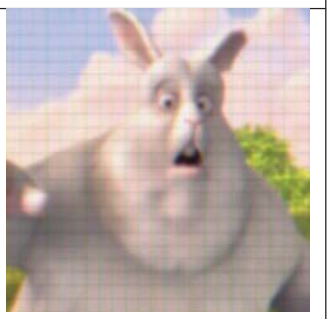
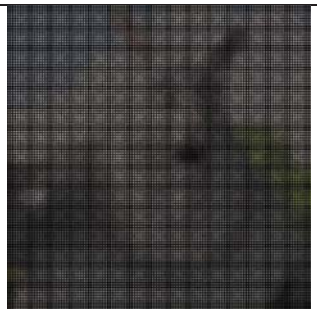
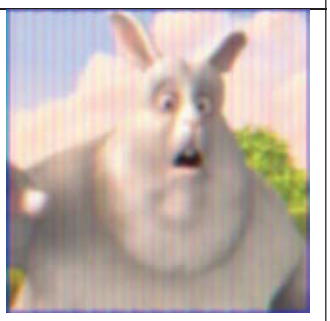
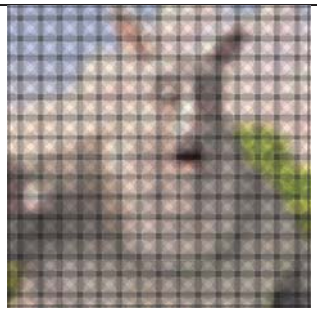
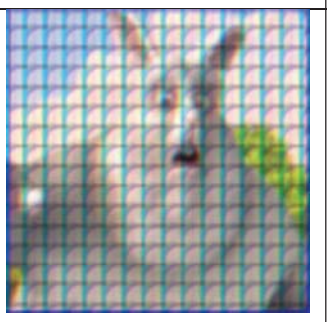
Algorithm	Prefiltered Image	Simulation
Backward Method for 5x5 Pinhole Mask		
Backward Method for 5x5 Lens Array		
Backward Method for 3x3 Pinhole Mask		
Backward Method for 3x3 Lens Array		

Table 9.4: PSNR

Algorithm	Pinhole Mask	Lens Array
Huang's Algorithm at One Angle	169.6088	170.1145
Huang's Algorithm at All Angles	167.6972	167.9586
5x5 Forward Method	193.3367	195.9025
3x3 Forward Method	197.2381	188.6442
5x5 Backward Method	197.6123	198.9921
3x3 Backward Method	203.575	181.2041

Table 9.5: RMSE

Algorithm	Pinhole Mask	Lens Array
Huang's Algorithm at One Angle	96197.46516	93412.11287
Huang's Algorithm at All Angles	107409.2307	105809.3705
5x5 Forward Method	24449.74558	21083.34326
3x3 Forward Method	19499.37489	32035.9960
5x5 Backward Method	19162.62383	17592.97749
3x3 Backward Method	13452.64547	49369.9328

Table 9.6: Contrast Loss

Algorithm	Pinhole Mask	Lens Array
Huang's Algorithm at One Angle	439806.6748	500561.2524
Huang's Algorithm at All Angles	526162.774	577569.688
5x5 Forward Method	507178.0664	587265.172
3x3 Forward Method	678130.4714	610534.7143
5x5 Backward Method	372767.6833	646387.2938
3x3 Backward Method	655407.2162	376476.1546

angular resolution of 3 is better for the pinhole mask, and an angular resolution of 5 is better for the lens array. The forward method performs much better than Huang's algorithm in terms of both PSNR and RMSE due to the little amount of light lost, but the simulated images for Huang's algorithm at multiple angles have the clearest bunny features (eyes, nose, mouth, etc.).

The backward method produces images of highly varying quality due to the different depths chosen and therefore the prefiltering image that is produced. The 5x5 pinhole mask simulation shows a clear image with a grid outline of a few pixels that receive little light. The 3x3 pinhole mask simulation looks equally clear and has no pixels that receive little light. The 5x5 lens array produces an image similar to that of the forward method, but has slightly more severe RGB stripes. The 3x3 lens array produces a completely distorted image. With the exception of the 3x3 lens array simulation, the backward method produces images with much higher PSNR and RMSE than Huang's algorithm and noticeably better RMSE and slightly better PSNR than the forward method. This shows that the backward method is able to retain more information about the original image than the two other algorithms. Finally, the 5x5 pinhole mask simulation yields the least amount of contrast loss out of all simulations, while the 3x3 pinhole mask and 5x5 lens array simulations have slightly higher contrast loss than the average forward method simulation.

Chapter 10

Conclusion

In conclusion, the results of the physical experiment and software simulation show that the vision correcting display drastically improves the quality of vision for people with eye aberrations. As the number of people with vision problems increases and the time people spend on digital screens increases, new methods for correcting aberrations will become more and more necessary. The backward method proposed runs faster than the comprehensive version of Huang's algorithm and produces clearer images than the forward method depending on the light field display and angular resolution. The forward method and simulation algorithm work well regardless of the pixel arrangement of the device. The wave optics simulation takes into account diffraction through the pinhole mask and has many applications in optical physics.

In the future, vision correcting displays will present a viable alternative to eyeglasses, contact lenses, and refractive eye surgery for people who want to view screens clearly. With the rapid increase in computing power, the algorithms will be able to run in real-time. Eye tracking software will be able to determine the parameters like distance to the eye. The next time a person watches TV or reads from a phone, he or she will not even remember his or her eye problems.

Bibliography

[1] Henry Lipson Ariel Lipson Stephen G. Lipson. *Optical Physics*. 4th ed. Cambridge University Press, 2011. ISBN: 0521151929.

[2] Tunç Ozan Aydin et al. "Dynamic range independent image quality assessment". In: *ACM Transactions on Graphics (TOG)* 27.3 (2008), p. 69.

[3] Brian A. Barsky. "12.1: Invited Paper: An Overview of Vision Realistic Rendering and Vision Correcting Displays". In: *SID Symposium Digest of Technical Papers* 46.1 (2015), pp. 138–142. ISSN: 2168-0159. DOI: 10.1002/sdtp.10326. URL: http://dx.doi.org/10.1002/sdtp.10326.

[4] Arnaldo Dias-Santos et al. "Higher order aberrations in amblyopic children and their role in refractory amblyopia". en. In: *Revista Brasileira de Oftalmologia* 73 (Dec. 2014), pp. 358 –362. ISSN: 0034-7280. URL: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0034-72802014000600358&nrm=iso.

[5] GlassesCrafter.com. *What Percentage of the Population Wears Glasses?* 2010. URL: http://glassescrafter.com/information/percentage-population-wears-glasses.html.

[6] Eugene Hecht. *Optics*. 2nd ed. Addison Wesley, 1987.

[7] F. Huang et al. "Eyeglasses-free Display: Towards Correcting Visual Aberrations with Computational Light Field Displays". In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 33.4 (2014), pp. 1–12. URL: http://web.media.mit.edu/~gordonw/VisionCorrectingDisplay/.

[8] Fu-Chung Huang. "A Computational Light Field Display for Correcting Visual Aberrations". PhD thesis. EECS Department, University of California, Berkeley, 2013. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-206.html.

[9] Fu-Chung Huang et al. "Correcting for Optical Aberrations Using Multilayer Displays". In: *ACM Trans. Graph.* 31.6 (Nov. 2012), 185:1–185:12. ISSN: 0730-0301. DOI: 10.1145/2366145.2366204. URL: http://doi.acm.org/10.1145/2366145.2366204.

[10] *Inside the Samsung Galaxy S5*. 2014. URL: https://www.chipworks.com/ko/node/126.

[11] iscreenvision.com. *Hyperopia FAQs*. URL: http://www.iscreenvision.com/faqs/hyperopia-farsightedness/.

[12] Gary Keiting. *Why Myopia Progression Is A Concern*. 2016. URL: http://www.allaboutvision.com/parents/myopia-progression.htm.

[13] Marc Levoy. *Light Field Sensing*. 2008. URL: https://graphics.stanford.edu/talks/lightfields-uncc-10jun08-public.pdf.

[14] Dong C. Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* 45.1 (1989), pp. 503–528. ISSN: 1436-4646. DOI: 10.1007/BF01589116. URL: http://dx.doi.org/10.1007/BF01589116.

[15] MathWorks. *PSNR*. URL: https://www.mathworks.com/help/vision/ref/psnr.html.

[16] NEI. *Farsightedness*. URL: https://nei.nih.gov/healthyeyes/hyperopia.

[17] Sathyanarayan Rao. *Fraunhofer Diffraction of Light by a Rectangular Aperture*. 2014. URL: https://www.mathworks.com/matlabcentral/fileexchange/47029-fraunhofer-diffraction-of-light-by-a-rectangular-aperture.

[18] Austin Roorda. *Optics and Image Quality in the Human Eye*. URL: http://roorda.vision.berkeley.edu/Pubs/Optics_of_the_Eye.pdf.

[19] *Samsung Galaxy Note II N7100*. 2012. URL: http://www.gsmarena.com/samsung_galaxy_note_ii-review-811p3.php.

[20] Susan Vitale, Robert D. Sperduto, and Frederick L. Ferris. "Increased prevalence of myopia in the United States between 1971-1972 and 1999-2004". In: *JAMA Ophthalmology* 127.12 (Dec. 2009), pp. 1632–1639. ISSN: 2168-6165. DOI: 10.1001/archophthalmol.2009.303.

[21] Eric W. Weisstein. *Zernike Polynomial*. URL: http://mathworld.wolfram.com/ZernikePolynomial.html.

[22] Zehao Wu. "Investigating Computational Approaches and Proposing Hardware Improvement to the Vision Correcting Display". MA thesis. EECS Department, University of California, Berkeley, 2016. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-67.html.

[23] Norman Yan. *Samsung Shows Off The New Diamond Pixel Layout Behind The S4's AMOLED Display*. 2013. URL: https://www.androidheadlines.com/2013/04/samsung-shows-off-the-new-diamond-pixel-layout-behind-the-s4s-amoled-display.html.