Algorithms for Displays that Correct Nearsighted and Farsighted Vision

Tarkan Al-Kazily, Melody Mao, Eric Liu, Chun-Ning Tsao, Wei-Chun Hwang, Eric Ying

Executive Summary

Millions of people are affected by nearsightedness and farsightedness, keeping them from reading screens without prescription lenses (Holden et al. 2016). Glasses and contacts can be expensive and inconvenient to change between, especially for those who need different prescriptions for different situations. We work with Vision Correcting Displays (VCD) to correct nearsighted and farsighted vision problems, by computationally reversing the blur the user sees to produce clear images.

Our project seeks to build upon years of previous research by quantitatively analyzing VCD algorithms, existing and novel, in a methodical fashion. We present three contributions to VCD research: 1) A C++ software library in which we document and implement vision correcting algorithms in a modular way. 2) Two new algorithms, the Parametrized Many to Many algorithm and the Forward Optimization algorithm, that are chosen to indicate strengths and weaknesses of previous algorithms. 3) A holistic testing methodology that we apply to quantitatively analyze and compare the performance of each of these algorithms.

In this report, we introduce the necessary background and previous research in VCDs and explain our project methodology. Then, we present our test results in terms of runtime and image quality analysis, along with our insights into the distinctions between each algorithm. In order to aid in future research and capstone teams, we suggest potential algorithm approaches as well as specific improvements that will be necessary for future testing methodologies. Our

research contributes to Vision Correcting Displays to help those with vision problems use the many screens that are necessary in our daily lives.

Table of Contents

Section 1: Literature Review: Nearsighted and Farsighted Vision Problems and Computational Vision Correction

1. Introduction: Correcting Nearsighted and Farsighted Vision with Vision Correcting Displays

Millions of people around the world suffer from vision problems, preventing them from clearly reading displays without prescription lenses (Holden et al. 2016). These vision problems can be characterized as high and low order aberrations, with high order aberrations like keratoconus being complicated and difficult to correct. In contrast, low order aberrations like nearsightedness and farsightedness are correctable by prescription lenses and are much more common. In particular, nearsightedness in children is becoming more prevalent in part thought to be due to the increase in display usage in our daily lives (Hashemi et al. 2017).

As a way to alleviate these vision problems and make displays readable without glasses, Vision Correcting Displays (VCD) were invented. Vision Correcting Displays combine computer algorithms and display hardware to modify displayed images in order to produce a clear image in the user's perception without prescription lenses. While several VCD solutions have been proposed over the past decade, previous VCDs were designed to correct general vision problems. In this work, we focus on algorithms that correct the more prevalent low order aberrations of nearsightedness and farsightedness. This allows us to utilize a simplified model to approximate the way users see the display. We present three contributions to VCD research: 1) A C++ software library in which we document and implement vision correcting algorithms in a modular way. 2) Two new algorithms, the Parametrized Many to Many algorithm and the Forward Optimization Algorithm that are chosen to indicate strengths and weaknesses of

previous algorithms. 3) A holistic testing methodology that we apply to quantitatively analyze and compare the performance of each of these algorithms.

The rest of this section serves to provide background information and to review relevant literature of VCDs. In Part 2 we review how VCDs and their underlying display hardware have improved over time. We primarily work with a type of display called a light field, which provides individual control of the directions light rays are emitted from the display. In Part 3 we describe both previous and newly proposed computational algorithms, called prefiltering algorithms, used for light-field-based VCDs. We build off prior work and use some of these algorithms as benchmarks for comparison with our algorithms. In Part 4 we describe the fundamental ray tracing method we use to model low order vision aberrations, originally introduced by Wang and Xiao (2013).

2. Existing Work: The Evolution of Vision Correcting Displays

Correcting vision computationally with displays has its roots in trying to understand how the eye sees the world. Brian Barsky (2004) introduced the concept of Vision Realistic Rendering, rendering scenes to simulate the image seen from a user's eye. This was done with a wavefront map of the user's eye to characterize the high and low order visual aberrations defining the user's vision problems. This idea of computationally simulating a user's vision naturally led to the idea of computationally correcting vision as part of the display process.

The general process to computationally correct vision involves both software and hardware. In software a "prefiltering" algorithm operates on the image to be displayed in order to compute a modified version that will resolve to the corrected image on the user's retina when displayed by the hardware component. Researchers realized this idea by applying various models of vision problems and incorporating new types of display hardware. Early approaches used standard LCD displays and treated the user's visual aberrations as a convolutional blur. By applying a matching inverse blur the user would see a clearer image; effectively prefiltering by deconvolving the image (Alonso and Barreto 2003). While applying this inversion process accounted for the user's vision problems, the displayed image had poor contrast and additional artifacts on normal displays. Huang et. al (2012) refined this approach by using multilayered displays with two specially chosen deconvolutions for the separate layers of the display. They obtained image results with fewer artifacts and better image quality. However, the results from the multilayer display and deconvolution process still appeared washed out, with low contrast in the displayed image.

The next major development came when Pamplona et al. (2012) first adopted light field displays for VCDs. Light field displays allow for greater control over the direction in which light is

emitted from each display pixel. Pamplona et al. (2012) and Huang et al. (2014) each demonstrated VCD methods using light field displays. Huang et al. physically implemented a light-field-based VCD using a pinhole mask layered in front of a standard LCD screen, which successfully corrected vision but blocked a significant amount of light. They also investigated microlens-array-based light field displays and concluded that microlens arrays would provide similar correction capabilities while allowing the full image brightness to shine through. Due to the promising results they have shown, we continue to focus on light-field-based VCDs and algorithms tailored to correcting low order aberrations.

3. Recent Developments: Light Field Based Prefiltering Algorithms

Prefiltering algorithms for light field displays began with two different directions of research from Pamplona et al. (2012) and Huang et al. (2014). These approaches demonstrate different ends of a spectrum in terms of light ray sampling amount and the resulting prefiltering computation approach. The algorithm Pamplona et al. developed traces individual light rays from each pixel on the display to the user's eye, incorporating additional information such as cataracts, and uses the final position on the retina to directly decide what color to display at that pixel. This simple, yet functional approach is most similar to what we call the Point to Point algorithm, and is the first prefiltering algorithm we analyze due to its simplicity when considering patients with only low order aberrations. In contrast, Huang et al. (2014) used a more robust prefiltering process where multiple light rays would be traced from each pixel on the retina in order to construct a linear map of rays from the retina to the display. With this map, Huang's algorithm solves a least squares optimization to obtain a best fit prefiltered image that would appear clear to the user. This method models more light rays at the cost of increased computation time.

Other researchers at UC Berkeley have continued to produce new advancements in prefiltering. Zhen (2016) developed algorithms specific to pinhole mask displays, theorizing an improvement from sampling multiple rays over the full area of each display or retina pixel, rather than the single ray used in Pamplona's original algorithm. This area concept gives rise to new algorithms, Point to Area and Area to Area, the latter of which is evaluated in our work, in addition to another version of our Parametrized Many to Many algorithm. Wu (2016) characterized the projection from display to retina as a function used in calculations to compute

the prefiltered image. We draw from Wu's characterization of the projection in our implementation redesign.

4. Optics Background: A Low Order Ray Tracing Light Model

For any prefiltering process, regardless of the algorithm used, we must define how we model the eye and its many refractive components. The general ray tracing method used in the aforementioned approaches is built on the thin lens assumption of optics in the eye. Specifically, we use the abstracted optical model of the human eye introduced by Wang and Xiao (2013), which effectively models low order aberrations like nearsightedness and farsightedness. They proposed using ray tracing in a Gaussian-like distribution to calculate how rays refract in the eye, without having to model the lens surface. The model considers only the aperture, the retina, and a convex lens that represents the cornea and other refractive elements. The image that forms on the retina is obtained as an integration of all of the light rays that hit each point, after being refracted through this system. This abstraction allows us to simplify calculations without sacrificing too much accuracy in our model. In this way, after prefiltering an image, we can also simulate how a user with low order aberrations will view our light field and the image displayed.

Section 2: Technical Work: Developing Vision Correcting Algorithms for Nearsighted and Farsighted Vision

1. Overview: Our Project Organization and Research Approach

To develop and evaluate our algorithms, we pursue two directions; the first is a software library in which we organize and implement our prefiltering algorithms, and the second is simulation and experiment methodology to evaluate the algorithm results. By implementing our algorithms in an object oriented library we structure them to be modular and provide an extendable framework for future research. Part 2 explains this library structure. Part 3 details the algorithms we are studying: the Point to Point, Area to Area, Parametrized Many to Many and Forward Optimization algorithms.

We also develop a comprehensive test suite to evaluate our algorithms. Part 4 explains our experiment methodology, which is primarily based around our simulation, built off of the original simulation by Wu (2016). We use this simulation on a number of images and example user prescriptions, and quantitatively measure the simulated results using a pair of image quality metrics, Peak Signal to Noise Ratio (PSNR) and HDRVDP-V2 (Mantiuk et al. 2011). Finally, we use physical hardware methods to test our algorithm results, capturing real images using a defocused camera, as well as measuring algorithm runtimes on an embedded Raspberry Pi computer. We present our results and analysis in Part 5, followed by a discussion of future research directions in Part 6, before providing a project summary and conclusion in Part 7.

2. Our Prefilter Library: An Extendable Algorithm Structure

We design our software library for Prefiltering algorithms with a goal of clearly organizing each algorithm to be modular and extendable, allowing for more straightforward analysis and testing. Our library modularizes each algorithm by standardizing their usage to be independent of how they are implemented, while simplifying algorithm design by sharing common functionality. Our library makes these algorithms extendable by clearly specifying algorithm inputs and outputs, which allows for future algorithms to be made by implementing derived classes. This library is one of the deliverables of our project, and serves as a complementary resource to this report. It is our goal that this library and its internal wiki documentation serve as a resource and a starting point for future researchers developing Prefiltering algorithms. See Appendix A for directions to access this library.

The Prefilter library and algorithms are implemented in C++ using a class hierarchy structure, illustrated in Figure 1. Each algorithm derives from a parent BasePrefilter class which defines, on a high level, the “prefilter” operation that processes a desired image to produce the prefiltered image to display on the screen. In order to further organize our work, we also specify the ProjectionConversionPrefilter class, which implements “prefilter” with a Project step followed by a Convert step. Establishing this interface for our prefiltering algorithms more clearly separates the process into 1) the Project step, tracing the light rays from the display, and 2) the Convert step, computing the final prefiltered image. This design simplifies how software applications use the Prefilter library by only requiring them to be able to use the BasePrefilter class. This design also allows for future algorithms to be added as new derived classes of any of the existing classes, without requiring any changes to existing work.

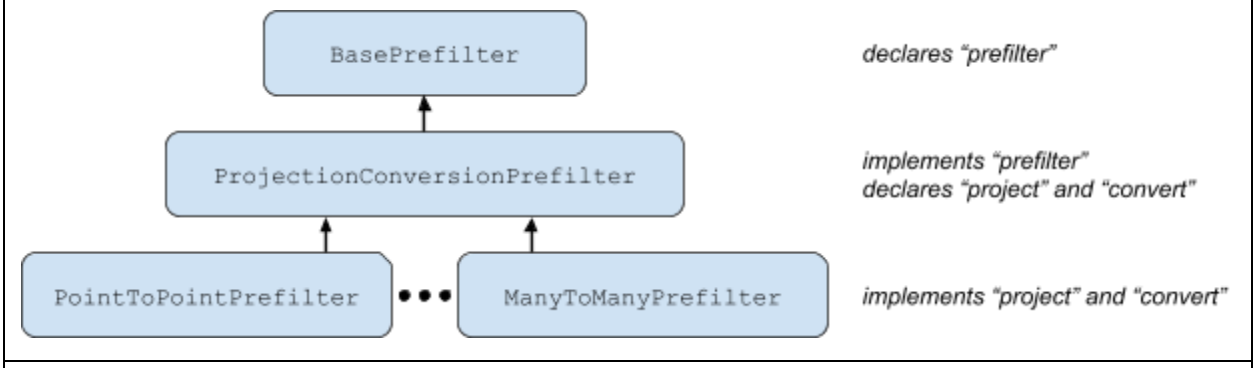


Figure 1: Prefilter library class hierarchy. The base class BasePrefilter is at the top, which specifies the "prefilter" function that all prefiltering algorithms must implement. ProjectionConversionPrefilter derives from BasePrefilter to implement "prefilter" in terms of Project and Convert steps, which are implemented by each of our algorithms.

```
Algorithm: ProjectionConversionPrefilter Outline

// Project:
Construct the projection M that traces light rays from display to retina

// Convert:
For each pixel D in the prefiltered image:
    Select the corresponding pinhole P
    Sample light ray(s) between D and P
    Using M, trace light ray(s) to endpoint(s) R on the retina
    Assign the color of D based on the light ray endpoint(s)
Return the prefiltered image
```

3. Our Prefiltering Algorithms: Diagrams and Pseudocode Explaining our Algorithms

We implement four Prefiltering algorithms, two from previous work: Point to Point and Area to Area (Zhen, 2016), and two new algorithms: Parametrized Many to Many, and Forward Optimization. These algorithms all follow the two-step process of Project and Convert as described in Part 2, but differ in the amount of light rays sampled and the methodology of the Convert step. In this way, they compute a prefiltered image that shows the desired image at the user's plane of focus when viewed on the light field display. We implement these algorithms for a pinhole-mask-based light field display, where each display pixel can project light through multiple pinholes. As such, we also need to devise a pinhole selection process in order to determine the direction each pixel would be seen from. We discuss this pinhole selection process and introduce the algorithms in the following sections.

3.1 Pinhole Selection Process

Due to simplicity and cost, we are using pinhole-mask-based light fields to test our algorithms. Figure 2 shows how a pinhole mask selectively blocks light from the display pixels, serving as a light field display. Light from pixels on the display diverges in all directions, but much of it will be blocked by the barrier of the mask, be blocked by the pupil, or miss the eye altogether. As such we can simplify our computation drastically if, for each display pixel, we can determine the best pinhole through which we can project unobstructed light rays.

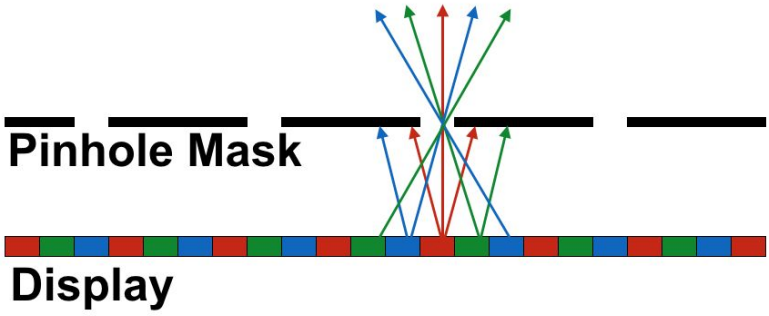


Figure 2: A 2D cross section of a pinhole-mask-based light field display. Each pixel in the display projects light in all directions, but only certain directions of light pass through the pinhole. Pinhole selection is the process of finding which pinhole a pixel shines light visible to the user.

This problem is related to an inquiry by Zhen (2016) called the One to One assumption. The One to One assumption states that each display pixel will be visible through a single pinhole at a time. Zhen proves a sufficient condition for the One to One assumption to hold by showing that for displays far enough from the user's eye, there is at most one pinhole through which the eye can see a given pixel. This depends on many factors, but generally applies when the display is at least 150 mm away from the user's eye, serving to support our algorithms.

To understand conceptually why this assumption is valid, consider an arbitrary point on the display. If there was no barrier between the display and the user's eye, the diverging light rays from that point that also reach the user's pupil would form an oblique cone. The cone is narrow near the initial point, and spreads out to the fill diameter of the pupil at the eye, but all light falling outside that cone is blocked by the pupil. The pixel is only visible through pinholes that fall inside of this cone, which is small when the distance to the display is great.

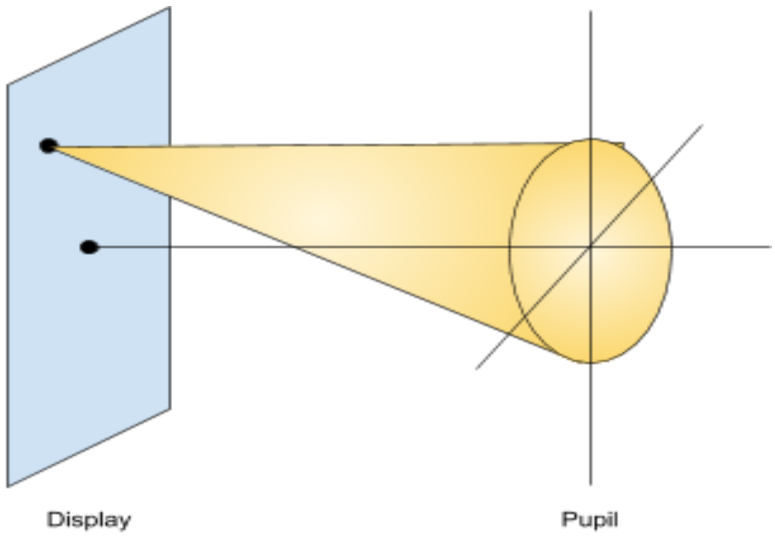


Figure 3: The cone of light visible from a point on the display

In order to determine the direction of light rays that reach the eye from a given pixel, we want to find a pinhole that intersects this cone of light. Since the eye is assumed to be looking perpendicularly at the display, the intersection area of the pinhole mask and the cone of light must be a circle. We also know that the ray from the starting point going through the center of the pupil will be inside this region. We call this the central ray, and the point where it intersects the pinhole mask will be the center of this circular region. In order to determine the radius of this region, we can use similar triangles taken from a cross section of the cone.

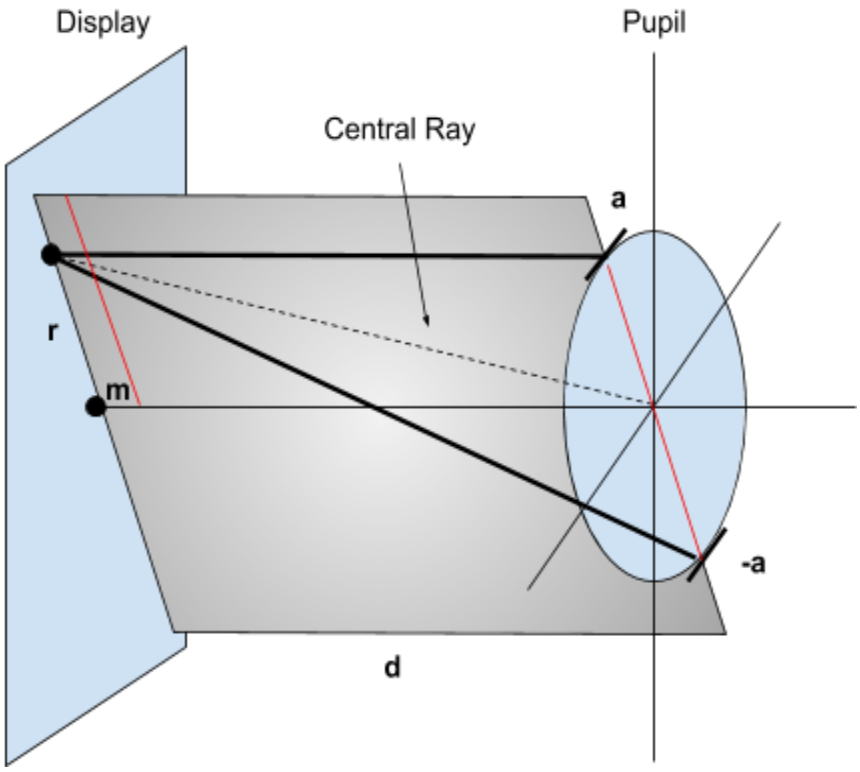


Figure 4: Annotated radial cross section of the light cone used to find the radius of the circle of intersection at the pinhole mask. **d** gives the distance from the display to the eye, **r** is the distance from the pixel to the display center, **m** is the distance between the display and the pinhole mask, and **a** is the radius of the pupil.

In Figure 4 we take the radial cross section of the cone of light, and consider the triangle formed by the intersection of this plane with the cone. Since the pinhole mask and the pupil are parallel to each other, this forms a pair of similar triangles in the diagram, sharing the same angle at the tip of the cone and parallel bases in red. Figure 5 illustrates the important 2D information in the plane. The diameter of the circle formed by the cone of light intersecting with the pinhole mask is given by the length of the smaller triangle's base. Let **m** be the offset between the display and the pinhole mask, and **d** be the distance from the display to the user's

eye. These two triangles are related by a scale factor of **m / d**. Therefore, the diameter of this circle is **2 a * (m / d)**, where **a** is the radius of the pupil.

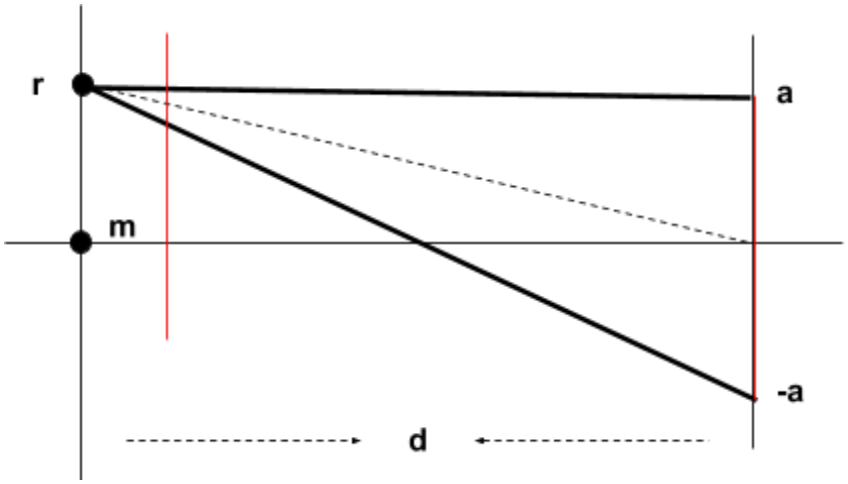


Figure 5: The same diagram as in Figure 4, redrawn to be clearer and in 2D.

This calculation tells us that we do not need to consider most pinholes to determine what directions a given pixel is visible. In fact, since we test with a display with mask offset **m = 6 mm**, pupil aperture radius **a = 2 mm**, and display distances **d > 150 mm**, the minimum circle diameter is **0.16 mm**. However, the spacing between pinholes is **0.39 mm**, determined by our angular resolution and pinhole size. As such, the One to One assumption must hold, and the only pinhole we need to consider is the one closest to the center of the circle, which is given by the point of intersection of the central ray with the pinhole mask. If light rays sampled from the pixel travelling through that pinhole reach the user's pupil, then that is the only pinhole through which the pixel will be visible. If it doesn't, then no other pinholes are closer to the circle to allow the pixel to be seen by the user anyway.

This is the idea behind our method of determining the pinhole to project light through. Given a starting point (x, y) on the display, it computes and returns the coordinates of the pinhole closest to the central ray from (x, y) through the center of the pupil. Since the intersection of a light ray and a plane can be efficiently computed by solving a linear equation, this method takes constant time.

3.2 Previous Work: Point to Point

Originally presented by Zhen (2016), this algorithm treats display pixels and pinholes as points by only considering the center of each pixel and pinhole. In the Project step, it traces one ray from each display pixel through a chosen pinhole to determine where the light ray would end up on the user's retina. Then it assigns the color of the projected pixel on the retina to the display pixel. This is the simplest algorithm in terms of sampling rate and the Convert method, and it is used as our baseline model. The crux of this algorithm relies on the One to One Assumption as described in the previous section. The One to One Assumption allows us to trace one ray for each display point, rather than multiple rays for every possible pinhole passed through, saving computation time. Given a display size of M by N pixels, the time complexity of this algorithm is O(M*N), because it iterates over each display pixel and traces a single ray per pixel.

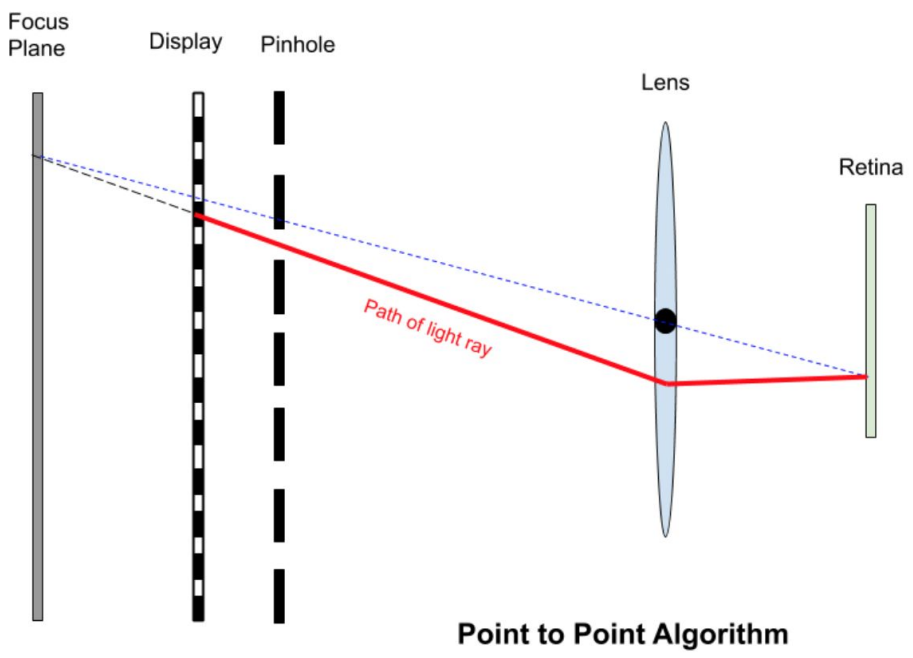


FIGURE 6: The diagram of Point to Point Algorithm, showing how a light ray is traced from a display pixel.

```
Algorithm: Point to Point
```

```
Let I be the input image placed at the retina
Let X be an empty image placed at the display
For each pixel D in X:
    Select the corresponding pinhole P
    Trace a light ray from D to P and get the endpoint R on the retina
    Assign the color of I at R to D
Return X as the prefiltered image
```

3.3 Previous Work: Area to Area

Originally presented by Zhen (2016), this algorithm treats display pixels and pinholes as square areas instead of single points, in order to account for the information loss in the Point to Point algorithm. If we trace only one ray from each pixel, and that ray is blocked by the aperture or lands outside of the retina, there is no way to assign a color to that pixel. However, by

considering light rays besides the one from the center of the display pixel to the center of the pinhole, in theory we should be able to retain more information from the original image and produce a better prefiltered image. The Area to Area algorithm realizes this by tracing two boundary rays from each display pixel and computing the prefiltered image by averaging the colors of the pixels in a projected area on the retina. Given a display size of M by N pixels and a projected area size of P by Q pixels, the time complexity is O(M*N * P*Q), because we integrate all pixel colors across the area projected from each display pixel. A more detailed derivation of projected area size can be found in Zhen (2016).

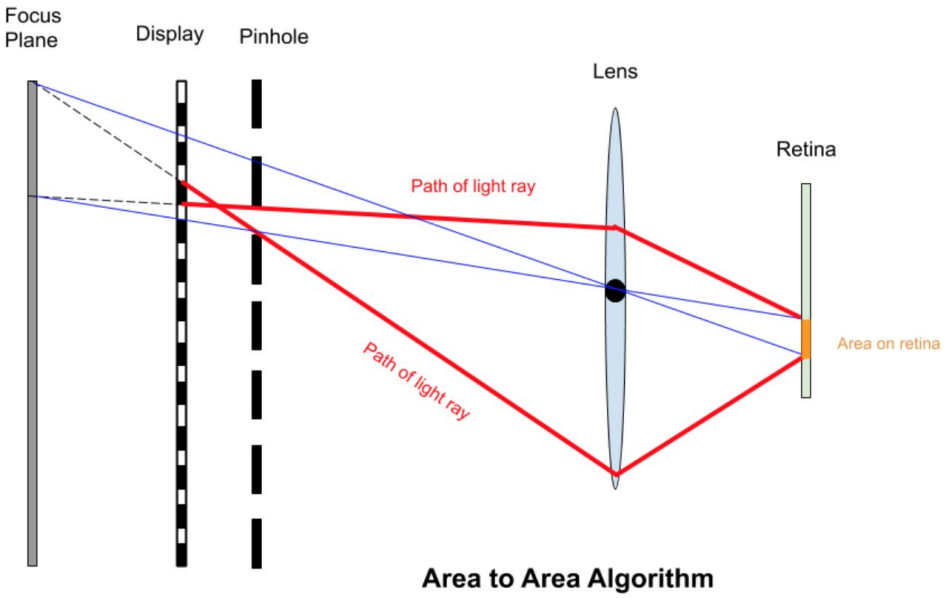


FIGURE 7: The diagram of Area to Area Algorithm, showing how two boundary light rays are traced from a display pixel.

```
Algorithm: Area to Area
```

```
Let I be the input image placed at the retina
Let X be an empty image placed at the display
For each pixel D in X:
    Select the corresponding pinhole P
    Trace a light ray from the top of D to the bottom of P
    Trace a light ray from the bottom of D to the top of P
```

```
    Get the projected area A on the retina
    Integrate the color of I within A and assign to D
Return X as the prefiltered image
```

3.4 Our Algorithm: Parametrized Many to Many

In the original Point to Point algorithm, each display pixel is simplified as a point, and only one sample light ray is traced from each point. If that sample ray is blocked by the aperture or lands outside of the retina, the algorithm loses all information about how to assign the corresponding display pixel. To account for this drawback, Zhen (2016) proposed treating display pixels and pinholes as areas and sampling more light rays for each display pixel. This concept can be realized in two ways. The first is to trace only two boundary rays from the display pixel to get the bounds of the projected area on the retina, which is the method used in the Area to Area algorithm. The second is to individually sample multiple light rays within the area, which is the basis for the Many to Many algorithm.

Building on the previous version of Many to Many by Zhen (2016), we reformalize this algorithm for non-randomized rays and added flexibility. First, we trace a specific set of rays from each pixel (by sampling across grids of points on the pixel and pinhole), rather than tracing randomly chosen rays from each pixel. Second, we explicitly parametrize the algorithm by the sampling rate, which controls the number of light rays we use to represent the display/retina area. When the sampling rate is one, the algorithm is equivalent to Point to Point, and as the sampling rate goes to infinity, the algorithm is equivalent to Area to Area with an ideal integration function (i.e. accounts for all possible light rays). In order to improve the quality of light rays sampled, we exclude light rays that are blocked by the aperture, unlike in our implementations of Point to Point or Area to Area. This produces a less blurry image than including all of the sampled rays, and reduces the average runtime (although it doesn't affect

asymptotic complexity). Given a display size of M by N pixels and a (1D) sampling rate K, the time complexity of this algorithm is $O(M*N*K^4)$, because it samples every possible pairing between K^2 display pixel points and K^2 pinhole points, for each display pixel. We develop this algorithm to be able to more closely investigate the differences in sampling between the Point to Point and Area to Area methods, and to understand if increasing sampling improves image quality. As such, we discuss how it compares for two different sampling rates, K = 2 and K = 3, in Section 5.1.

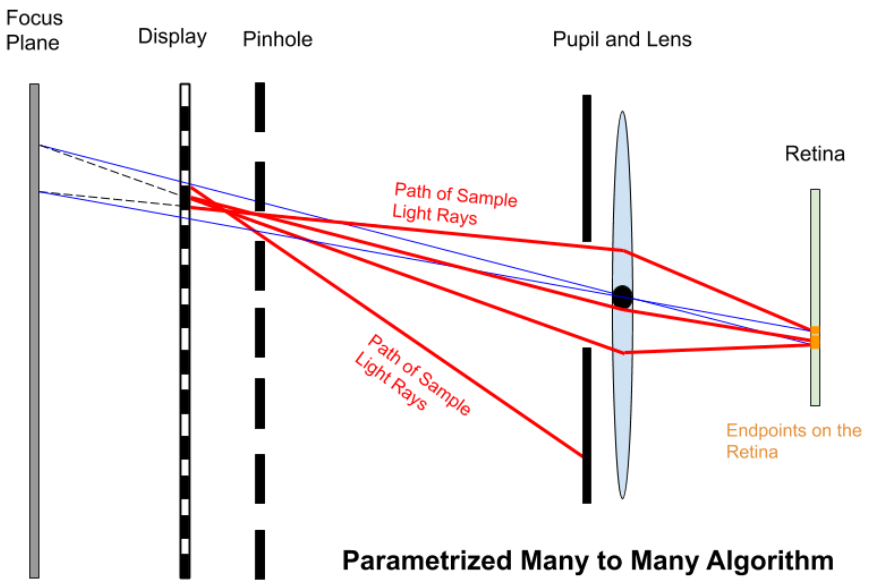


Figure 8: The diagram of Parametrized Many to Many algorithm, showing how the area illuminated by a display pixel is determined by multiple samples. Ignoring light rays that are blocked by the aperture serves to improve computation time and the final image quality.

```
Algorithm: Parametrized Many to Many
```

```
Let I be the input image placed at the retina
Let X be an empty image placed at the display
Let K>=1 be sampling rate (number of samples we take in each dimension)
For each pixel D in X:
    Select the corresponding pinhole P
```

```
    Uniformly take K^2 sample points within D and K^2 sample points within P
    For each sample point S_D within D and sample point S_P within P:
        Trace a light ray from S_D to S_P to get the endpoint S_R on the retina
    Average the colors of I at {S_R} and assign to D
Return X as the prefiltered image
```

3.5 Our Algorithm: Forward Optimization

We develop this algorithm as a variation of Fu Chung Huang's "backward" optimization algorithm (Huang et al., 2014) in order to attempt to address the lack of sharpness in the results from Point to Point, Area to Area and Parametrized Many to Many. In particular, we want to determine if significant improvements on image quality are attainable by tracing light rays from the display to the eye, as opposed to how Huang traces light rays from the user's eye to the display. As such, we trace light rays in the "forward" direction from the display to the retina in the same way as the Parametrized Many to Many algorithm. We then use these light rays to construct a matrix Q describing how light rays are projected, with which we perform bounded least squares to solve the optimal prefiltered image just as in Huang's algorithm. The matrix Q maps a column vector of pixel intensities in the prefiltered image to a column vector for what the user would see. To determine the prefiltered image, we solve the equation `I = Q * X` for `X`, where `I` is the ideal image and `X` is the prefiltered image. Since there may not be an exact solution, we use a bounded least squares optimization solver to find an `X` that minimizes $||I - Q * X||_2$.

For our tests we used a sampling rate of K=2, resulting in 16 light rays sampled for each pixel to compute the matrix M. Given a display size of M by N pixels and a sampling rate of K, the time complexity of the optimization problem setup is $O(M*N * K^4)$. There are M*N pixels in the display, and we account for K^4 rays from each pixel for $M*N*K^4$ rays total. The matrix M maps how each light ray contributes to each result pixel. The time complexity of the bounded

least squares optimization itself is O(J * M*N), based on J, the number of steps it keeps in memory, and the number of variables in the equation to solve, which in this case is M*N (the number of pixels).

```
Algorithm: Forward Optimization

Let Q be a matrix representing how light is projected from the display
Let I be the input image placed at the retina
Let X be an empty image placed at the display
Let K>=1 be sampling rate (number of samples we take in each dimension)

For each pixel D in X:
    Locate the corresponding pinhole P
    Uniformly take K^2 sample points within D and K^2 sample points within P
    For each sample point S_D within D and sample point S_P within P:
        Trace a light ray from S_D to S_P to get the endpoint S_R on the retina
        Use the pair D and S_R to construct entries in Q

Normalize the rows of Q to correct for final color quality
Construct stacked vectors R G and B from each color channel in I
Solve the bounded least squares problems for r, g, b > 0 such that:
    Q * r = R
    Q * g = G
    Q * b = B

Unstack the solution vectors r g and b to construct X
Return X as the prefiltered image
```

4. Experiment Methodology: Measure the Performance of Our Algorithms Against One Another Through a Set of Test Cases

4.1 Software Simulation

We use software simulation to efficiently run tests in parallel across a range of settings and capture results in a standardized way. Our simulator is built on the simulation first developed by Zehao Wu (Wu, 2016) in C++ that simulates how a camera sees an image on a display through a pinhole mask with 5 pixels of angular resolution. By adjusting the settings to approximate a human eye, we can then simulate users of various nearsighted and farsighted prescriptions. We adapt the simulator to use a standardized configuration file used by the prefiltering process so that the interfaces are consistent across the two programs. This also ensures that one can easily generate, record, and reproduce experiments across machines.

Initial tests of our algorithms with the simulator showed very little improvement in quality over the uncorrected images, which we hypothesized was due to the fact we didn't consider the pinhole mask limiting resolution. We resolve this by increasing the spatial resolution of the pinhole mask light field to match the ideal image resolution. Uncorrected images are shown on our display by resizing the image to use the full display size.

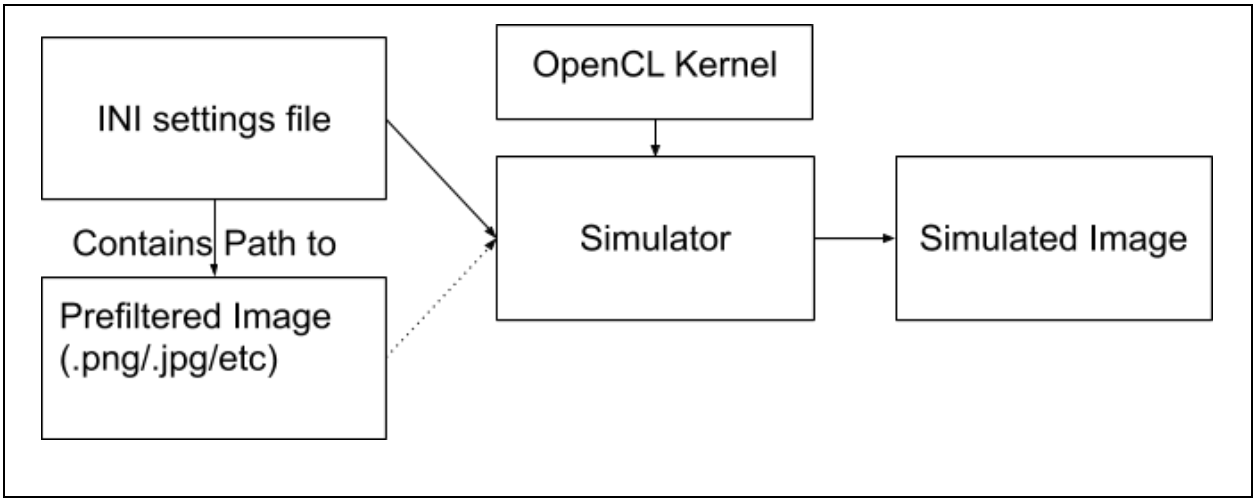


Figure 9: Block diagram of the simulator inputs and outputs.

4.2 Image Quality Metrics

We select two image quality metrics that indicate how well our prefiltering algorithms work in terms of correction quality to match the original image. The first is Peak Signal to Noise Ratio (PSNR) which represents the average of the squares of differences in corresponding pixel values between two images. The higher the PSNR, the better the images correspond. The other is High Dynamic Range Visual Difference Predictor Version 2 (HDRVDP-V2), which seeks to distinguish two images with high dynamic range by accounting for many aspects of how light is processed by the human visual system, and overall is better at predicting a human's perception of image quality (Mantiuk et al. 2011). In order to evaluate our results in comparison to a traditional display, we produce an uncorrected image by passing the original image through the simulator. We then compare our algorithms metric results to the same metrics computed from the uncorrected image. We use Matlab to iterate over each test case and compute the resulting metrics.

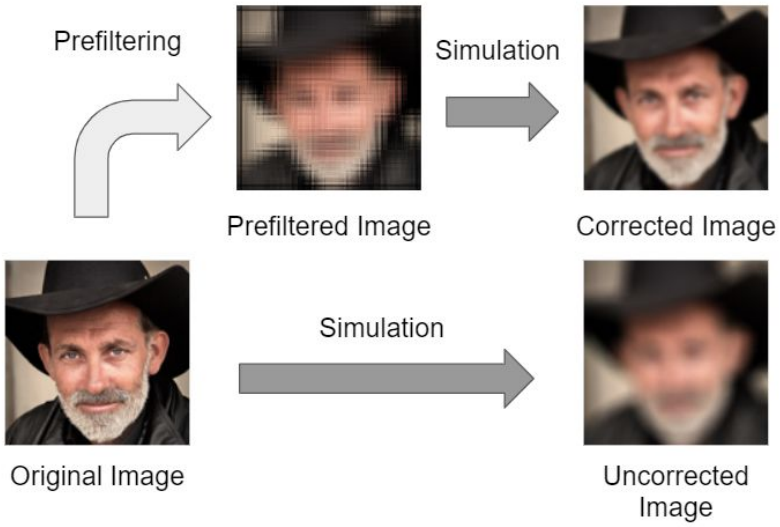


Figure 10: Flow chart of how we use our Prefiltering library and Simulation to generate corrected and uncorrected images for testing.

4.3 Overview of Test Cases

To generalize our results, we selected a set of 5 square images to cover a range of image types in our tests (Figure 11). We then test each algorithm on each image from multiple viewing distances and for multiple degrees of nearsightedness and farsightedness. For nearsighted vision, we set the focus plane to a far point, i.e. the farthest point a user with a given prescription can see clearly. For farsighted vision, we set the focus plane to a near point, i.e. the nearest point a user with a given prescription can see clearly. Nearsighted use cases test the display being further than the focus plane, while farsighted use cases test the display being closer than the focus plane.

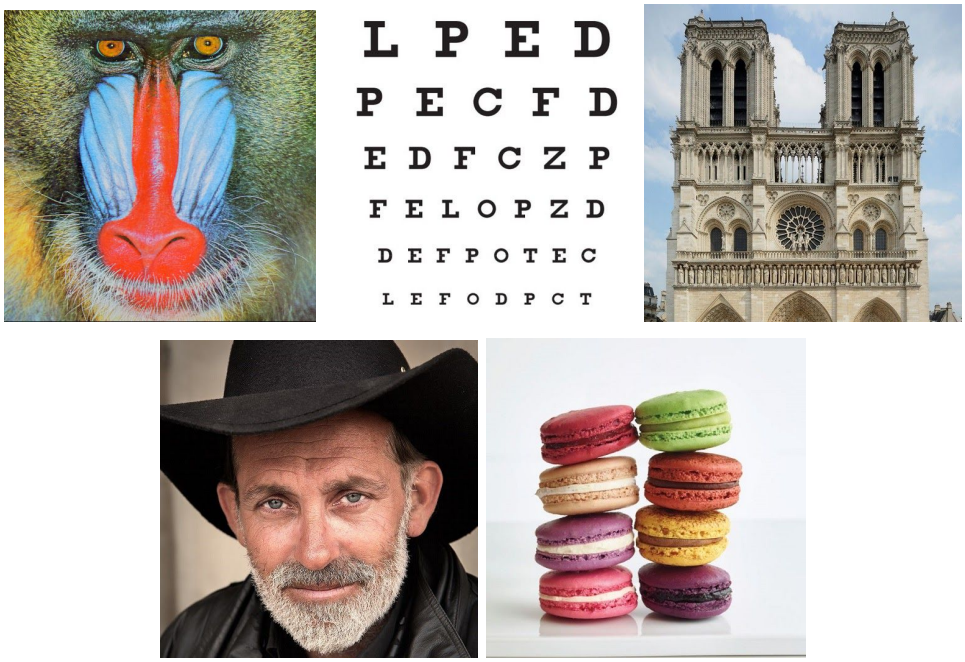


Figure 11: (Clockwise from Top Left) Mandrill, Eye Chart, Notre Dame, Macarons, and Cowboy

Table 1: Prescriptions along with their corresponding near/far point

Nearsightedness	**Far Point (mm)**	**Farsightedness**	**Near Point (mm)**
-2D (Myopia)	500	+1D (Presbyopia)	330
-3D (Myopia)	330	+2D (Presbyopia)	500
-5D (Myopia)	200	+3D (Presbyopia)	1000

4.4 Hardware-Based Tests for Image Quality and Runtime Analysis

We perform camera-based tests following the methods of Fu Chung Huang (2014) in order to capture real images of our corrected images. We implement a light field display using an iPhone 6S Plus layered with a pinhole mask array. We use a NEX-5N camera to focus at specific distances in order to model nearsighted and farsighted vision. A VL53L0X laser range

finder module is used to calibrate the display distance. This system allows us to validate how our algorithms work in the physical world and correspond with what our simulation produces.

Figure 12: The camera is mounted on the tripod, with a laser range sensor mounted on the camera. An iPhone 6S Plus with pinhole mask is located at the bottom.

We evaluate the runtime of each algorithm on a Raspberry Pi 4B with 4GB memory. For each algorithm, we record the average time taken to compute the prefiltered image on a single user case. This runtime analysis serves to demonstrate the relative complexity of each algorithm, as well as provide a sense of scale for how each algorithm will perform on a mobile device, the intended use case for VCDs.

5. Experimental Results: Comparisons and Analysis of Image Quality and Runtime

5.1 Results of Parametrized Many to Many

In order to determine how increasing sampling improves the resulting corrected image quality, we tested two levels of sampling with Parametrized Many to Many: K = 2 with 16 light rays traced per display pixel, and K = 3 with 81 light rays traced per display pixel. In simulation, the Parametrized Many to Many algorithm performs poorly on nearsighted cases. For example, in Figure 13, both corrected results appear blurrier than the uncorrected result, producing no improvement. In farsighted cases, Parametrized Many to Many performs better - Figure 14 shows simulations of a +1D farsighted scene where the correction does produce clearer results than the uncorrected image. Figure 15 shows that a +1D farsighted user would be able to see slightly sharper letters with the correction Parametrized Many to Many provides, for all but the smallest two letter sizes. From our side by side comparisons and nearsighted and farsighted testing, we find that higher sampling rate does not visually improve image quality. However, Parameterized Many to Many with K = 2 runs almost five times faster than Parametrized Many to Many with K = 3, which leads us to conclude that the optimal sampling rate is K = 2.

Simulated Many to Many K=2	Simulated Many to Many K=3	Uncorrected
		

Figure 13: Results for a -2D nearsighted user viewing the display 610 mm away.

Simulated Many to Many K=2	Simulated Many to Many K=3	Uncorrected
		

Figure 14: Results for a +1D farsighted user viewing the display 200 mm away.

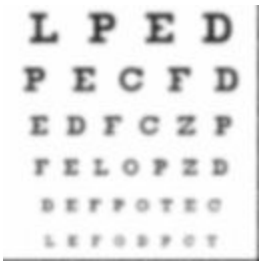
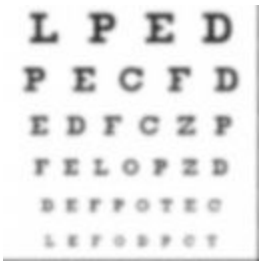
Simulated Many to Many K=2	Simulated Many to Many K=3	Uncorrected
		

Figure 15: Results for a +1D farsighted user viewing the display 225 mm away.

5.2 Results of Forward Optimization

The results for Forward Optimization show that the algorithm sharpens the image for farsighted users, but performs worse in nearsighted use cases similar to Parametrized Many to Many. For example, in Figure 16, the corrected output is much sharper than the uncorrected

output for a simulated user with +2D farsightedness. However, on pure images of text like the eye chart in Figure 17, the smaller letters show aberrations. This disparity is likely in part due to the image content, as small, high contrast letters are more challenging for a display to correct, and in part due to the fact that aberrations are easily noticed in the black and white image. In nearsighted cases, like for the -3D nearsighted user in Figure 18, the corrected results are almost indistinguishable from the uncorrected image. We discuss some of our thoughts on the differences between nearsighted and farsighted correction in Section 5.4.

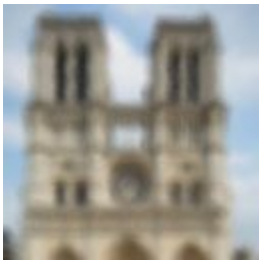
Simulated Forward Optimization Corrected	Uncorrected
	

Figure 16: Results for a +2D farsighted user viewing the display 150 mm away.

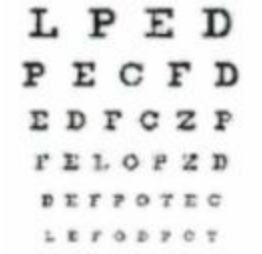
Simulated Forward Optimization Corrected	Uncorrected
	

Figure 17: Results for a +1D farsighted user viewing the display 225 mm away.

Simulated Forward Optimization Corrected	Uncorrected
	

Figure 18: Results for a -3D nearsighted user viewing the display 515 mm away.

5.3 Comparison Between Algorithms

In comparing each algorithm (Figure 19), we see that they all follow the same pattern of performing better on farsightedness than nearsightedness. In both farsightedness and nearsightedness, the Forward Optimization algorithm produces the sharpest image but loses a little quality on the resulting rough edges. Another notable observation is that Parametrized Many to Many produces sharper images compared to Area to Area. Even though both algorithms use similar pixel averaging for the prefiltered image, Parametrized Many to Many accounts for a more refined region of pixels. In the nearsighted case, none of the algorithms demonstrate a noticeable improvement over the uncorrected image, and Point to Point spectacularly fails.

Test Case	Point to Point	Area to Area	Parametrized Many to Many (K=2)	Forward Optimization	Uncorrected
+2D farsighted, display at 350mm					
-2D nearsighted, display at 650 mm					

Figure 19: Comparison between +2D farsighted and -2D nearsighted simulation results of each algorithm, corresponding to a display offset of 150 mm.

In contrast to the visual comparison, Figures 20 and 21 give the metric results of each algorithm for +2D farsightedness and -2D nearsightedness for the Cowboy image. Our other metric data for the remaining images and user cases is provided in Appendix C, along with

some additional side by side image comparisons. Surprisingly, these figures show that Area to Area has the highest PSNR and HDRVDP-V2 quality score, although the metric scores for every algorithm tend to the same value as the distance between the display and focal plane increases. This doesn't seem to correspond with our visual comparison in Figure 19, in which the Forward Optimization results look perceptually sharper than the Area to Area results. We believe that these results are due to the PSNR and HDRVDP-V2 metrics – these metrics are not tailored for detecting image sharpness, and instead only measure the overall difference with the original image, which Area to Area preserves by taking the average of reached pixel values. Forward Optimization also has an issue where some pixel values are corrupted by being clipped to the maximum value.

Quantitatively the PSNR and HDRVDP-V2 quality measures of the corrected images are almost always worse than the uncorrected image, i.e. the correction performs worse than doing nothing. While visual comparison of the Point to Point and Area to Area results make it difficult to notice an improvement, Forward Optimization does have cases with image quality improvements, such as in Figure 19. Due to our reliance on our simulation in order to generate these metrics, and because these metrics do not fully measure differences in image sharpness, we cannot draw definitive conclusions about the quality distinctions between the Area to Area, Parametrized Many to Many, and Forward Optimization algorithms. However, Point to Point consistently has the worst image quality metrics, and with side by side comparison showing Point to Point does little to improve image quality, we can make the conclusion that the one light ray sample Point to Point uses for each pixel is not enough information to correct a user's vision.

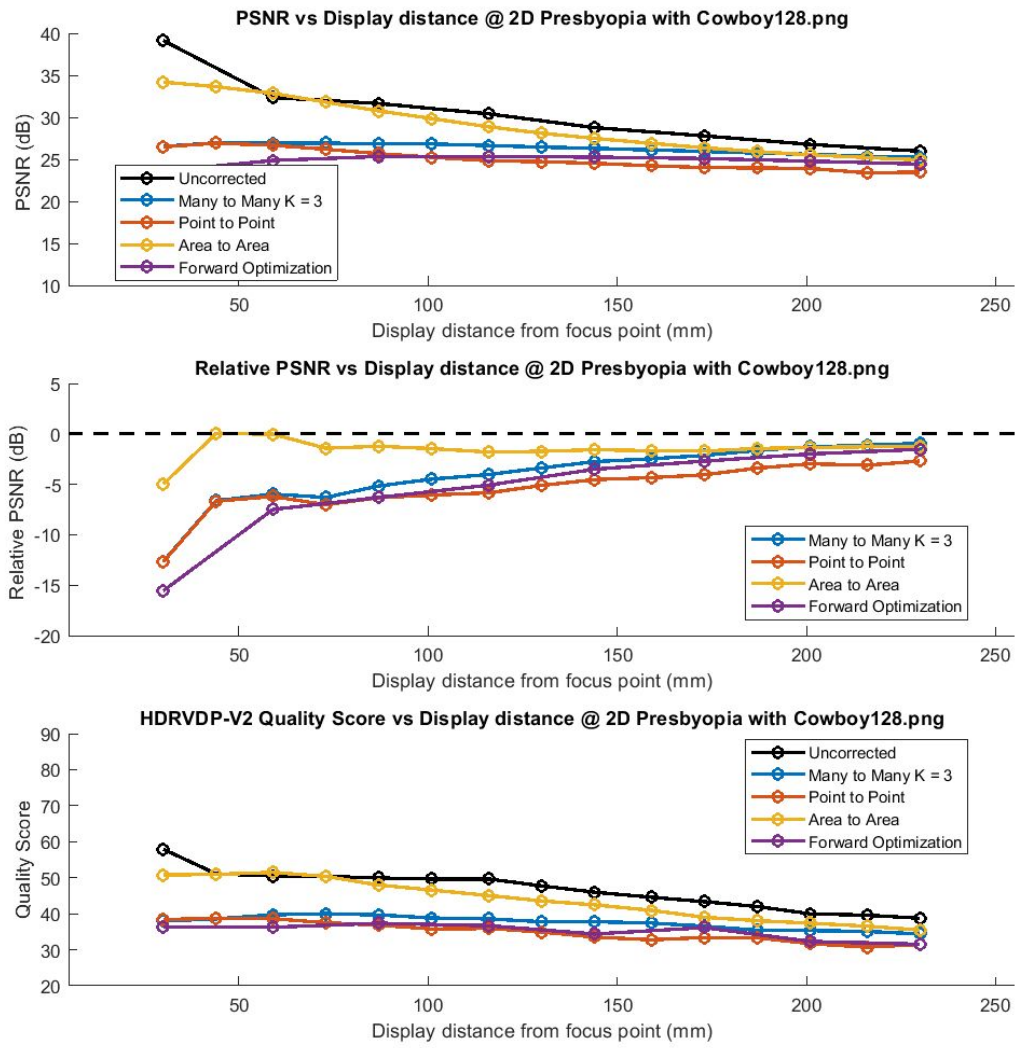


Figure 20: Metrics results for +2D Presbyopia cases on the Cowboy image.

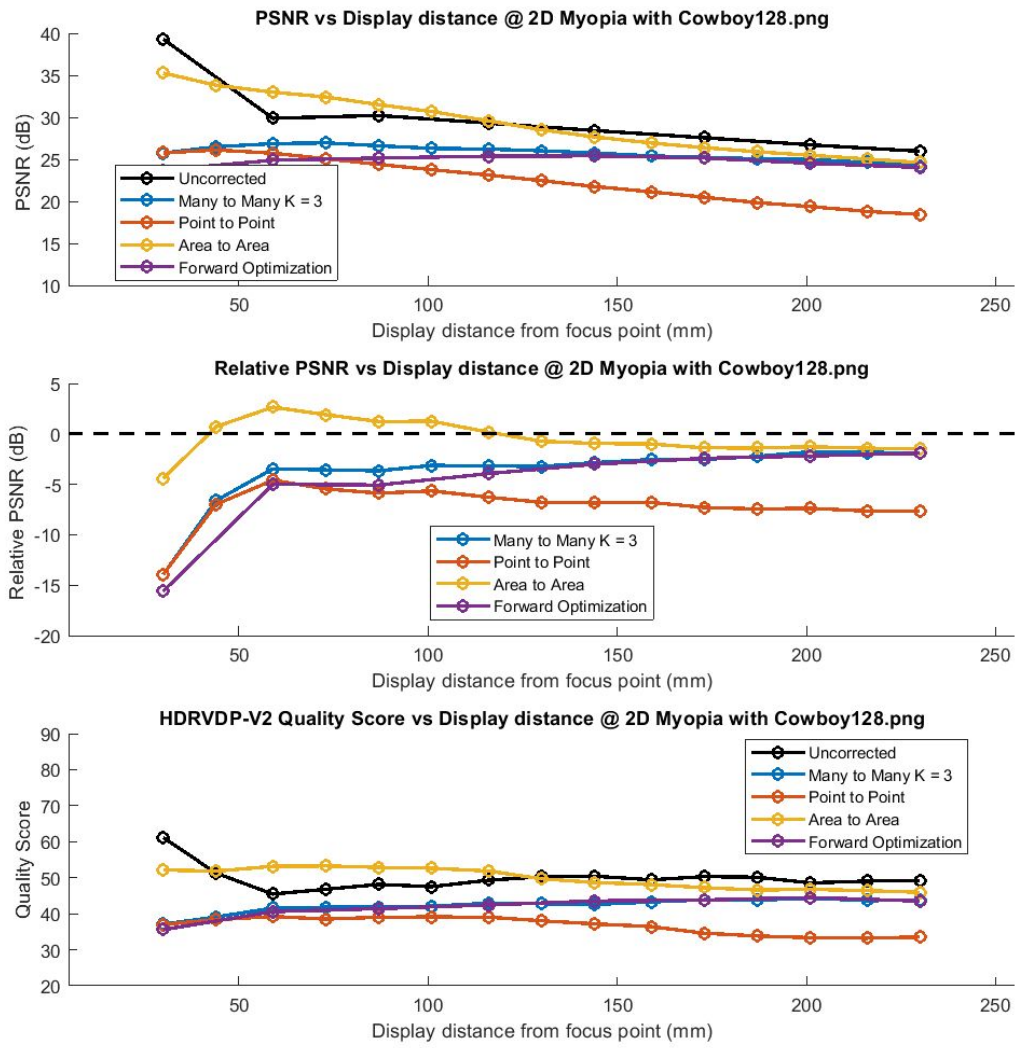


Figure 21: Metrics results for -2D Myopia cases on the Cowboy image.

5.4 Addressing Differences in Nearsighted and Farsighted Results

One question that repeatedly came up during our analysis was what accounted for the difference in performance between nearsighted and farsighted cases for each of our algorithms. Most obvious is how Point to Point fails in the nearsighted case, but Forward Optimization and

Parametrized Many to Many also seem to correct images less for nearsighted users. This difference is surprising, due the fact that nearsightedness and farsightedness are both caused by the eye focusing light at a distance offset from the retina. Figure 22 shows the wavefront maps for nearsighted and farsighted vision, which is the formal way of characterizing these aberrations. In the field of optics, these two wavefront maps both have the same mathematical representation using what are called the Zernike polynomials, which serve as a basis for any wavefront map. In terms of the six “low order” Zernike polynomials, shown visually in Figure 23, nearsighted and farsighted vision are positive and negative amplitudes of defocus respectively. Because the wavefront maps of nearsighted and farsighted vision are so similar, one would expect the performance of these algorithms to be more similar across both kinds of aberration. We believe that the disparity in results is not due to the algorithm performance, but instead that the light-field-based Vision Correcting Display represented in testing is naturally better suited to correcting farsighted vision than nearsighted vision.

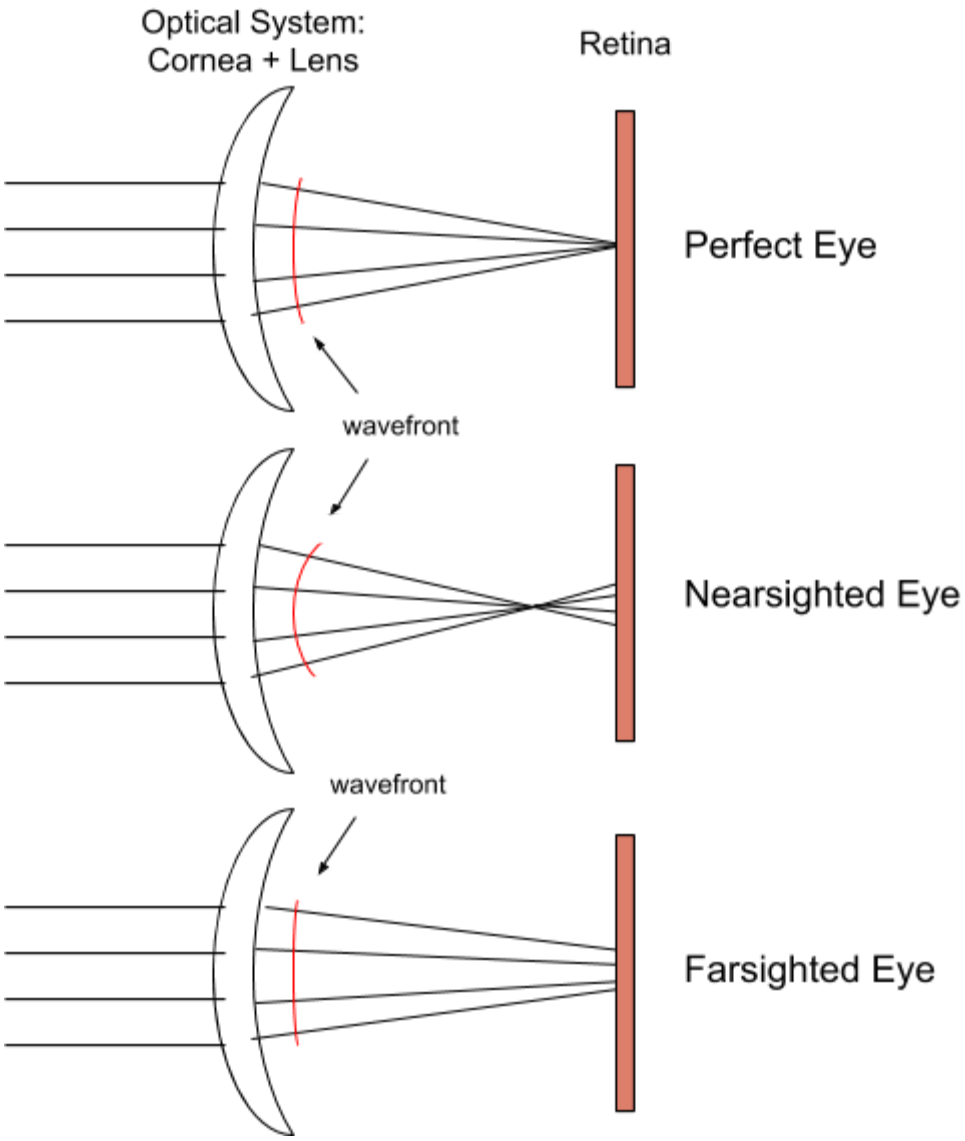


Figure 22: Wavefront maps for a perfectly focused eye, a nearsighted eye and a farsighted eye. The curved wavefront in red characterizes the refraction the light undergoes due to the optics of the eye.

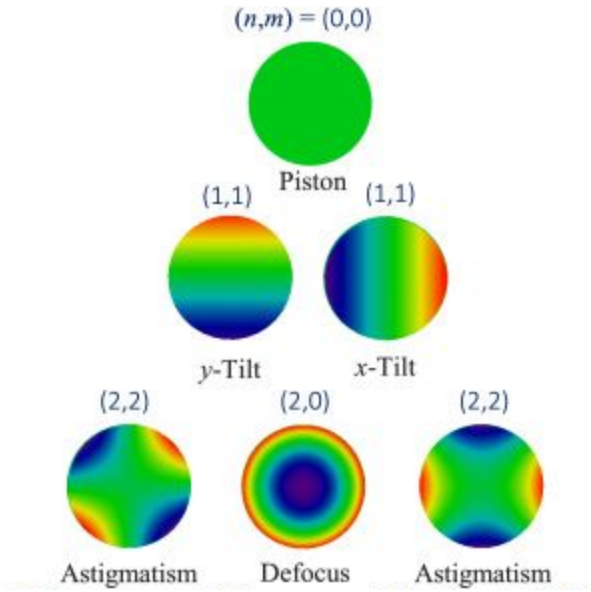


Figure 23: These images show the six low order Zernike polynomials which are used to describe the shape of the wavefront map over the circular pupil for low order aberrations (Hsieh, Y. H. et al. 2020).

At a high level, our vision correcting algorithms compute a prefiltered image that the light field projects to a plane other than the display, so that the user can see the projected image in focus. The exact prefiltered image is computed based on the visible pixels in the display and the user's prescription, but the end goal is for the light field to reconstruct the desired image at a distance where it is in focus to the user. For farsighted users, this distance is behind the display, and for nearsighted users, this distance is in front of the display, as illustrated in Figure 24.

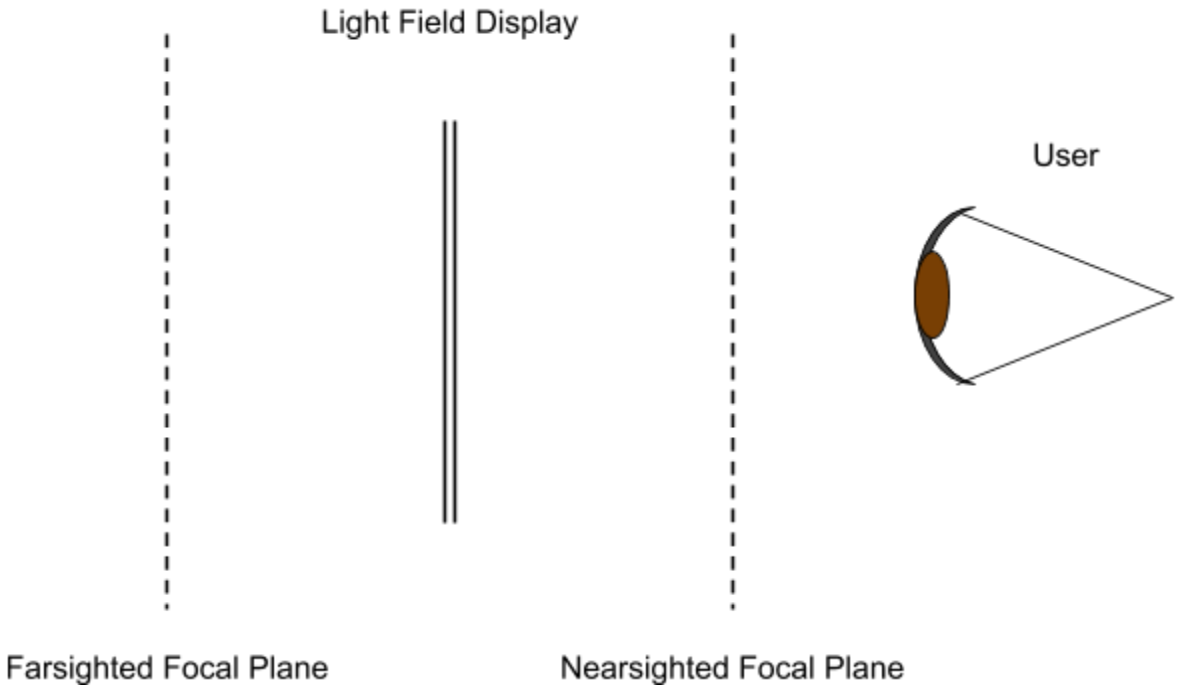


Figure 24: In the farsighted case, the user can see images clearly at the farsighted focal plane behind the light field display. For the nearsighted case the user can only see images clearly in front of the display at the nearsighted focal plane.

When the focal plane is behind the display, the virtual image at the focal plane is constructed at a size larger than the display in order to form an image of the same size on the user's retina, and when the focal plane is in front of the display, the virtual constructed image is smaller than the display, as illustrated in Figure 25. As a result, in the nearsighted case, many light rays projected from the light field display intersect the nearsighted focal plane outside the boundary of the rescaled image size, and therefore cannot contribute to the prefiltered image. Further, because the scale of the image is smaller, we believe that the light field display cannot construct as clear an image at that relative size. We think this problem could be addressed by rescaling the virtual constructed image in the nearsighted case to make a larger image on the retina – however, this has the problem where light in the environment from behind the display will reduce the quality of the edges of the image. A larger virtual image would allow for more

light rays from the light field display to be used to reconstruct the image, and be able to better reproduce the spatial frequencies of the image.

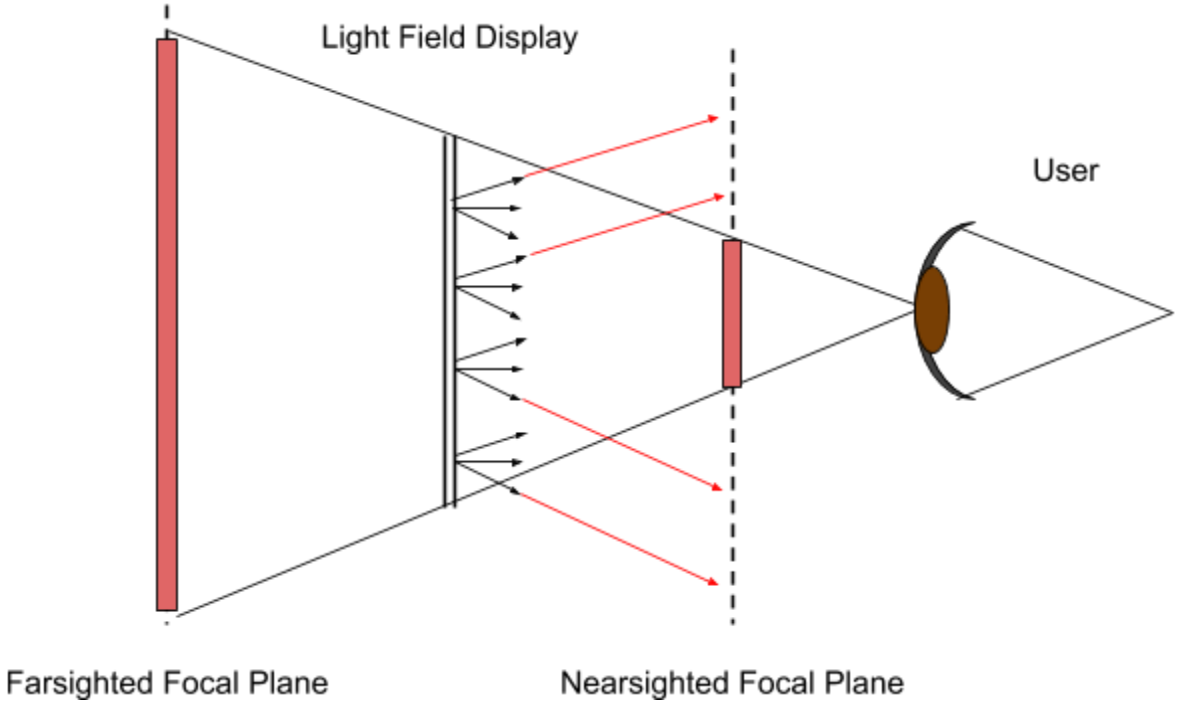


Figure 25: The size of the light field display determines the virtual constructed size of the projected image on the user's focal plane. In the farsighted case the desired image size is large, and results in many light rays from the light field display contributing to the retina image. In the nearsighted case the desired image is small, causing some light rays (shown in red) to fall outside the boundary of the desired image.

5.5 Hardware Based Results

Finally, we evaluate our algorithms using the physical methods described in Section 2 Part 4.4. Figure 26 presents the camera results for each algorithm, from which the most obvious difference among the algorithms is the relative brightness. This comes from the different ways we treat the aperture during the prefiltering process. As part of our observations of the Point to Point and Area to Area algorithms, we found that the naively sampled light rays are often blocked by the pupil aperture, meaning that omitting those light rays would result in a mostly

black prefiltered image. To compensate for this, our implementations of Point to Point and Area to Area ignore the aperture and illuminate the full display. In contrast, the Parametrized Many to Many and Forward Optimization algorithms sample many different light rays for each pixel, so we can account for the aperture without the entire image becoming black. This results in a prefiltered image that has a lower brightness due to light rays being blocked by the aperture. Overall, however, the brightness is most affected by the pinhole mask array light field display that blocks a significant percentage of the light from the display, and can be improved with the usage of other light field displays such as microlens array based light fields.

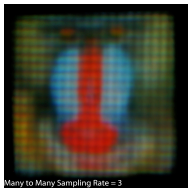
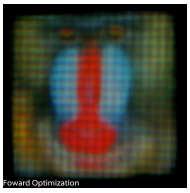
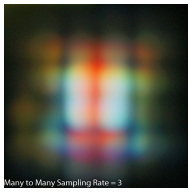
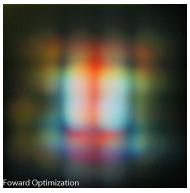
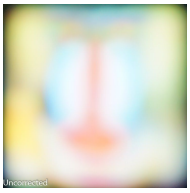
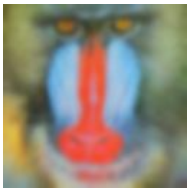
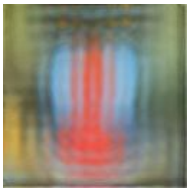
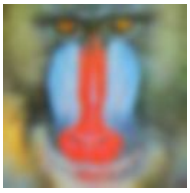
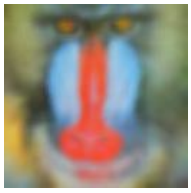
Test Case	Point to Point	Area to Area	Parametrized Many to Many (K=3)	Forward Optimization	Uncorrected
+2D farsighted Display at 200mm					
-2D nearsighted Display at 800 mm					

Figure 26: Comparison between camera photos on prefiltered results of each algorithm.

In regards to image quality, Parametrized Many to Many and Forward Optimization visually stand out as the best, followed by Area to Area with a slight improvement achieved in Point to Point. These images differ from our simulation results in Figure 27 by showing more blurriness overall, but the relative image quality among algorithms are consistent with simulation results. We believe both physical testing and the simulation software need to be further

calibrated to produce more faithful renderings of human vision. Note that this calibration will impact image quality measurements and future research may need to revisit these to establish expectations for needed quality.

Test Case	Point to Point	Area to Area	Parametrized Many to Many (K=3)	Forward Optimization	Uncorrected
+2D farsighted Display at 200mm					
-2D nearsighted Display at 800 mm					
Figure 27: Comparison between simulation results of each algorithm.					

Finally, we benchmark these algorithms based on average run times on a Raspberry Pi Model 4B. The results are as we expect, where the Point to Point algorithm takes the minimum amount of computation. Area to Area and Parametrized Many to Many (K = 2) have similar runtimes, but increasing the sampling rate for Many to Many has a quartic effect on runtime. Forward Optimization takes the longest due to its large matrix calculations and optimization process. Every algorithm besides Forward Optimization operates on each pixel independently, which means their runtimes can be improved using parallelization.

Table 2: Average algorithm runtime on a Raspberry Pi 4B over 10 trials	
Algorithm	**Average Runtime (ms)**
Point to Point	1293.11
Area to Area	9416.89
Parametrized Many to Many (K = 2)	10976.89
Parametrized Many to Many (K = 3)	50833.33
Forward Optimization (K = 3)	2187366.89

6. Future work: Directions for Future Projects

In developing and testing our library and prefiltering algorithms, we found many possible improvements and avenues for future research. Most promising is the idea of simplifying the Forward Optimization algorithm for faster runtimes. This algorithm provided the sharpest corrected images, and should continue to be closely studied. Second, there are a number of natural continuations for our software library, including extending the algorithms to work with higher order aberrations and different display types. Finally, we suggest a ground-up redesign on software simulation of human vision for the purposes of Vision Correcting Display development, in order to address difficulties we had while testing and make it possible to continue to support future Vision Correcting Displays research.

6.1 Forward Optimization Prefiltering Continuations

Our side by side comparisons results shows that the Forward Optimization algorithm performs the best in terms of qualitative image correction. In terms of runtime, however, it is clearly the worst. In order to provide users with real time noticeable image correction, future Prefiltering algorithm research should continue to focus on this optimization-based approach with the express goal to make it faster. This is critical in order to correct higher resolution images in a reasonable amount of time. While hardware approaches are one way to improve runtime, another possible direction is to selectively prefilter regions of the image. Generating high quality images may still be possible by using faster prefiltering algorithms like Area to Area on low frequency regions of the image. The biggest issue that the Forward Optimization algorithm encounters is the size of matrices involved in optimization. Since the matrix size scales with the display and desired image resolutions, finding a way to prefilter disjoint regions

of the display image separately and reduce the sizes of the matrices is necessary to keep the algorithm efficient for larger images.

6.2 Prefiltering Library Research Extensions

In developing our software library, we exclusively focus on correcting low order aberrations with pinhole mask arrays. As such, there is an opportunity to extend our work to include more advanced display types and address higher order aberrations. Incorporating new display types, such as microlens arrays, is important to support future research, since pinhole mask arrays have the issue of blocking too much light. This would require modifying the Convert step to account for the display in a modular way, allowing light ray samples to be determined based on the display type. This would also require updating the simulator to use a different OpenCL kernel to simulate.

Extending these algorithms to higher order aberrations will also be important if there is to be an efficient algorithm to correct any vision problem. These aberrations will affect the way light enters the pupil, and requires modifying each algorithm's Project step to use the wavefront map of the eye defined in terms of the Zernike polynomials. Unlike nearsightedness and farsightedness, glasses/contacts cannot correct these aberrations, but Vision Correcting Displays theoretically should be able to.

We also recommend continuing to investigate the differences between nearsightedness and farsightedness. As described in section 5.4, even though the Zernike polynomial representations of nearsightedness and farsightedness are the same, nearsighted cases performed visually worse in our tests. As described in that section, one possible approach is to experiment with the size of the virtual image being projected on the user's focal plane. Additionally, different display types may have better results for nearsighted cases than the

pinhole mask light field we tested. For instance, the increased light projected by a microlens array light field may lead to better image quality.

6.3 Vision Simulation Software Redesign

Our work uses simulation extensively to develop and test our vision correcting algorithms, but over the course of our research we encountered a number of problems that could be addressed by future development. First and foremost, the results of our simulation do not exactly match our physical test results or expectations for human perception. Second, while our configuration process is adequate enough for us to run our tests, our simulation software still lacks several useful features. Third, the software design does not lend itself well to future studies of higher order vision problems or alternative display types, such as microlens arrays. Developing a more robust and meaningful simulation software will be extremely beneficial to future research in Vision Correcting Displays and can be the focus of one or more research projects. In this section, we explain in more detail some specific goals an ideal simulation tool should accomplish to address the above problems, and we provide suggestions for future teams on how to approach these tasks.

Simulating a scene falls squarely in the domain of computer graphics, but the purpose of this tool is to simulate human vision. As such, it is necessary to understand how the human visual system works. Light and optics also play a fundamental role in simulating the properties of the light field display. Our simulator uses a camera model to render images, but other models for vision may provide better results. In particular, we found it difficult to assign parameters to the camera model in order to match human vision, since the camera model assumes that we see in the same way a camera sees. By beginning with basic models like the camera model and addressing previous assumptions, it will be possible to learn the impact these assumptions have

on the final visual render. Possible research directions to pursue for more complicated rendering models include Professor Barsky's work in Vision Realistic Rendering (Barsky, Brian A. 2004) and optometry eye models.

There are a number of useful software features that would help with future testing. For our research, we use configuration files specifying user and display parameters for both prefiltering and simulation. This provides the benefit of having a single place to configure settings between both programs, but requires manually writing the configuration file and doesn't lend itself to explaining what all the settings do. Creating a user interface to generate configuration files for the simulation and prefilter process will help make these settings clearer and guide users how to run tests. Another missing feature of our simulation is simplified viewing of the final results. Our simulation program runs entirely on the command line and does not provide any method to visualize or compare images - in order to compare the images users must open the simulated images manually. Adding options to run with a graphical interface and display the simulated images will make the simulator easier to use, especially for new or inexperienced researchers, and aid researchers in testing new vision correcting ideas.

Finally, it is important that the simulation design should be modular and extendable in order to support future VCD research, as future research may be based on new types of displays, eye models, or any kind of new ideas. As such, it is important to clearly define and separate distinct concepts in the simulation process. In particular, the simulator must be designed to accommodate novel displays and eye models. While we only support pinhole mask array displays in our existing simulation software, there is code for other display types present in our simulation codebase developed by previous researchers, including Zehao Wu (2016). This code can be used as a reference, but does not provide an example of good extendable software.

7. Conclusion: Summary of our Results

In this report, we quantitatively and qualitatively showed how our Parameterized Many to Many and Forward Optimization vision correcting algorithms perform in comparison to previously established algorithms for nearsighted and farsighted vision. In order to complete this analysis, we developed a testing process built upon the simulator introduced by Zehao Wu in 2016, implemented image quality metrics in Matlab, and ran physical camera-based testing. This undertaking required organization and planning in order to develop software that is maintainable and useful for future research.

Our visual comparison showed that Forward Optimization provides the sharpest images, but had extremely high run times to produce the prefiltered images. Point to Point performed the worst, having the worst metrics and visually providing the least correction, and indicates that some amount of sampling is necessary beyond a single light ray for each pixel, since the averaging techniques employed by Area to Area and Parametrized Many to Many do improve on the results of Point to Point. Area to Area had the highest PSNR and HDRVDP-V2 quality metrics on the display distances we considered, higher than Forward Optimization. We think that this is due to pixel noise due to clipping in the Forward Optimization output, as well as the PSNR and HDRVDP-V2 quality metrics not measuring image sharpness. The metric values all followed similar trends and approached the same scores as the display distance increased, but never significantly improved beyond the uncorrected image quality metrics. Even so, we believe that Forward Optimization is the most promising algorithm due to the visual sharpness it provides, and urge future researchers to focus on this approach.

We believe that this is the first time that comprehensive simulation-based tests measuring image quality and runtime have been done on the Point to Point and Area to Area

algorithms. As our simulation results did not precisely correspond to physical experiments, we encourage further development to improve on simulation tools, in order to make robust and verifiable software that produces useful image quality metrics. Still, our efforts provide a set of procedures and strong directions for future research in Vision Correcting Displays. We think with further improvements to our simulation and Prefiltering library, progress can be made towards developing efficient Vision Correcting Display algorithms. As a result, users will be able to clearly see their screen and improve reading comfort, even if they can’t reach their glasses.

Appendix

Appendix A: Existing Codebase and Quick Start Guide

- The prefiltering repository is here: https://github.com/vision-correcting-display/prefiltering
- The simulation and testing repository is here: https://github.com/vision-correcting-display/simulation

Note that repositories and document links are private, so you will have to request permission from Professor Barsky's research group in order to access them. Information on how to build and run the executable is located within the Github wikis of the repositories:

- The prefiltering wiki is here: https://github.com/vision-correcting-display/prefiltering/wiki
- The simulation wiki is here: https://github.com/vision-correcting-display/simulation/wiki

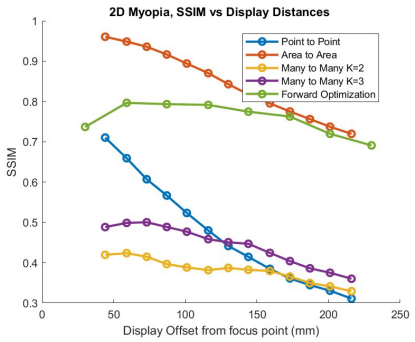
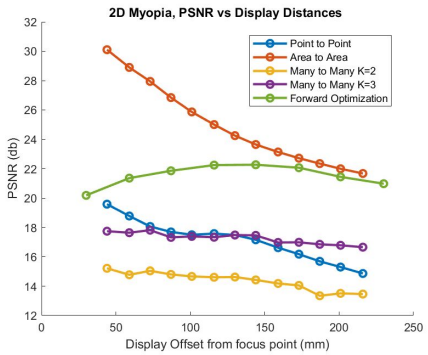
In addition, some helpful past research papers for understanding the basics of vision correcting displays and documents have been listed below.

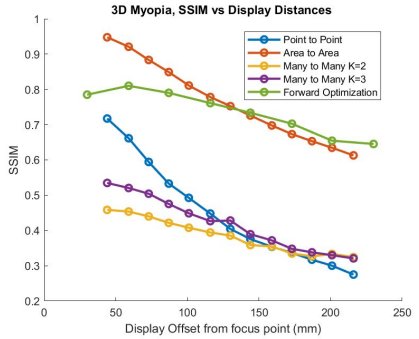
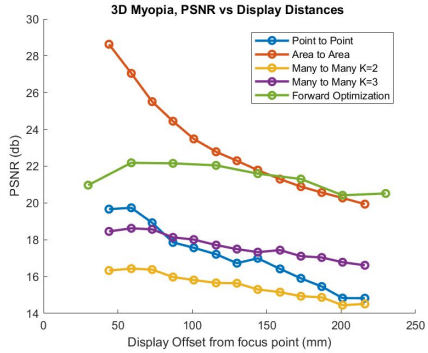
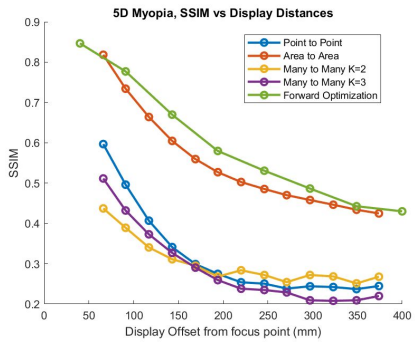
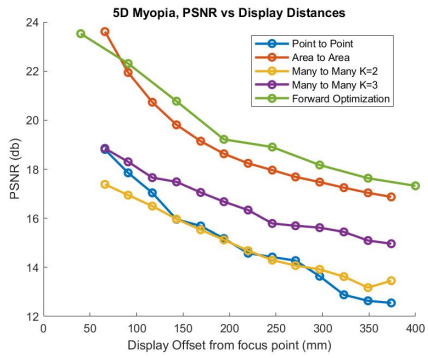
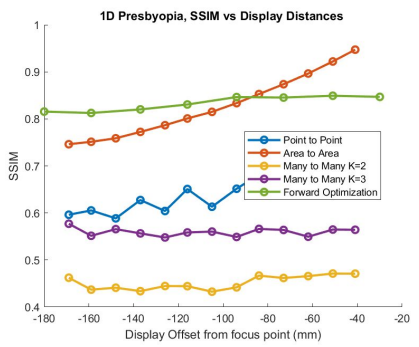
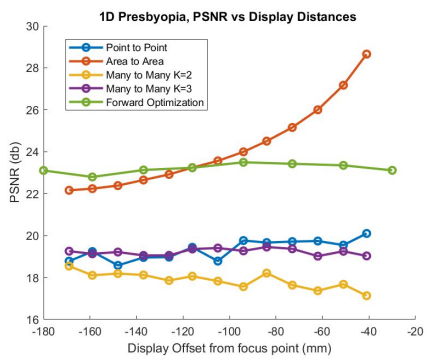
- Charles Ding "Algorithms and Applications of the Vision Correcting Display": https://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-103.pdf
- Yirong Zhen "New Algorithms for Vision Correcting Display": https://drive.google.com/drive/u/0/folders/1XDYZVP5DMNaTJODboXrCw_vsZ4psABAf
- Fu-Chung Huang "Eye-glasses Free Display: Towards Correcting Visual Aberrations with Computational Light Field Displays": http://graphics.berkeley.edu/papers/Huang-EFD-2014-08/Huang-EFD-2014-08.pdf
- Simulation Tools Guide: https://docs.google.com/document/d/1MJxNCocRkmcT74xYmFIe_4QEHp7hk-oKWJ-4gX3NYYQ/edit

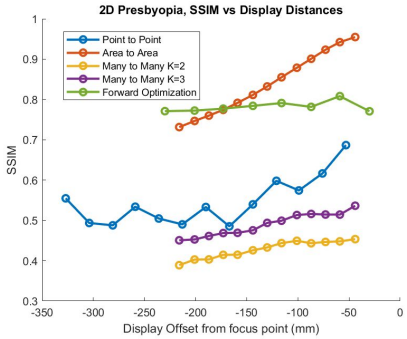
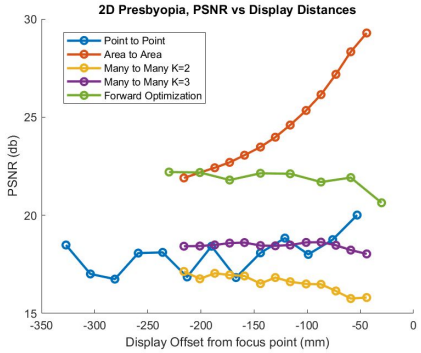
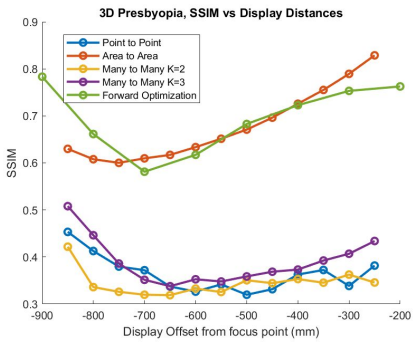
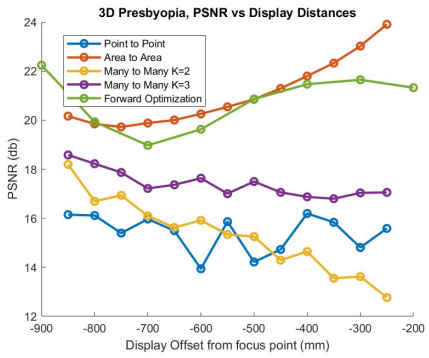
Appendix B: SSIM as an image metric

Structural Similarity (SSIM) is another image quality metric that seeks to extend upon PSNR and factors luminance, contrast, and structure into its calculation. SSIM is also widely used as a metric and is considered by its authors as a better indicator of human opinion than PSNR. This is because PSNR is relatively simple and is computed from pixel-wise differences while SSIM analyzes images as a whole.

In the figure below, the results for SSIM and PSNR show that they roughly follow the same patterns. SSIM is shown to have a preference for Area to Area and Point to Point over Many to Many. While SSIM is a useful metric, we felt that PSNR and HDRVDP-V2 covered both ends of the spectrum (simple/mathematical vs. good predictor of human perception) well enough that SSIM doesn't provide much new insight for the purposes of our analysis.

Prescription	SSIM	PSNR
2D Myopia		

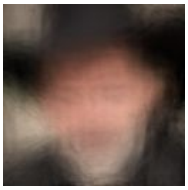
3D Myopia		
5D Myopia		
1D Presbyopia		

2D Presbyopia		
3D Presbyopia		

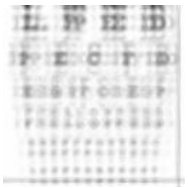
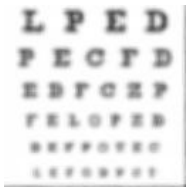
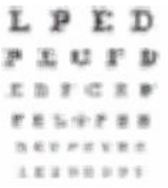
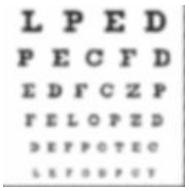
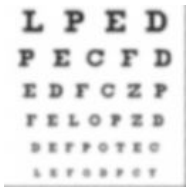
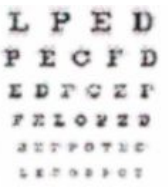
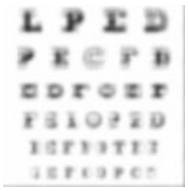
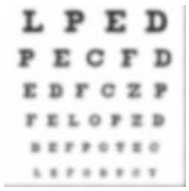
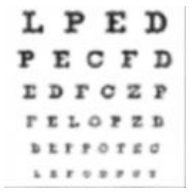
Appendix C: Additional Data and Plots

All final data can also be found in drive here:
https://drive.google.com/drive/u/0/folders/1NYP8wNRI1RBmayCi6jHTSV4lp2JWUVfN

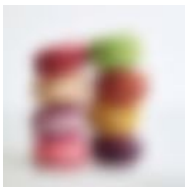
Simulation Results of Cowboy

Test Case	Point to Point	Area to Area	Parametrized Many to Many (K=2)	Forward Optimization	Uncorrected
5D Myopia Display at 446mm					
3D Myopia Display at 560mm					
2D Myopia Display at 730mm					
1D Presbyopia Display at 150mm					
2D Presbyopia Display at 270mm					
3D Presbyopia Display at 750mm					

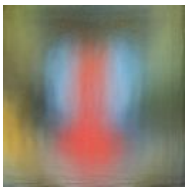
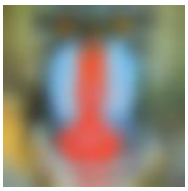
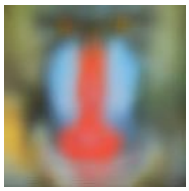
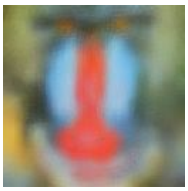
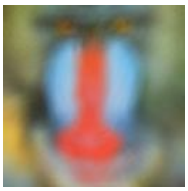
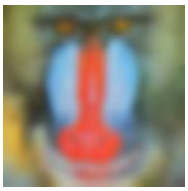
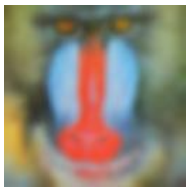
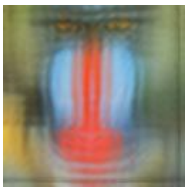
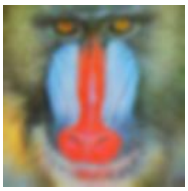
Simulation Results of Letters

Test Case	Point to Point	Area to Area	Parametrized Many to Many (K=2)	Forward Optimization	Uncorrected
5D Myopia Display at 446mm					
3D Myopia Display at 560mm					
2D Myopia Display at 730mm					
1D Presbyopia Display at 150mm					
2D Presbyopia Display at 270mm					
3D Presbyopia Display at 750mm					

Simulation Results of Macarons

Test Case	Point to Point	Area to Area	Parametrized Many to Many (K=2)	Forward Optimization	Uncorrected
5D Myopia Display at 446mm					
3D Myopia Display at 560mm					
2D Myopia Display at 730mm					
1D Presbyopia Display at 150mm					
2D Presbyopia Display at 270mm					
3D Presbyopia Display at 750mm					

Simulation Results of Mandrill

Test Case	Point to Point	Area to Area	Parametrized Many to Many (K=2)	Forward Optimization	Uncorrected
5D Myopia Display at 446mm					
3D Myopia Display at 560mm					
2D Myopia Display at 730mm					
1D Presbyopia Display at 150mm					
2D Presbyopia Display at 270mm					
3D Presbyopia Display at 750mm					

Simulation Results of NotreDame

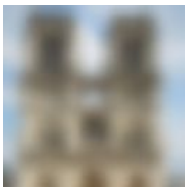
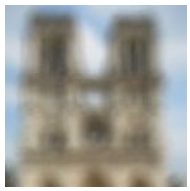
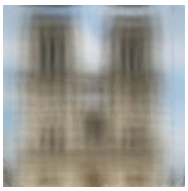
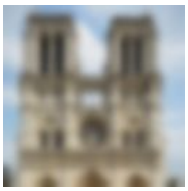
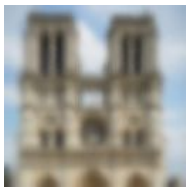
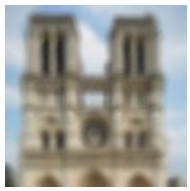
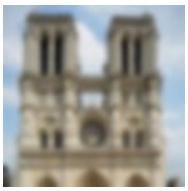
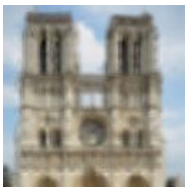
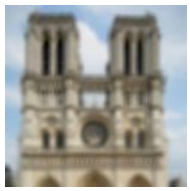
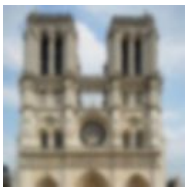
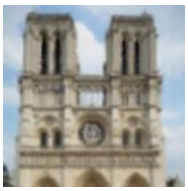
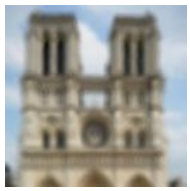
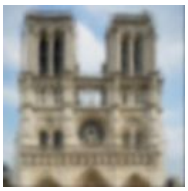
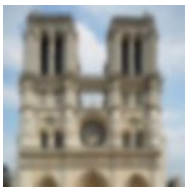
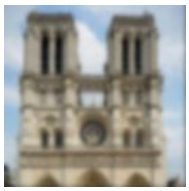
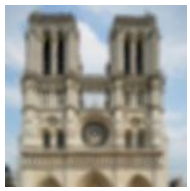
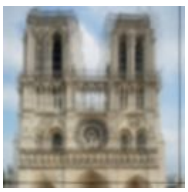
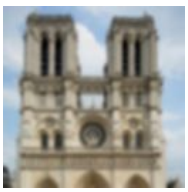
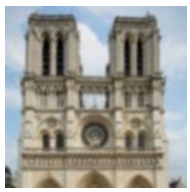
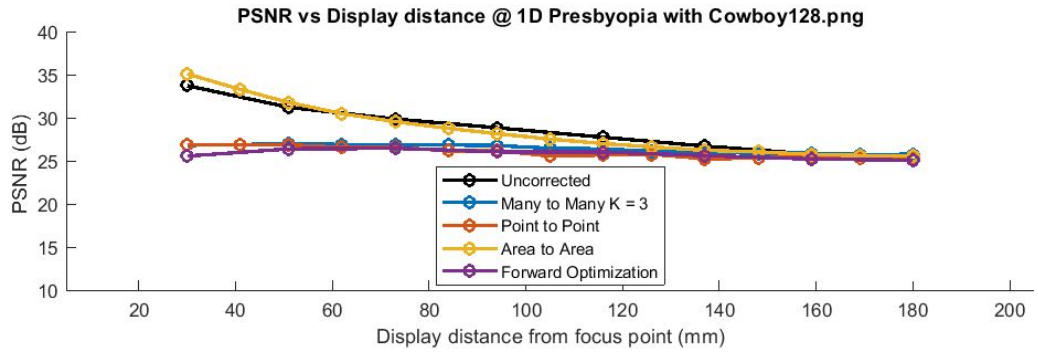
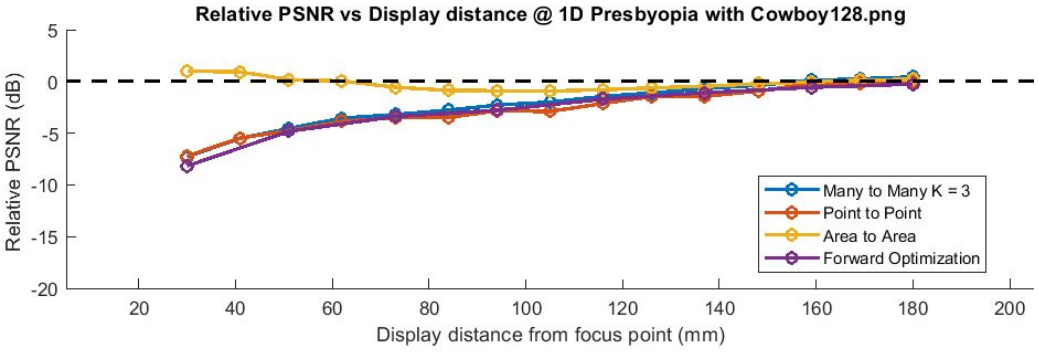
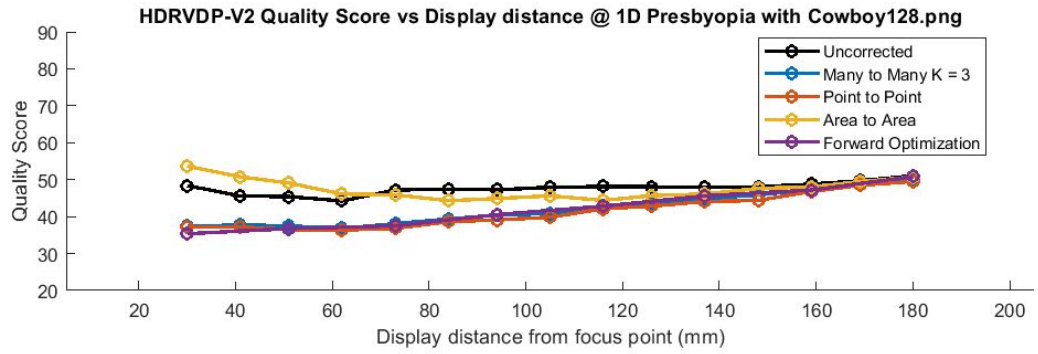
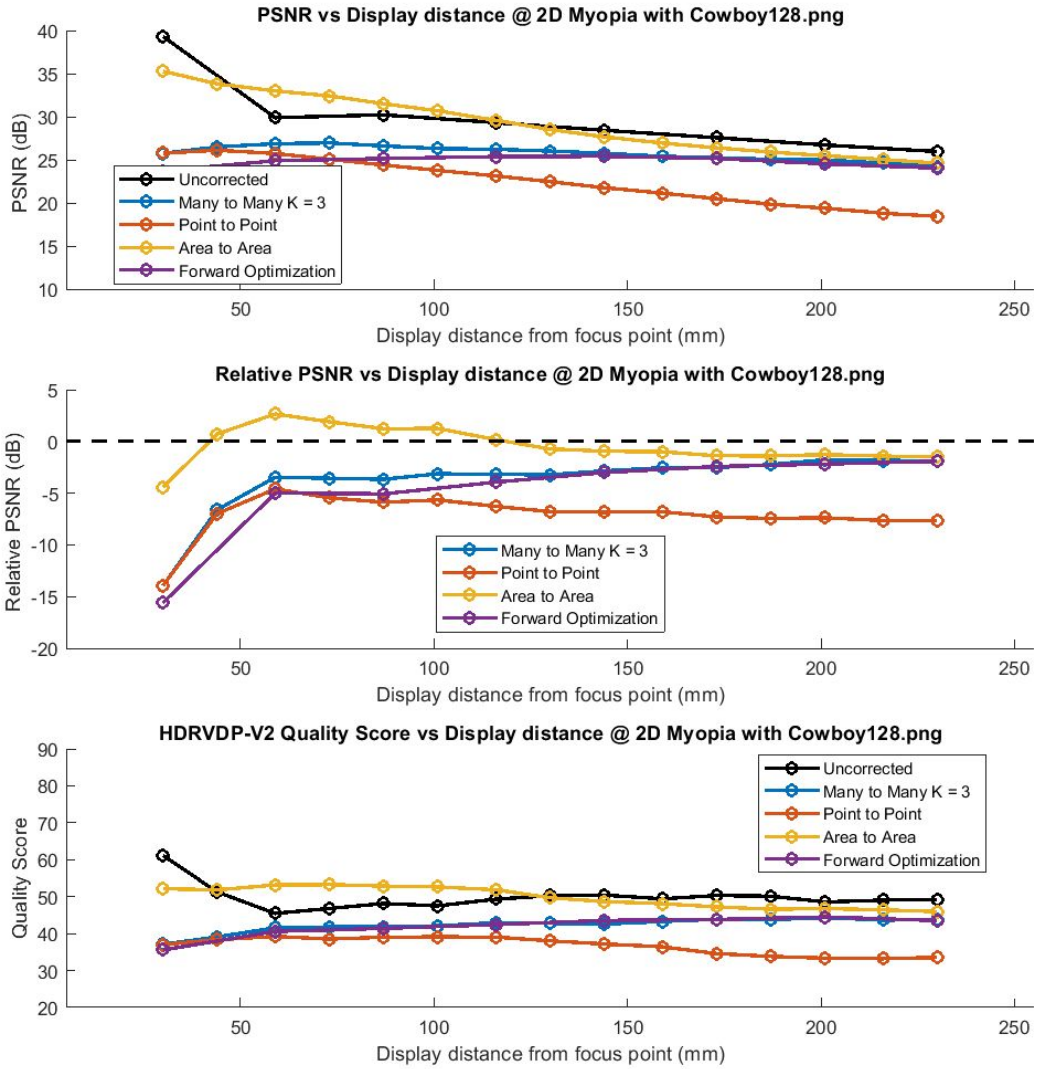
Test Case	Point to Point	Area to Area	Parametrized Many to Many (K=2)	Forward Optimization	Uncorrected
5D Myopia Display at 446mm					
3D Myopia Display at 560mm					
2D Myopia Display at 730mm					
1D Presbyopia Display at 150mm					
2D Presbyopia Display at 270mm					
3D Presbyopia Display at 750mm					

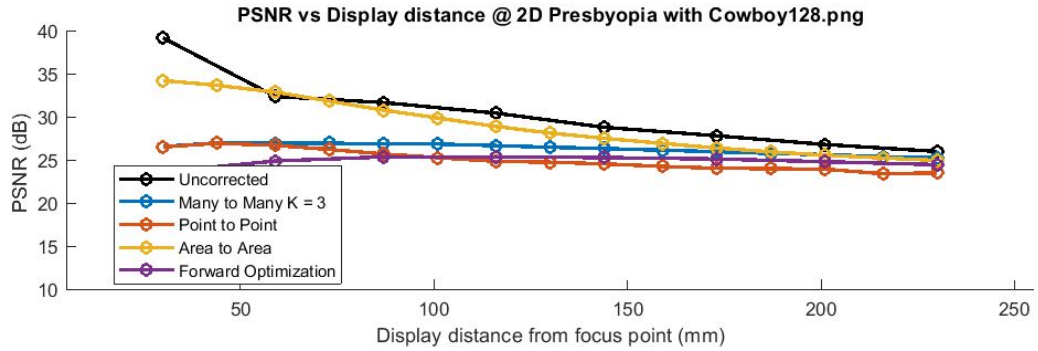
Image Quality Metrics of Cowboy

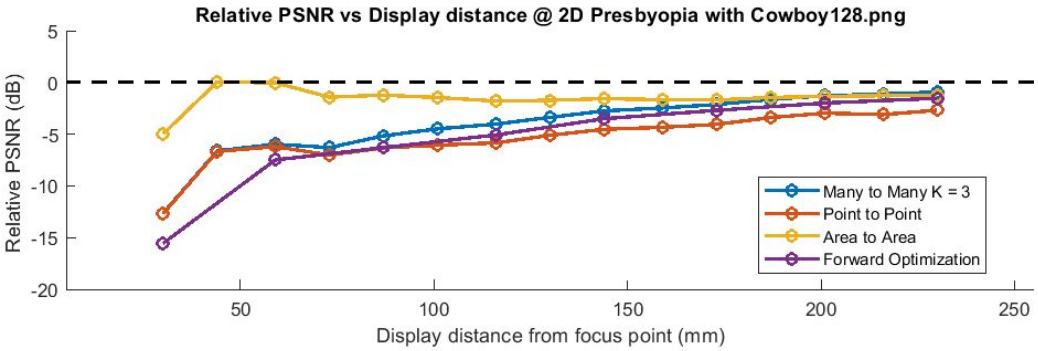


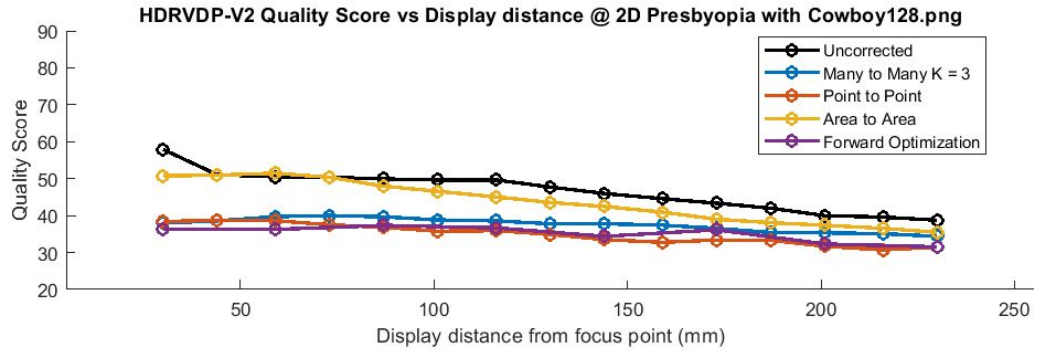


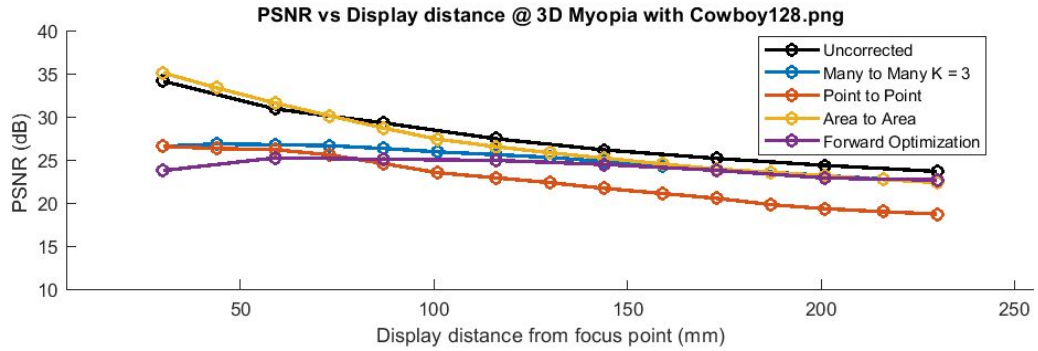


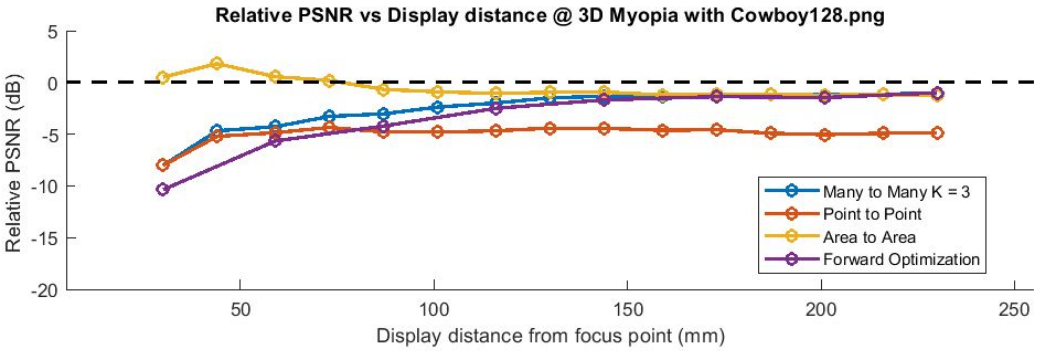

PSNR vs Display distance @ 2D Myopia with Cowboy128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization
Relative PSNR vs Display distance @ 2D Myopia with Cowboy128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization
HDRVDP-V2 Quality Score vs Display distance @ 2D Myopia with Cowboy128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

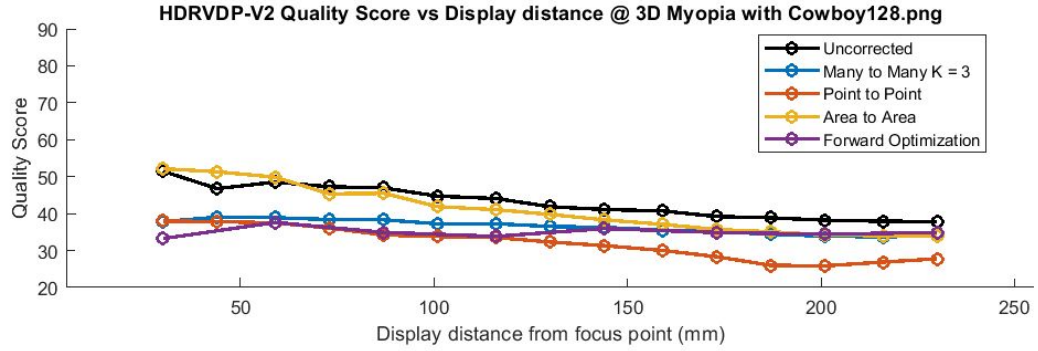

PSNR vs Display distance @ 2D Presbyopia with Cowboy128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

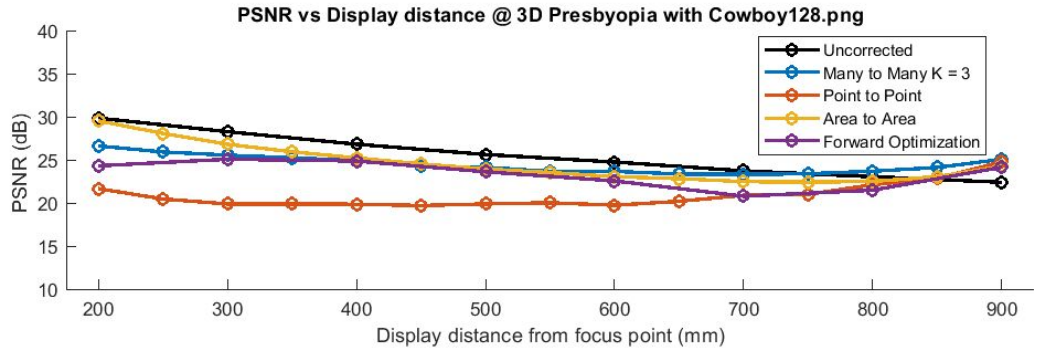

Relative PSNR vs Display distance @ 2D Presbyopia with Cowboy128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

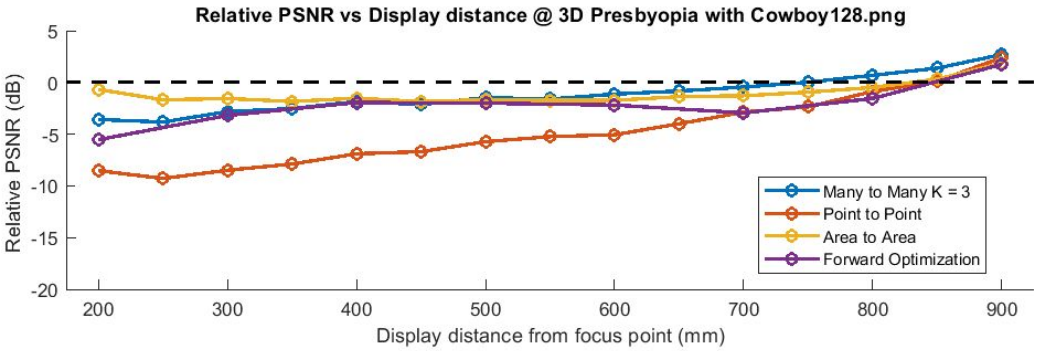

HDRVDP-V2 Quality Score vs Display distance @ 2D Presbyopia with Cowboy128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

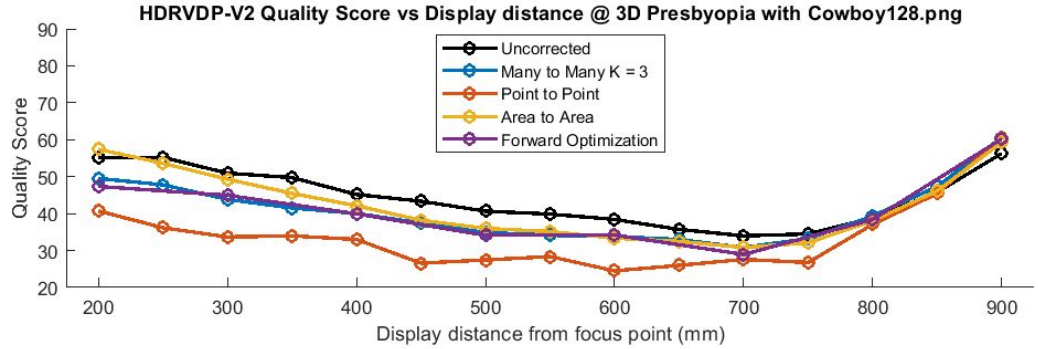

PSNR vs Display distance @ 3D Myopia with Cowboy128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


Relative PSNR vs Display distance @ 3D Myopia with Cowboy128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


HDRVDP-V2 Quality Score vs Display distance @ 3D Myopia with Cowboy128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


PSNR vs Display distance @ 3D Presbyopia with Cowboy128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


Relative PSNR vs Display distance @ 3D Presbyopia with Cowboy128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


HDRVDP-V2 Quality Score vs Display distance @ 3D Presbyopia with Cowboy128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

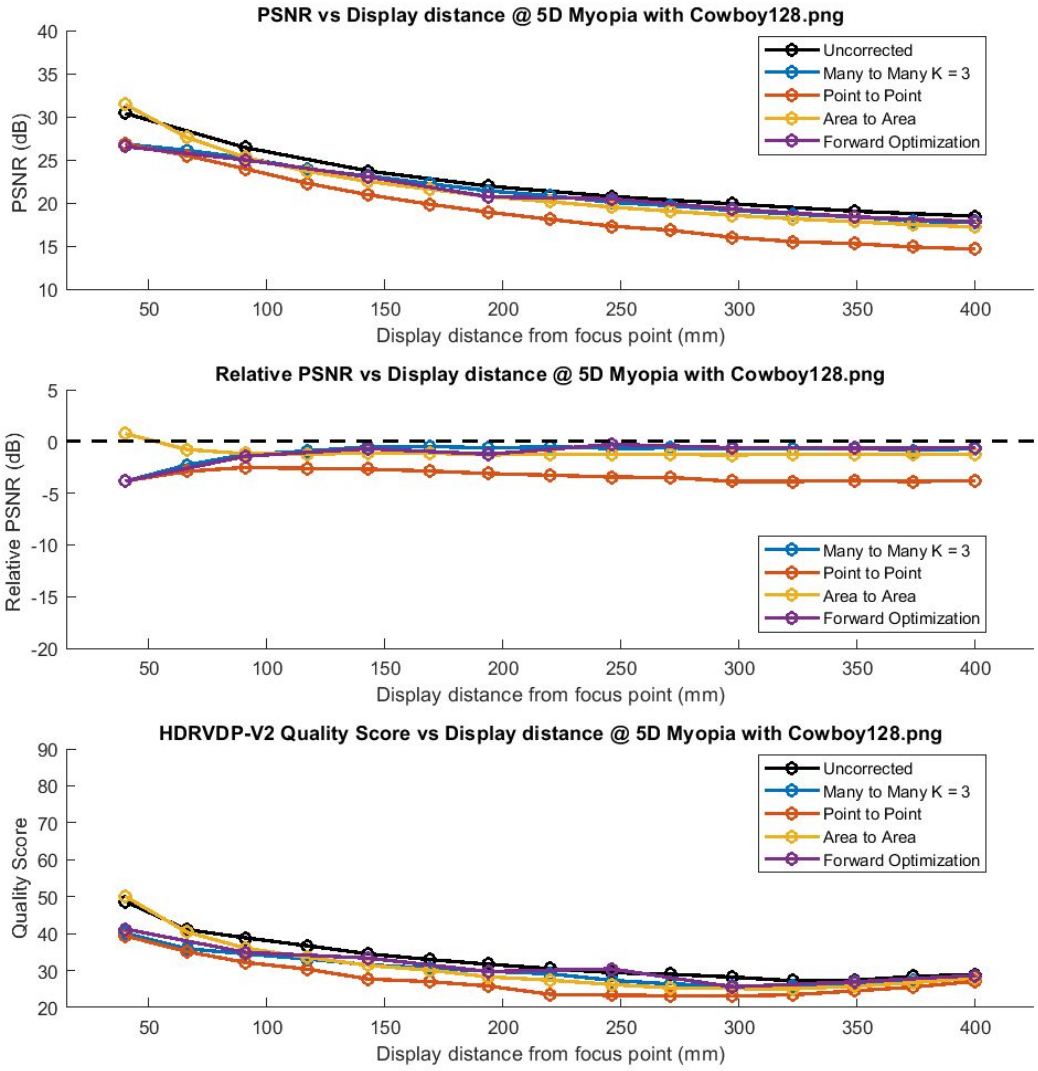
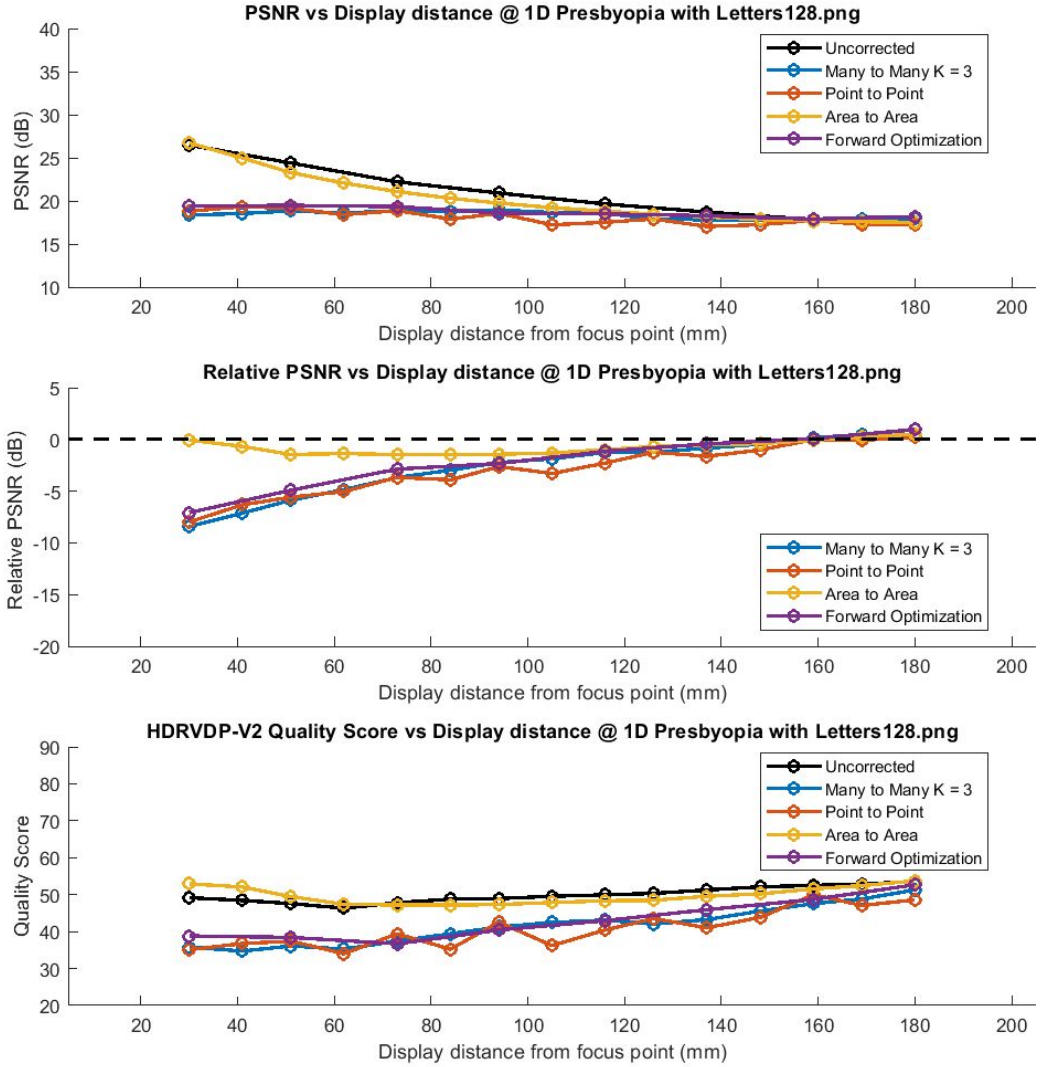
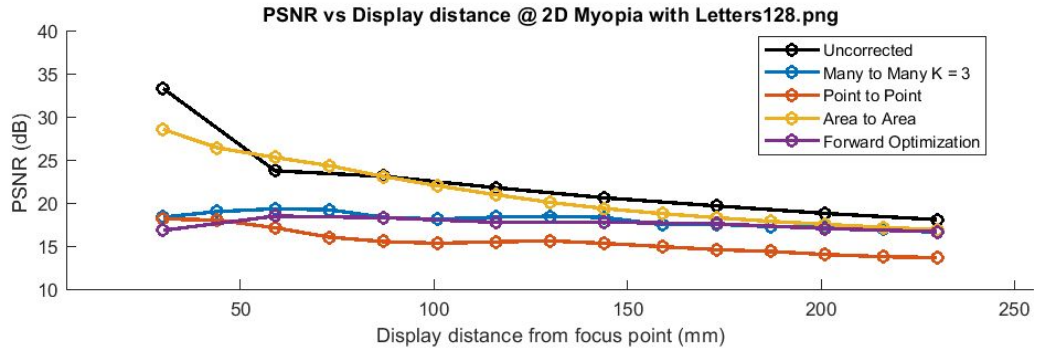
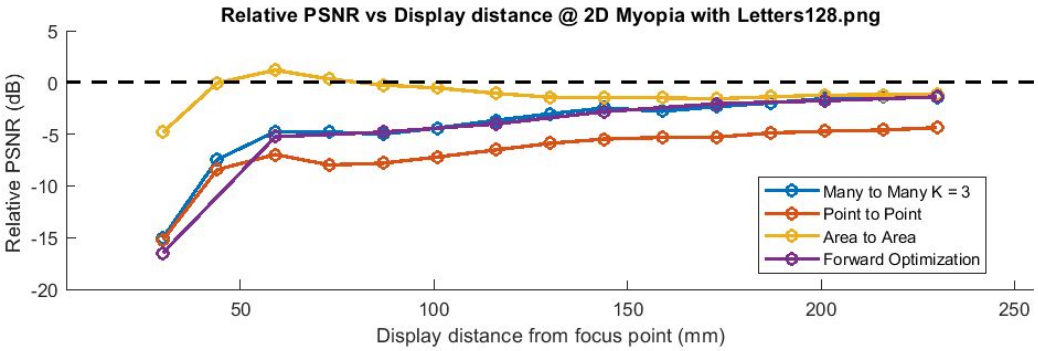
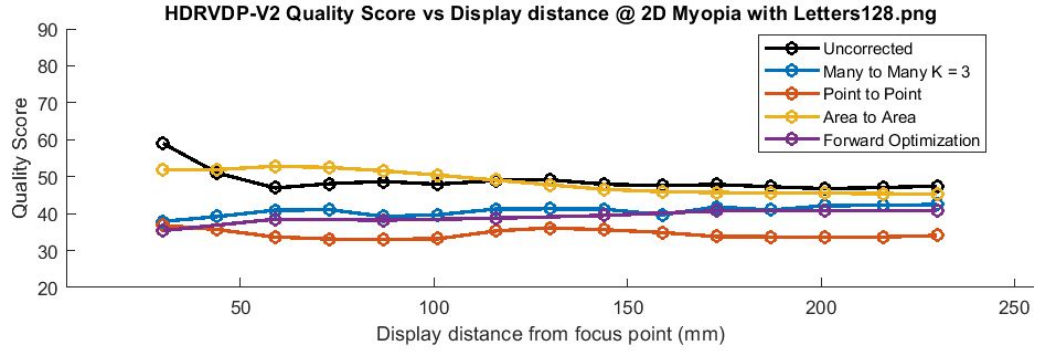

PSNR vs Display distance @ 5D Myopia with Cowboy128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization
Relative PSNR vs Display distance @ 5D Myopia with Cowboy128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization
HDRVDP-V2 Quality Score vs Display distance @ 5D Myopia with Cowboy128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

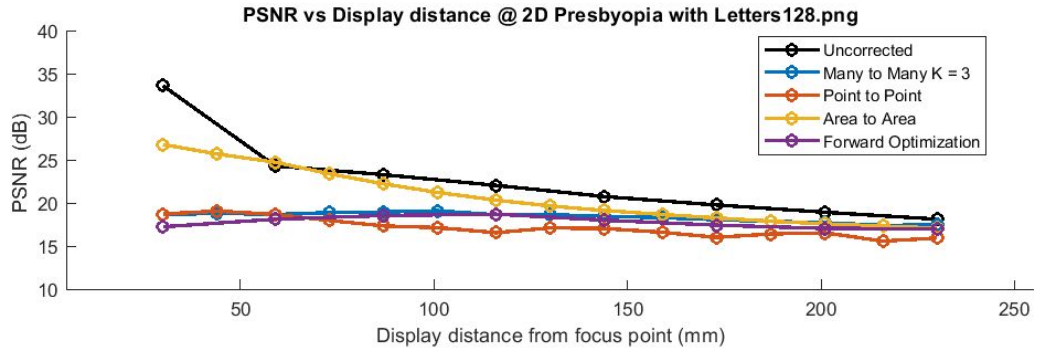
Image Quality Metrics of Letters

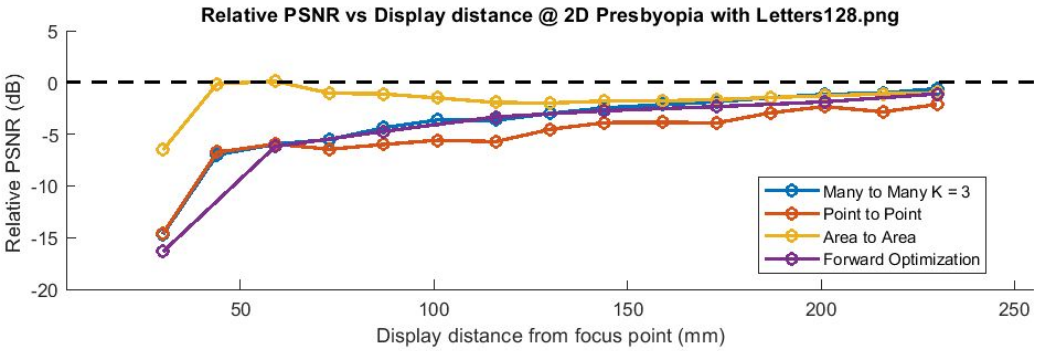


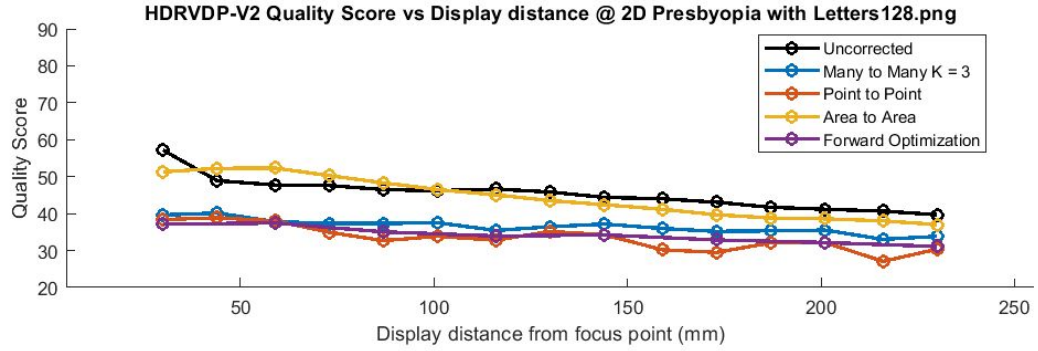

PSNR vs Display distance @ 2D Myopia with Letters128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

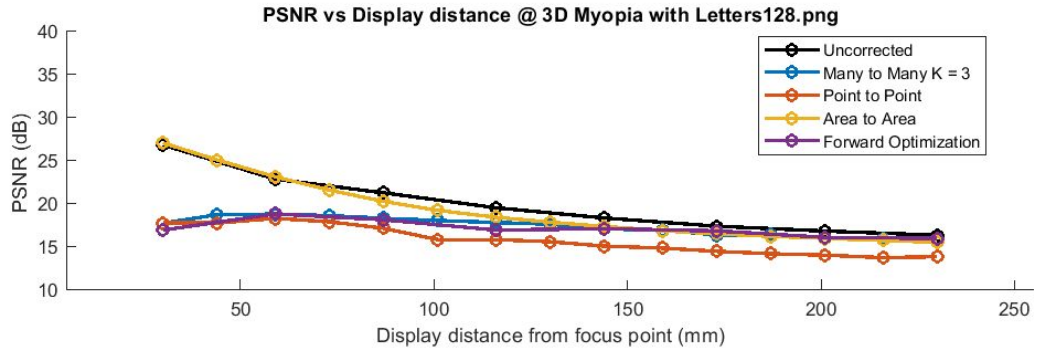

Relative PSNR vs Display distance @ 2D Myopia with Letters128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

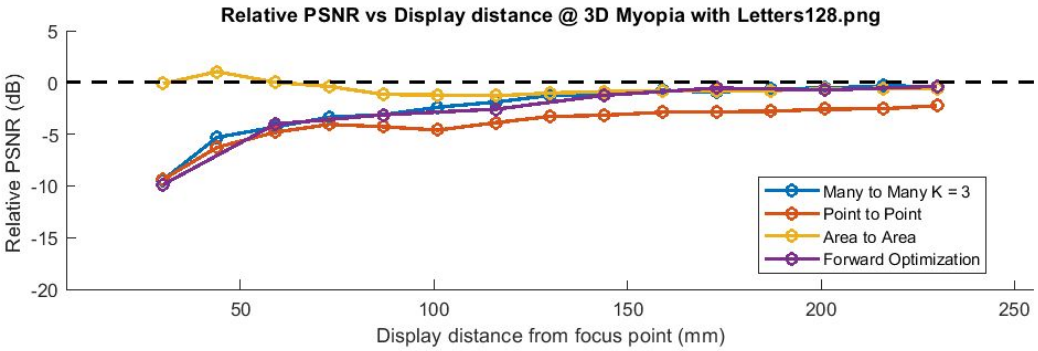

HDRVDP-V2 Quality Score vs Display distance @ 2D Myopia with Letters128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

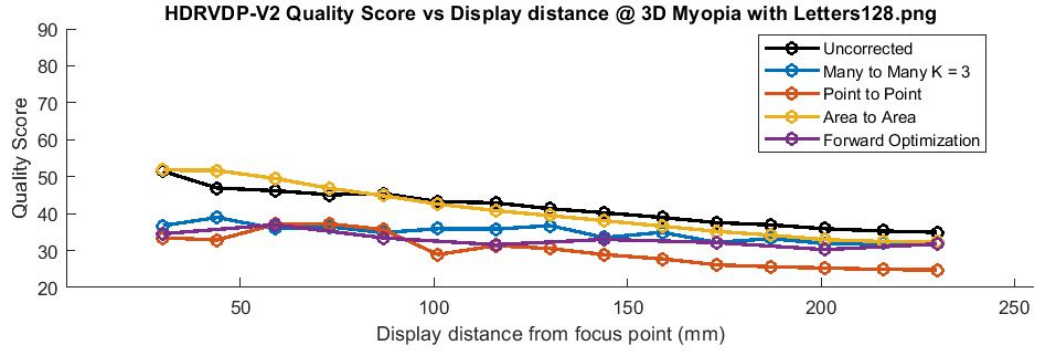

PSNR vs Display distance @ 2D Presbyopia with Letters128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

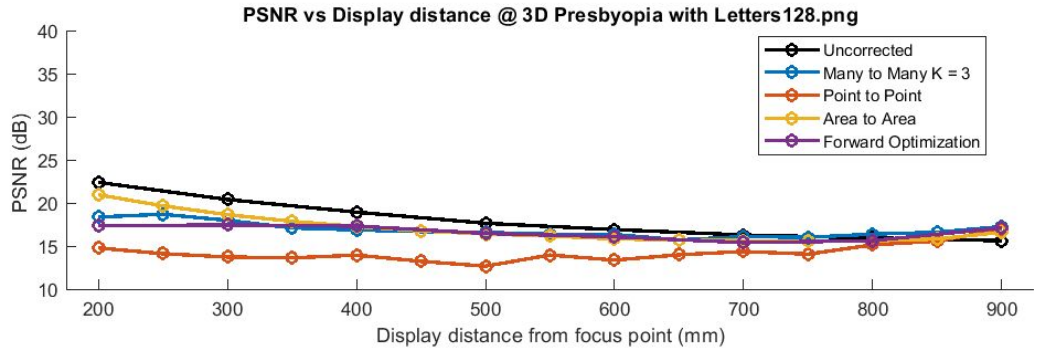

Relative PSNR vs Display distance @ 2D Presbyopia with Letters128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

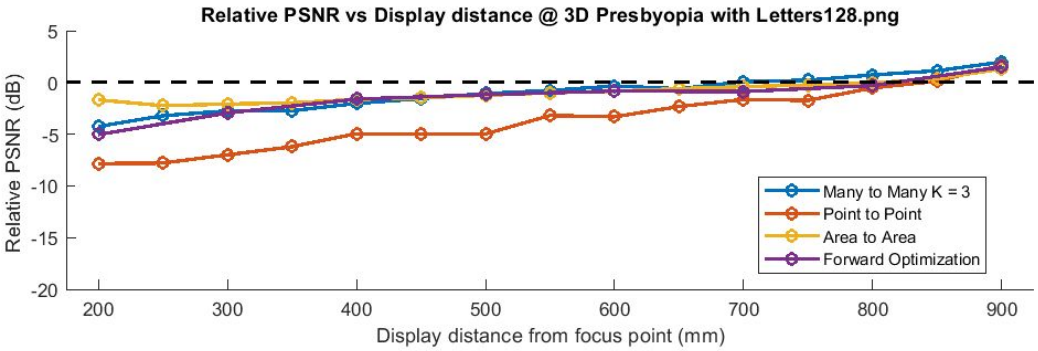

HDRVDP-V2 Quality Score vs Display distance @ 2D Presbyopia with Letters128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

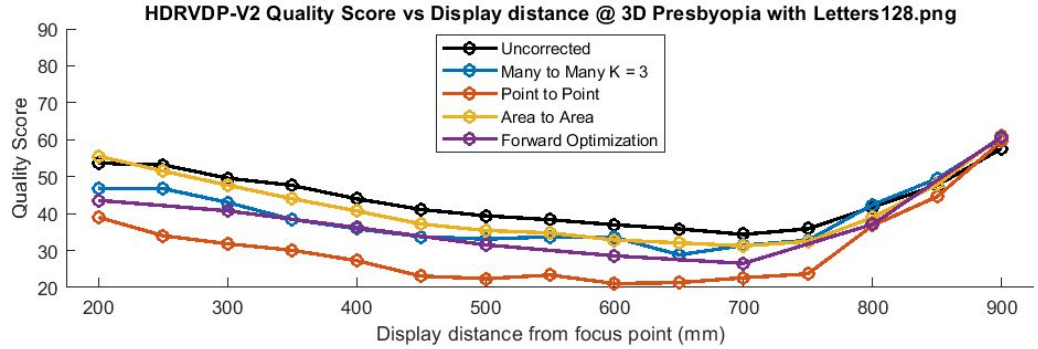

PSNR vs Display distance @ 3D Myopia with Letters128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


Relative PSNR vs Display distance @ 3D Myopia with Letters128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


HDRVDP-V2 Quality Score vs Display distance @ 3D Myopia with Letters128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


PSNR vs Display distance @ 3D Presbyopia with Letters128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


Relative PSNR vs Display distance @ 3D Presbyopia with Letters128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


HDRVDP-V2 Quality Score vs Display distance @ 3D Presbyopia with Letters128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

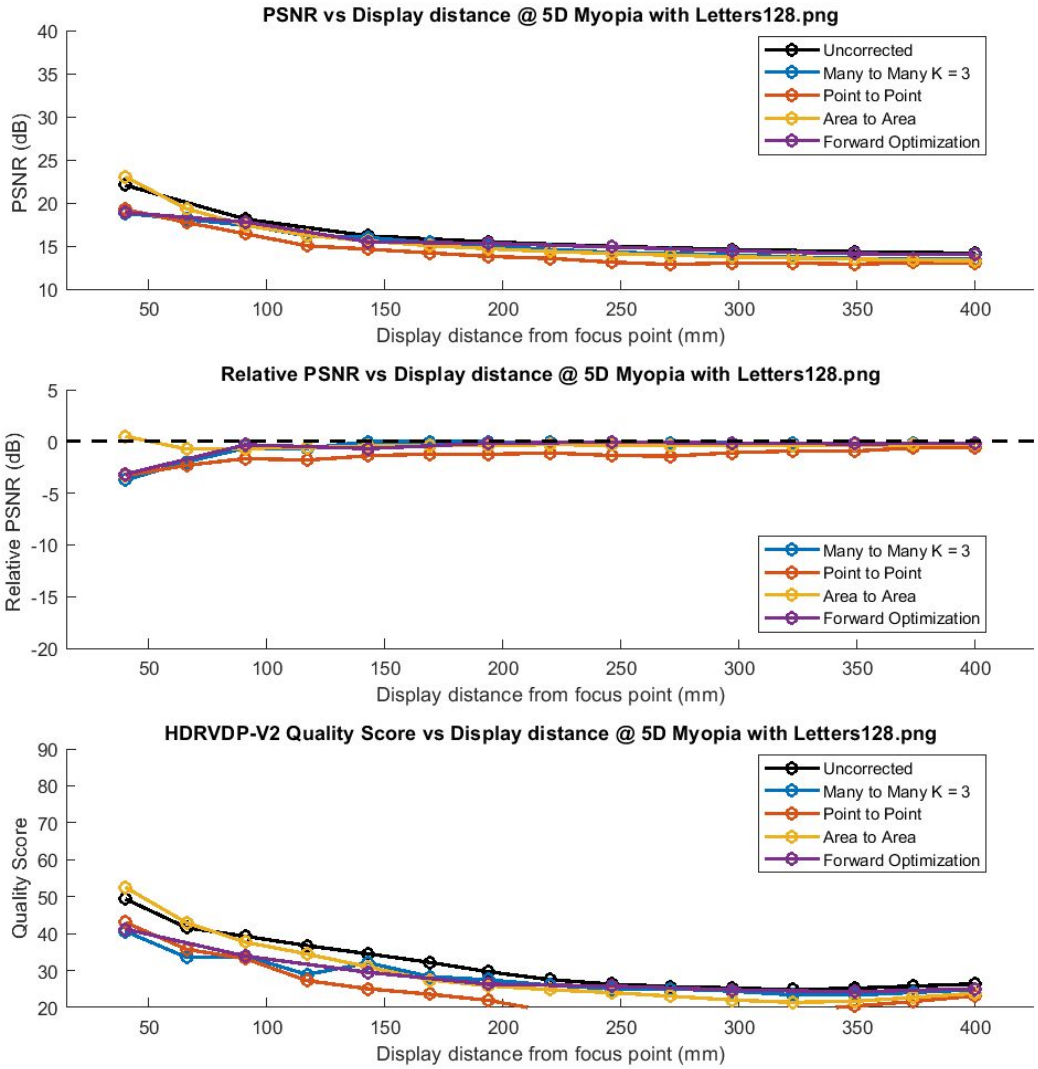
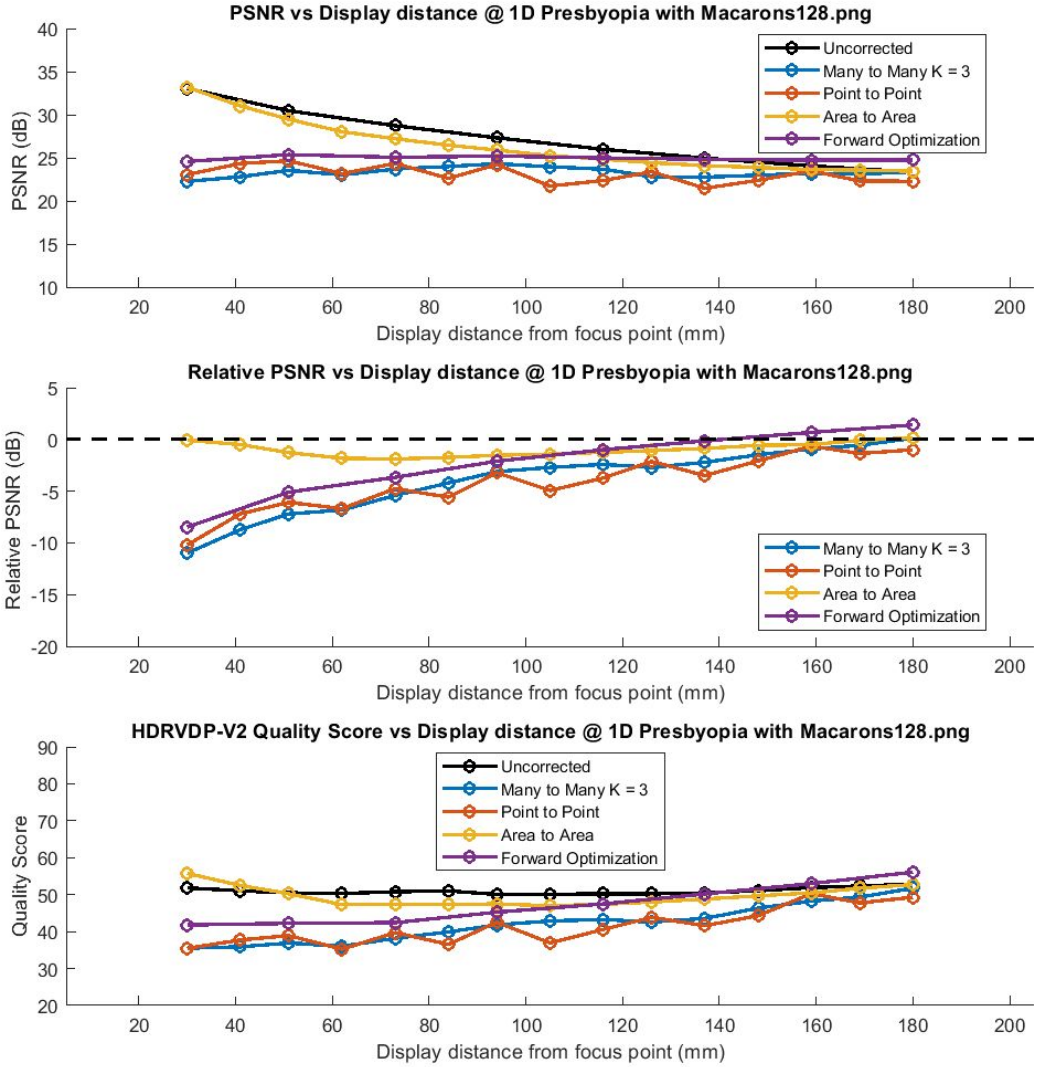
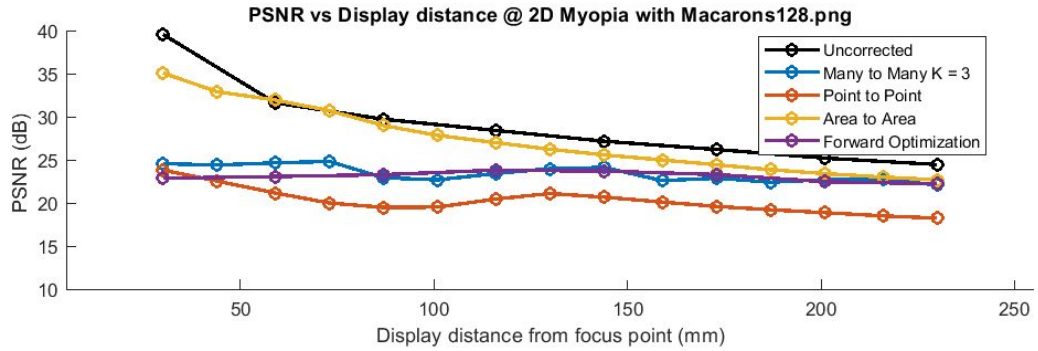
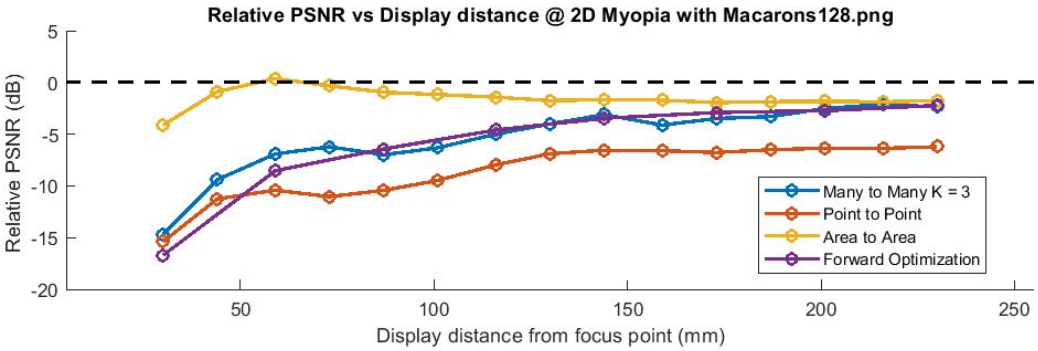
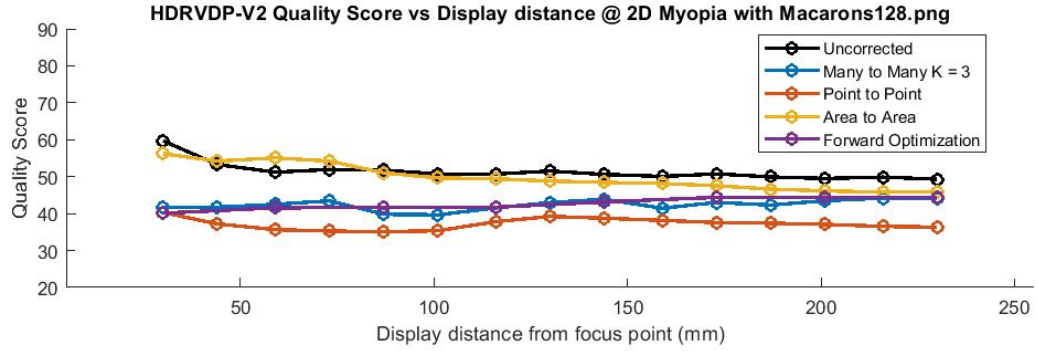

PSNR vs Display distance @ 5D Myopia with Letters128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization
Relative PSNR vs Display distance @ 5D Myopia with Letters128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization
HDRVDP-V2 Quality Score vs Display distance @ 5D Myopia with Letters128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

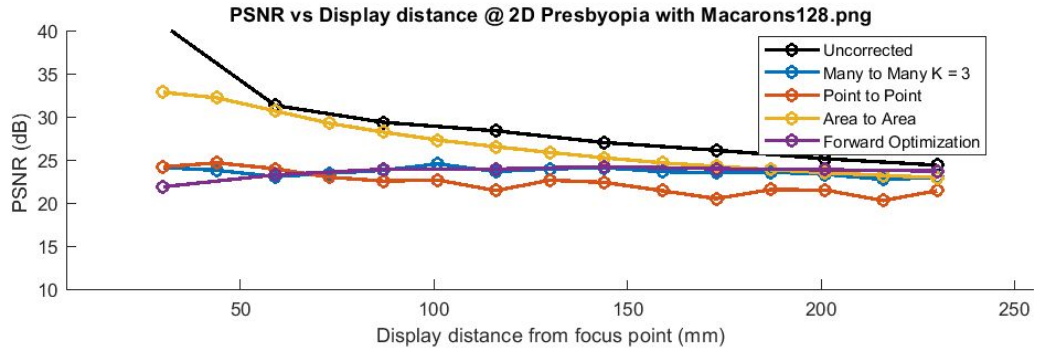
Image Quality Metrics of Macarons

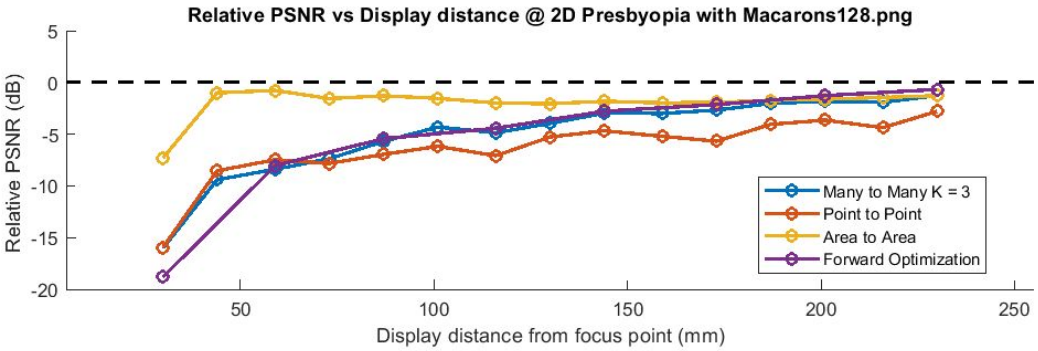


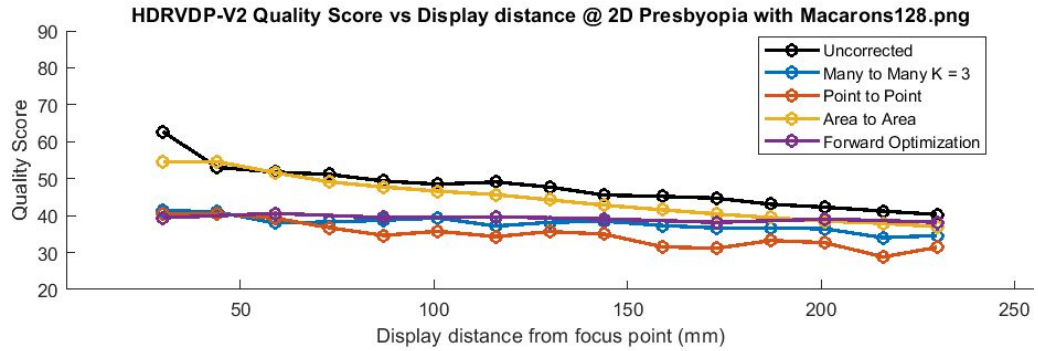

PSNR vs Display distance @ 2D Myopia with Macarons128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

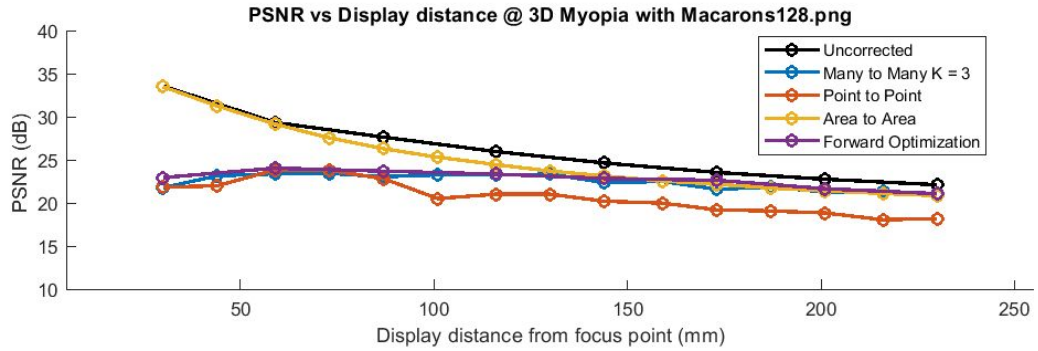

Relative PSNR vs Display distance @ 2D Myopia with Macarons128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

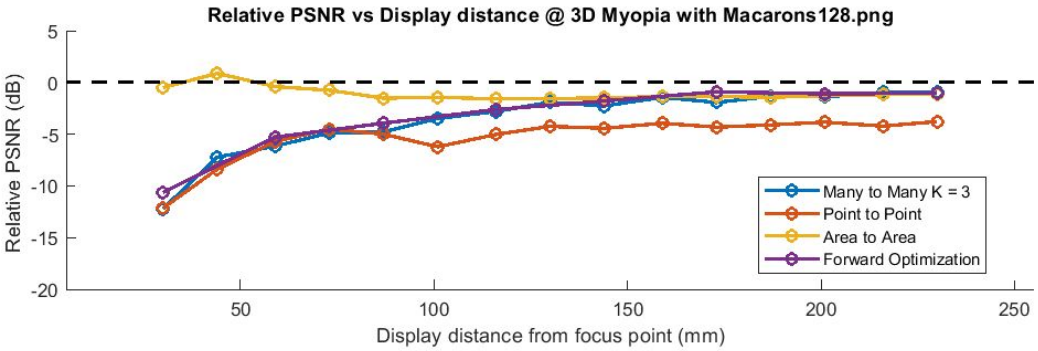

HDRVDP-V2 Quality Score vs Display distance @ 2D Myopia with Macarons128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


PSNR vs Display distance @ 2D Presbyopia with Macarons128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


Relative PSNR vs Display distance @ 2D Presbyopia with Macarons128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


HDRVDP-V2 Quality Score vs Display distance @ 2D Presbyopia with Macarons128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


PSNR vs Display distance @ 3D Myopia with Macarons128.png
PSNR (dB)
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization


Relative PSNR vs Display distance @ 3D Myopia with Macarons128.png
Relative PSNR (dB)
Display distance from focus point (mm)
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

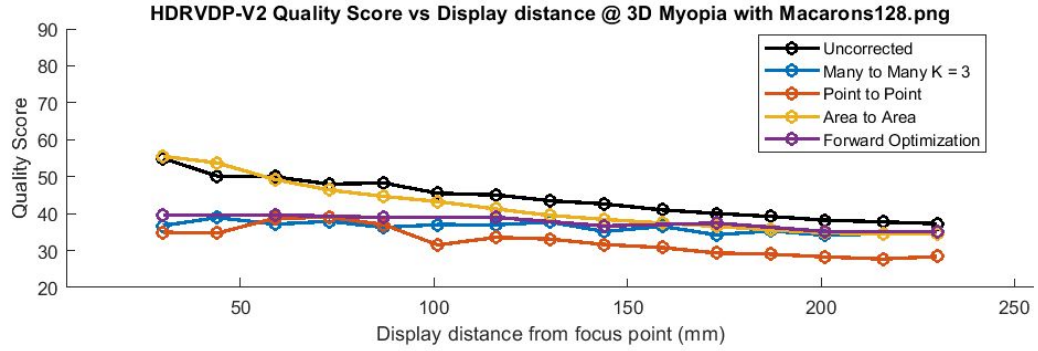

HDRVDP-V2 Quality Score vs Display distance @ 3D Myopia with Macarons128.png
Quality Score
Display distance from focus point (mm)
Uncorrected
Many to Many K = 3
Point to Point
Area to Area
Forward Optimization

Image Quality Metrics of Mandrill

Image Quality Metrics of NotreDame

References

Alonso, M., & Barreto, A. B. (2003, September). “Pre-compensation for high-order aberrations of the human eye using on-screen image deconvolution.” In Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No. 03CH37439) (Vol. 1, pp. 556-559). IEEE.

Barsky, Brian A. “Vision-realistic rendering: simulation of the scanned foveal image from wavefront data of human subjects.” In Proceedings of the 1st Symposium on Applied perception in graphics and visualization, pp. 73-81. 2004.

Hashemi, Hassan et al. “Global and regional estimates of prevalence of refractive errors: Systematic review and meta-analysis.” Journal of current ophthalmology vol. 30,1 3-22. 27 Sep. 2017, doi:10.1016/j.joco.2017.08.009

Holden, Brien A., et al. “Global Prevalence of Myopia and High Myopia and Temporal Trends from 2000 through 2050.” *Ophthalmology*, vol. 123, no. 5, 2016, pp. 1036–1042., doi:10.1016/j.ophtha.2016.01.006.

Huang, Fu-Chung, Douglas Lanman, Brian A. Barsky, and Ramesh Raskar. “Correcting for optical aberrations using multilayer displays.” ACM Transactions on Graphics (TOG) 31, no. 6 (2012): 1-12.

Hsieh, Y. H., Y. T. Yu, Y. H. Lai, M. X. Hsieh, and Y. F. Chen, “Integral-based parallel algorithm for the fast generation of the Zernike polynomials.” Opt. Express 28, 936-947 (2020)

Huang, Fu-Chung, Gordon Wetzstein, Brian A. Barsky, and Ramesh Raskar. 2014 “Eyeglasses-free display: towards correcting visual aberrations with computational light field displays.” *ACM Transactions on Graphics* (TOG) 33, no. 4: 1-12.

Mantiuk, Rafal, Kil Joong Kim, Allan G. Rempel, and Wolfgang Heidrich. “HDR-VDP-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions.” ACM SIGGRAPH 2011 Papers: pp. 40:1–40:14.

Pamplona, Vitor F., Manuel M. Oliveira, Daniel G. Aliaga, and Ramesh Raskar. “Tailored displays to compensate for visual aberrations.” ACM Transactions on Graphics (TOG) 31, no. 4 (2012): 1-12.

Wang, Zhou, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity.” in IEEE Transactions on Image Processing, vol. 13, no. 4 (2004): pp. 600-612.

Wang, Zilong, and Shuangjiu Xiao. “Simulation of Human Eye Optical System Properties and Depth of Field Variation.” International Journal of Machine Learning and Computing 3, no. 5 (2013): 413.

Wu, Zehao. “Investigating Computational Approaches and Proposing Hardware Improvement to the Vision Correcting Display.” Master’s thesis, University of California, Berkeley, 2016.

Yellott, John I., and John W. Yellott. “Correcting spurious resolution in defocused images.” In Human Vision and Electronic Imaging XII, vol. 6492, p. 64920O. International Society for Optics and Photonics, 2007.

Zhen, Yirong. “New Algorithms for the Vision Correcting Display.” Master’s thesis, University of California, Berkeley, 2016.