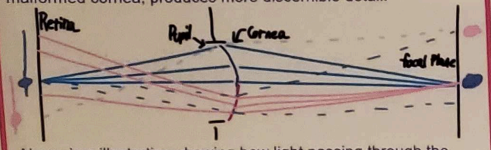
Inverting Keratoconus Refractive Errors On Screen

Paul Duggan

Motivation

Keratoconus is a degenerative eye condition that results from the cornea, the part of the eye in front of the lens, becoming malformed over time. This results in all light entering the eye having its incidence angle offset by a function of the part of the cornea it goes through. In essence, this results in the image perceived by the eye being convolved with some PSF we denote A. Since the cornea remains relatively stable on a short term basis, this kernel is typically fairly static and can be formalized as $A=(1-\alpha)K_g+\alpha K_b$ where K_g is the PSF of a healthy eye and K_b is the PSF of the damaged portion. We assume α isn't excessively close to 1 in this project (<0.7 or so). Keratoconus can make it difficult to resolve detail in images, especially in dark settings with some bright points of light. The aim of this project is to create a tool to quickly and easily learn an individual's PSF and then create a 2D image that, when perceived through the malformed cornea, produces more discernible detail.



Above is a illustration showing how light passing through the damaged portion of the cornea results in a convolved image of the focal plane being projected onto the retina. Notice how, if the orange dot were far brighter than the purple one, the purple dot may become difficult to resolve.

Related Work

There currently exist no methods to cure or reverse keratoconus medically, barring risky and invasive surgeries such as cornea transplants. The most common way to restore vision quality is to physically correct the shape of the cornea by using a hard contact lens with a similar IOR to the cornea to mitigate refraction disparities caused by its uneven thickening and thinning. Application and wearing of these contacts, however, can damage the eye further. Aside from direct corrections to the eye's biomechanics, there is research on modifying the image source to correct general refractive errors. Lightfields can create a 4d image where both the position and direction of outgoing rays are controlled. Through this, the part of the pupil a given pixel's ray goes through can be chosen, allowing for the direct accommodation of the PSF and for the desired image to be created on the retina. This method is very sensitive to the pupil position in all three axis of space as well as its rotation, and requires specialized hardware to be used. There is also research on finding the eye's refractive error with common hardware. NETRA has created an interactive application that puts the human in the measurement loop to evaluate refraction, allowing only a screen and human feedback to be used instead of direct scanning of eye topology with specialized hardware.

References

[1] Moschos, Nitoda, Georgoudis, Balidis, Eleftherios, Karageorgiadis, Kozeis, Contact Lenses for Keratoconus-Current Practice, The Open Ophthalmology Journal, 2017

[2] Huang, Wetzstein, Barsky, Raskar, Eyeglasses-Free Displays Siggraph 2014

[3] Pamplona, Mohan, Oliveira, Raskar, NETRA: Interactive Display for Estimating Refractive Errors and Focal Range, ACM Siggraph Papers, 2010

[4] Wang, Bovik, Sheikh, Simoncelli, Image Quality Assessment: From Error Visibility to Structural Similarity, IEEE Transactions on image processing, 2004

Step 1: Learn PSF

First, we create a interactive loop to learn the PSF kernel A by using human feedback from their own vision. Rather than using any hardware or screen light filters, we use only a conventional 2d screen, requiring only that the user stay at a fixed distance and rotation relative to the screen. We have the user draw the deviation from a single dot they see in a given image, using fast toggling to precisely align the drawn error with their sight..

- Initialize $b = b_g + b_d \in \mathbb{R}^{w \times h}$ where b_g is a solid gray image and b_d is one at its center, 0 elsewhere. Then, in a loop, letting A_0 be the 'identity convolution'
- Compute $x_i = \min_{x \in \mathbb{R}^{w \times h}} ||A_i * x - b||_2$
- Present x_i to the user, who perceives $A * x_i$
- The user can toggle between x_i and b_g and is instructed to draw features from $A * x_i$ onto b_g such that they align
- The user returns this drawing y_i. Note that $y_i \approx A * x_i - \alpha b_d$
- Assuming $\mathbb{E}[y] = A * x_i - \alpha b_d$, we see we want

$$A_{i+1} = \min_{\hat{A}} \left|\left| \hat{A} * x_i - A * x_i \right|\right|_2 = \min_{\hat{A}} \left|\left| \hat{A} * x_i - \mathbb{E}[y + \alpha b_d] \right|\right|_2^2$$

so we take a SGD step where $\frac{\partial Loss}{\partial x_i} = A_i * x_i - (y_i + \alpha b_d)$ where $\alpha = \left|\left| K_b^{(i)} \right|\right|_1$ and constrain A_{i+1} to be positive and sum to 1.

Step 2: Find Inverse Image

Our goal is to find, for a given image b, a prior x such that A*x is close to b. This inverse is not feasible without negative radiance for most images and kernels. Therefore, we make the decision that it is more important to maintain structural information than it is to keep overall brightness the same. By locally increasing the 'floor brightness' of a given image where needed, we can fake negative radiance, allowing for contrast details to be seen at the cost of direct image accuracy. We want a metric that we can tune to prioritize structural information over other brightness image. For this, we utilize MSSIM error with a low luminance weighting. MSSIM error is the average SSIM error over patches of the image. For each patch, we compute the weighted luminance means, standard deviations and covariance. The final error is given by (adding an additional epsilon term to fractions)

$$-\left(\frac{2\mu_1\mu_2}{\mu_1^2+\mu_2^2}\right)^{\alpha}\left(\frac{2\sigma_1\sigma_2}{\sigma_1^2+\sigma_2^2}\right)^{\beta}\left(\frac{Cov_{1,2}}{\sigma_1\sigma_2}\right)^{\gamma}$$

Where to facilitate sacrificing luminance quality for structural quality, we use a small alpha and large gamma.To find x, we simply perform gradient descent on MSSIM(A*x, b)

Results

The interactive PSF guessing tool, while functional, is slow and difficult to use. It requires highly precise inputs since drawing even a pixel off will result in error that must be corrected in the next step. While this offset is visually apparent, it is frustrating to fix in practice due to the difficulty of precisely moving a mouse for each part of the PSF. Tens of loops are required for high-quality convergence considering expected human error, making its usage very laborious. On the other hand, after a long period of use, I did get a good estimate for my own eye's PSF.

The kernel inverter worked very well. Using the chosen MSSIM parameters (0.03,1,2 respectively for luminance, contrast and similarity) did the best job of maintaining image detail while retaining a semblance of detail. Varying by the strength of the distortions (α), yields the following quality improvements.

α	$MSSIM_{orig}$	$MSSIM_{after}$	MSE increase
0.6	0.67	0.89	51%
0.45	0.78	0.95	43%
0.3	0.87	0.97	78%