An Algorithm for Rapid Measurement of Aberrations in Pairs of Out-Of-Focus Images

by

Ryan J. Janish

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

Author .....
Department of Physics
May 13, 2012

Certified by .....
Paul L. Schechter
William A. M. Burden Professor of Astrophysics
Thesis Supervisor

Accepted by .....
Nergis Mavalvala
Thesis Coordinator, Department of Physics

An Algorithm for Rapid Measurement of Aberrations in Pairs of Out-Of-Focus Images

by

Ryan J. Janish

Submitted to the Department of Physics
on May 13, 2012, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Physics

Abstract

In this thesis, I present a new technique for measuring the optical aberrations produced by a telescope, with an eye towards future use of these aberration measurements to align wide-field telescopes. This method determines the aberrations by simultaneously fitting a pair of oppositely defocused images to a mostly analytic model. I develop the model and describe its software implementation in detail, and then report on the results of tests with simulated and real data. This technique is able to extract the aberrations from simulated data rapidly and accurately, and it has been used with mixed success to analyze data from the VISTA telescope. With the VISTA data, the algorithm is unable to match small-scale brightness variations in the images. However, it was able to determine aberrations with median accuracies of 0.08 um for coma, 0.08 um for astigmatism, 0.9 um for tilt, and 0.3 um for defocus. It was also quite fast, with an average of 34 iterations until convergence.

Thesis Supervisor: Paul L. Schechter
Title: William A. M. Burden Professor of Astrophysics

Acknowledgments

I have been able to complete this thesis only by the gracious help of many people. First of all, I am deeply thankful for my incredible wife, Amanda. Without her support and help, this thesis would certainly never have come to be. I would have never survived the rigors of MIT without her by my side, nor would I have had nearly as good of a time doing so.

I want to thank my advisor, Professor Paul Schechter, who has shown me much patience and guidance during this endeavor. I have learned a lot by working with him, both about the topic of this thesis and about research in general, and by his influence I am certainly a much better scientist now than when I began this work.

I would also like to thank Nancy Savioli, Nancy Boyce, and all of the administrators who have helped me navigate the Physics Department, both while completing this thesis and during my entire time here as an undergraduate. It is such friendly and helpful people that make this Department such a great place to be.

Finally, this work makes extensive use of data that I did not collect. I would like to thank Thomas Szeifert, Andreas Kaufer and Magda Arnaboldi for arranging for the VISTA data used in this work to be recorded.

Contents

Chapter 1

Introduction

Wide field telescopes are very sensitive to misalignments and require frequent adjustments to ensure that the collected data is of high quality. [7] The development of fast and accurate alignment techniques is therefore an integral part of fully utilizing the capabilities of large telescopes. This is a particularly important concern in the weak lensing community, as the errors caused by misalignments appear as a false lensing signal and the lensing signal itself is very small. [5]

To correct a misalignment, one must first measure it. The basic strategy for determining misalignments has been to measure the optical aberrations of the telescope in question, which are related to the misaligning displacements and rotations of the mirrors. [7] Until recently, the most common method for measuring aberrations has been the use of Shack-Hartmann wavefront sensors [7] [9], however the latest generation of wide-field telescopes, such as VISTA [2] and the upcoming LSST [3] and DES [4], instead rely on analyzing pairs of out-of-focus images. The VISTA collaboration was the first to successfully realize this technique, using an image analysis algorithm based on comparing observed images to those simulated with ray-tracing methods. This is time-consuming, as it can take hundreds of iterations for such an algorithm to converge. [1]

In this thesis, I will present an alternative method that fits out-of-focus image pairs directly to a mostly analytic model, which promises to be a much faster technique. In Chapter 2, I discuss the optical theory behind the model and develop the necessary

mathematics. Chapter 3 focuses on the software implementation of the model, where I discuss at a high level all of the principle algorithms involved. Chapter 4 discusses the results of testing the fit procedure on both simulated data and data taken with the VISTA telescope. Finally, in Appendices A and B, I give all of the practical details necessary for the reader to understand the current progress of this work and be able to develop it further. In Appendix A I derive all of the mathematical results necessary to implement the algorithms discussed in Chapter 3, and in Appendix B I discuss my implementation in more detail and at a much lower level than in Chapter 3.

Chapter 2

Image Model

The goal of this chapter is to develop a model of out-of-focus images of point sources. I begin by briefly introducing some of the theory of optical aberrations, and then use these notions to construct a description of defocused images.

2.1 Aberrations and Imaging

The propagation of light through an optical system can be described in terms of the surfaces of constant phase that lie perpendicular to the light's direction of travel, known as the *wavefronts*. [1] While there is a wavefront passing through any point in the light's path, of particular interest is the wavefront that passes through the center of the optical system's exit pupil. We can parameterize this wavefront surface as a function $\omega(\vec{x}_p)$, where $\omega(\vec{x}_p)$ denotes the distance between the wavefront surface and the plane of the exit pupil at point $\vec{x}_p$ on the pupil. Now, for a perfect optical system, that is, for a system in which the light leaving the pupil is focused to a single focal point located a distance f away from the pupil along the optical axis, the central wavefront $\omega(\vec{x}_p)$ must be a spherical cap from a sphere of radius f centered on the focal point. For any real optical system, the wavefront $\omega(\vec{x}_p)$ will differ from a perfect spherical cap, and we can therefore characterize the imperfections of an optical system

[1] I present a very shallow treatment of wavefronts. For those interested, a thorough discussion of wavefront optics can be found in Majahan's text *Optical imaging and aberrations, part I* [6]

by the deviation of its wavefront from a sphere. Define the *wavefront deviation* to be

$$W(\vec{x}_p) = \omega(\vec{x}_p) - \omega_{\text{sphere}}(\vec{x}_p) \tag{2.1}$$

where $\omega_{\text{sphere}}(\vec{x}_p)$ is the spherical wavefront the system would have if it focused perfectly and $\omega(\vec{x}_p)$ is the actual wavefront. For simplicity, I will typically refer to $W(\vec{x}_p)$ as just *the wavefront*.

Describing the imaging process in terms of $W(\vec{x}_p)$ is particularly useful for two reasons:

1. $W(\vec{x}_p)$ tells us where a ray that crosses the pupil at $\vec{x}_p$ will eventually intersect the image plane. If the image point is denoted by $\vec{x}_i$, then (in Cartesian coordinates):

   $$\vec{x}_i = \frac{f}{R_{out}} \vec{\nabla}_p W(\vec{x}_p) \tag{2.2}$$

   where $\vec{\nabla}_p$ indicates the gradient with respect to the pupil position, f is the focal length, and R_{out} is the outer pupil radius. [6]

Thus, if we know the function $W(\vec{x}_p)$, we can use it to construct a transformation that maps pupil points to image points. Note that the choice of f/R_{out} as a pre-factor is mere convention. This allows us to both have $\vec{x}_p$ be a dimensionless quantity that is normalized to 1 at the outer pupil radius and also let $W(\vec{x}_p)$ be measured in units of length.

2. We *do* know the function $W(\vec{x}_p)$, approximately, and with seven free parameters. For a realistic imaging system, $W(\vec{x}_p)$ is well-approximated by the following third-order polynomial:

$$\begin{aligned} W(x_p, y_p) \approx\ & c_x x_p^3 + c_y y_p^3 + c_y x_p^2 y_p + c_x x_p y_p^2 \\ & + (d + a_x) x_p^2 + (d - a_x) y_p^2 + 2 a_y x_p y_p \\ & + t_x x_p + t_y y_p \end{aligned} \tag{2.3}$$

The seven parameters c_x, c_y, a_x, a_y, t_x, t_y and d are referred to as the *third-order*

optical aberrations. [6] Specifically, c_x and c_y are the *coma*, a_x and a_y are the *astigmatism*, t_x and t_y are the *tilt*, and d is the *defocus*. These parameters are functions of the physical properties of the imaging system.

Note that the above aberration parameters are *not* the usual Zernike coefficients. For a conversion to Zernikes, see Section 2.5. By the conventions stated above for the units of $W(\vec{x}_p)$ and $\vec{x}_p$, all of the aberration parameters are measured in units of length. And in particular, since they are the only dimension-full quantities in the wavefront, we say they are measured in *distance of wavefront error*.

Combining Equations 2.2 and 2.3 gives an explicit transformation between the pupil and image planes:

$$\begin{aligned} x_i &= 3c_x x_p^2 + c_x y_p^2 + 2c_y x_p y_p + 2(d + a_x)x_p + 2a_y y_p + t_x \\ y_i &= c_y x_p^2 + 3c_y y_p^2 + 2c_x x_p y_p + 2a_y x_p + 2(d - a_x)y_p + t_y \end{aligned} \tag{2.4}$$

This transformation completely determines the observed image, provided we know the distribution of rays across the pupil. Now, if we restrict attention to the case of imaging a distant point source, then it is reasonable to assume that the incoming light is uniformly distributed across the pupil, and therefore the aberration parameters fix the resulting image.

2.2 Brightness

In order to compare an observed image with the image specified by a given set of aberration parameters, it is necessary to know the brightness of any given pixel in an image as a function of the seven aberration parameters.

Suppose there is a pixel centered on $\vec{x}_i$ with a small area dA_i. If a total power dP is being delivered to this pixel, the intensity averaged over the pixel is given by $I_i(\vec{x}_i) = dP/dA_i$, and $I_i(\vec{x}_i)$ is proportional to the observed brightness. Now, let $\vec{x}_p$ be the pupil point [2] that maps to $\vec{x}_i$ via the transformation given in Equation 2.4. Since

[2] While in principal there could be multiple pupil points that map to one image point, I will assume there is only one. For a physical justification of this, see Section 3.1.

this transformation is continuous, the area dA_i of the pixel will map to some small area dA_p surrounding the point $\vec{x}_p$ on the pupil plane, and we must have that a power dP is crossing the area dA_p as well. If we restrict attention to images of distant point sources, then we can assume that the intensity across the pupil is constant. Denote this constant by F, which will depend on the physical properties of the source and imaging system, as well as weather conditions and exposure time. Now, for any realistic imaging system, only a finite area of the pupil plane is actually illuminated and it is possible that part of the area dA_p lies outside the illuminated region. Suppose that a fraction f of the area dA_p is illuminated. Then,

$$dP = F f dA_p$$

and the intensity of the image pixel is:

$$I(\vec{x}_i) = F f \frac{dA_p}{dA_i}$$

Assuming that the areas dA_p and dA_i are small, then their ratio dA_p/dA_i is just the inverse of the Jacobian determinant of the transformation given by Equation 2.4. Now, the Jacobian is the matrix of first derivatives of the transformation in question, and since in this case the transformation is given by $\vec{x}_i = \vec{\nabla}_p W(\vec{x}_p)$, the Jacobian will consist of the second derivatives of the wavefront $W(\vec{x}_p)$. The matrix of second derivatives of a function is called the *Hessian* and I will denote the Hessian of the wavefront as $H(\vec{x}_p)$. So we have that

$$I(\vec{x}_i) = \frac{F}{\det H(\vec{x}_p)} f \tag{2.5}$$

where the right-hand side is evaluated at the the pupil point $\vec{x}_p$ that maps to $\vec{x}_i$. On the right-hand side of Equation 2.5, F is a constant particular to the image being recorded, $H(\vec{x}_p)$ will depend on $\vec{x}_p$ and the aberration parameters, and $f(\vec{x}_p)$ will depend on $\vec{x}_p$ and physical properties of the telescope, such as pupil size and pixel size. Now, $\vec{x}_p$ can be computed from $\vec{x}_i$ and the aberration parameters using the

transformation in Equation 2.4. Thus, Equation 2.5 gives the brightness at $\vec{x}_i$ as a function of only $\vec{x}_i$, the aberration parameters, the total flux parameter F, and physical properties of the telescope.

Equation 2.5 is the foundation of the defocused image model.

2.3 Defocus-Only Formulation

The purpose of this model is to fit for aberrations in pairs of images that have been intentionally defocused. As such, the goal is to quantify the effect that the other six aberrations, plus any additional, unintentional defocus, has on the images. The "perfect" image in this case is not a single point, but rather a uniform annulus (assuming that the pupil is a perfect annulus) with a distribution of rays that is exactly the pupil brightness distribution scaled by $2d$. This factor of $2d$ results from Equation 2.4, which says that the position of a ray on a perfect image with all aberrations zero except the defocus, is given by:

$$\vec{x}_{df} = 2d\vec{x}_p \tag{2.6}$$

Now, the implicit picture in Equation 2.4 is that we have some ray whose natural position is at $(0, 0)$, but due to the presence of aberrations it is going to be displaced to some different position $\vec{x}_i$, given by evaluating the right-hand side. But, in our case the natural position is not $(0, 0)$, but rather $2d\vec{x}_p$, and so the mathematics would be more transparent if we could arrange Equation 2.4 to give the displacement from $2d\vec{x}_p$ that a ray suffers at the hands of a non-defocus aberration. To do this, eliminate $\vec{x}_p$ in Equation 2.4 using Equation 2.6 and rearrange:

$$\begin{aligned} x_i - x_{df} &= \frac{3c_x}{4d^2}x_{df}^2 + \frac{c_x}{4d^2}y_{df}^2 + \frac{c_y}{2d^2}x_{df}y_{df} + \frac{a_x}{d}x_{df} + \frac{a_y}{d}y_{df} + t_x \\ y_i - y_{df} &= \frac{c_y}{4d^2}x_{df}^2 + \frac{3c_y}{4d^2}y_{df}^2 + \frac{c_x}{2d^2}x_{df}y_{df} + \frac{a_y}{d}x_{df} - \frac{a_x}{d}y_{df} + t_y \end{aligned} \tag{2.7}$$

This is what we were after, with $\vec{x}_i$ representing the final ray position on the distorted image, and $\vec{x}_{df}$ representing its natural, unaberrated position. This equation has two advantages, the first being that it expresses more explicitly the question that we are interested in answering. The second is that, since the VISTA images we will be working with have only a few pixels of distortion (see Figure 4-1), we now know that the right-hand side of Equation 2.7 is small, on the order of a few pixel lengths. This is useful for solving Equation 2.7 (see Section 3.1), and it is also useful for justifying the assumptions made in computing the illuminated fraction f (see Section A.1).

Now, at this point we can (mostly) forget about the physical telescope pupil plane. The natural way to view Equation 2.7 is to say that we start with a *defocus-only image*, which is a uniform annular image with radii $r_{in} = 2dR_{in}$ and $r_{out} = 2dR_{out}$, where R_{in} and R_{out} are the radii of the physical telescope pupil. Then, each point on the defocus-only image suffers a translation given by 2.7. This completely describes the imaging process, up to the extent that the aberrations are small enough for the the wavefront to be well approximated by its third-order expansion, and so from this section on I work exclusively with the defocus-only description. Now, there still remains one more piece of the puzzle, that being the need to find the analog of Equation 2.5 in the defocus-only formulation. The same reasoning that led to Equation 2.5 clearly holds again, however in this case the image brightness will be given by the inverse Jacobian determinant of the transformation in Equation 2.7 multiplied by the brightness of the defocus-only image and the fraction of the image pixel that is illuminated if we displace it back to its original defocus-only position.

First, note that the Jacobian of the transformation in Equation 2.7 is inherently the same object as the Jacobian of the transformation in Equation 2.4. If we compute the Jacobian from Equation 2.4 and then apply the transformation in Equation 2.6 we will get the same result as if we differentiated Equation 2.7 directly. In a sense, we are still working with the same Hessian of the wavefront, but have just made a coordinate transformation $\vec{x}_p \to \vec{x}_d f/2d$. Thus, I will continue to denote the Hessian by H, but include the argument $H(\vec{x}_{df})$ to emphasize that I am specifically thinking of the Hessian as expressed in defocus-only coordinates.

Second, the brightness of the defocus-only image will be the brightness of the actual pupil F scaled by the ratio of the area of the pupil to the area of the defocus-only image. This ratio is $1/4d^2$, and so the defocus-only brightness is $F/4d^2$. Putting this together, the appropriate defocus-only brightness formula is:

$$I(\vec{x}_i) = \frac{F}{4d^2} \frac{1}{\det H(\vec{x}_{df})} f \tag{2.8}$$

2.4 Background and Seeing

There are two more practicalities that must be added to the model. The first is the fact that the sky has an intrinsic brightness that will apply a constant background to the images. Thus, we need to add a constant background parameter b to Equation 2.8:

$$I(\vec{x}_i) = \frac{F}{4d^2} \frac{1}{\det H(\vec{x}_{df})} f + b \tag{2.9}$$

The second effect that needs to be accounted for is atmospheric seeing. For this, apply a convolution to the unblurred model computed with Equation 2.9. For simplicity, we will take the convolution to be conical, so the weight factor is given by:

$$K(r) = \begin{cases} N\left(1 - \frac{r}{r_0}\right) & : r < r_0 \\ 0 & : r > r_0 \end{cases}$$

where r_0 is the *seeing radius*, measure in seconds of arc. N is set such that the sum of all elements in K is 1. The linear dimension of K can be chosen to be as small as possible while still giving a reliable fit, though the most accurate fit will obviously be obtained if the size of K is greater than the diameter of the weighting cone. For the VISTA data, the seeing radius varies between about 0.6" and 1.6", which is between about 2.5 and 7 pixels. A 7x7 kernel was used with good results.

2.5 Zernike Coefficients

Optical aberrations are commonly expressed not in the form of Equation 2.4, but rather in terms of the orthogonal (on the unit disk) Zernike polynomials. If we make the assumption that the aberrations are small enough that we can neglect higher order terms in Equation 2.4 and terms higher than the seventh term in a Zernike expansion, then the above seven aberration parameters can be exactly related to the coefficents of a Zernike expansion. This approximation is very likely true given the small distortion of the VISTA data (also, see Sections 2.3).

Following the convention used by Wilson, [10] the first eight Zernike polynomials are:

0. $Z_0(\vec{\rho}) = 1$
1. $Z_1(\vec{\rho}) = \rho\cos(\phi)$
2. $Z_2(\vec{\rho}) = \rho\sin(\phi)$
3. $Z_3(\vec{\rho}) = 2\rho^2 - 1$
4. $Z_4(\vec{\rho}) = \rho^2\cos(2\phi)$
5. $Z_5(\vec{\rho}) = \rho^2\sin(2\phi)$
6. $Z_6(\vec{\rho}) = (3\rho^2 - 2)\,\rho\cos(\phi)$
7. $Z_7(\vec{\rho}) = (3\rho^2 - 2)\,\rho\sin(\phi)$

To relate these to the aberration parameters used above, express Equation 2.3 in polar form:

$$W(\vec{\rho}) = C\rho^3\cos(\phi - \phi_c) + A\rho^2\cos(2\phi - \phi_a) + T\rho\cos(\phi - \phi_t) + d\rho^2 \qquad (2.10)$$

where these polar aberrations are related to the Cartesian aberrations of Equation 2.3 in the standard fashion:

$$c_x = C\cos\phi_c\ ,\ c_y = C\sin\phi_c$$

$$a_x = A\cos\phi_a\ ,\ a_y = A\sin\phi_a$$

$$t_x = T\cos\phi_t\ ,\ t_y = T\sin\phi_t$$

Using the above definitions of the polar aberration parameters, one can verify by

straightforward but tedious arithmetic that Equations 2.3 and Equations 2.10 are indeed identical. From the above list of Zernike polynomials, we can express the spacial-dependent factors in Equation 2.10 as:

$$\rho^3 \cos(\phi) = \frac{1}{2} Z_6(\vec{\rho}) - \frac{2}{3} Z_1(\vec{\rho}), \quad \rho^3 \sin(\phi) = \frac{1}{2} Z_7(\vec{\rho}) - \frac{2}{3} Z_1(\vec{\rho})$$

$$\rho^2 = \frac{1}{2} \left(Z_3(\vec{\rho}) + 1 \right)$$

$$\rho^2 \cos(2\phi) = Z_4(\vec{\rho}), \quad \rho^2 \sin(2\phi) = Z_5(\vec{\rho})$$

$$\rho \cos(\phi) = Z_1(\vec{\rho}), \quad \rho \sin(\phi) = Z_2(\vec{\rho})$$

Using these relations in Equations 2.10 gives:

$$\begin{aligned} W(\vec{\rho}) = {} & \frac{c_x}{3} Z_6(\vec{\rho}) + \frac{c_y}{3} Z_7(\vec{\rho}) + a_x Z_4(\vec{\rho}) + a_y Z_5(\vec{\rho}) + d Z_3(\vec{\rho}) \\ & + \left(t_x - \frac{2}{3} c_x \right) Z_1(\vec{\rho}) + \left(t_y - \frac{2}{3} c_y \right) Z_2(\vec{\rho}) + \frac{1}{2} d \end{aligned} \tag{2.11}$$

Thus, for Zernike coefficients z_i defined by $W(\vec{\rho}) = \sum_i z_i Z_i(\vec{\rho})$, we have the following relations:

$$z_0 = \frac{d}{2} \qquad z_4 = a_x$$

$$z_1 = t_x - \frac{2}{3} c_x \qquad z_5 = a_y$$

$$z_2 = t_y - \frac{2}{3} c_y \qquad z_6 = \frac{c_x}{3}$$

$$z_3 = d \qquad z_7 = \frac{c_y}{3}$$

Chapter 3

Algorithms

3.1 Computing Images

To compute an image from a set of aberration parameters I've used to following algorithm, looping over all pixels:

1. Starting with a set of aberration parameters and an image point $\vec{x}_i$, compute the associated defocus-only point by solving the set of Equations 2.7 using the Newton's method routine published in *Numerical Recipes*. [8] Since the distortion in the images we are concerned with is small, the initial guess can be taken to simply be the image point $\vec{x}_i$. Equations 2.7 can, in principle, have 0, 1, 2, 3, or 4 solutions. But, since the aberration parameters are small, the conic sections in Equations 2.7 are predominately linear and we can reliably assume that we will only have either zero or one solution. Thus, if the solver fails to converge, it is because no solution exists and the image pixel is non-illuminated, so its brightness is simply set to zero. If the solver does converges, assume that it is the only solution and continue.

2. Use $\vec{x}_{df}$ to evaluate Equation 2.9. The two main components of this computation are the illuminated fraction $f(\vec{x}_{df})$ and the inverse Hessian determinant, formulas for which can be found in Appendix A.

3. Convolve the unblurred image using the kernel given in Equation 2.4.

Six examples computed with this routine are given in Figure 3-1.

3.2 Fitting for Aberrations

The pairs of defocused images are fit using a Levenberg-Marquardt modified from *Numerical Recipes'* published version. [8] In order to maximize speed, all of the derivatives are computed analytically using the formulas given in Appendix A. The fit contains fifteen total parameters: x-coma, y-coma, x-astigmatism, y-astigmatism, and the seeing radius are fit simultaneously using both images, while x-tilt, y-tilt, defocus, total flux, and the background level are fit separately for each image in the pair. These five latter parameters are fit separately since, as the images in an out-of-focus pair are recorded on different detectors, [2] their tilts, total fluxes, and background levels are likely different. Further, the defocus parameters are certainly different, as the detectors have been intentionally defocused to opposite sides of the focal plane.

The fit proceeds as follows:

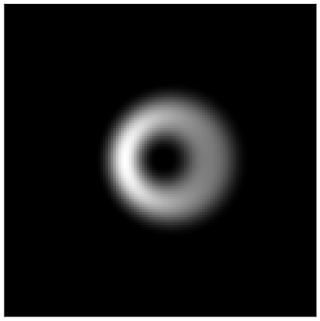
1. Initial guesses are computed, as described in Section 3.3.

2. The Levenberg-Marquardt routine fits the images without fitting for the seeing radius.

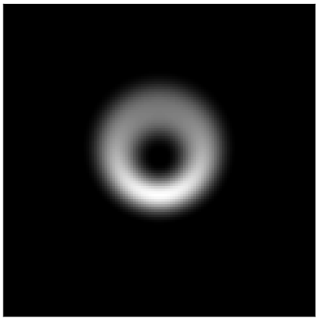
3. The results of the previous fit are used as initial guesses for another Levenberg-Marquardt fit, this time with the seeing radius included. This gives the final parameter values.

3.3 Initial Guesses

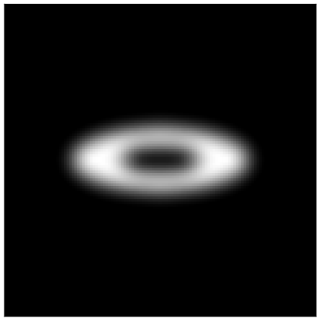
Initial guesses are generated for the fitting routine by first making the assumption that the coma and astigmatism parameters are zero. This is a reasonable assumption, as the maximum coma or astigmatism in the VISTA data is about 10 microns of wavefront error. Assuming only defocus and tilt are present, the aberrations can be estimated by iterating several image moments.



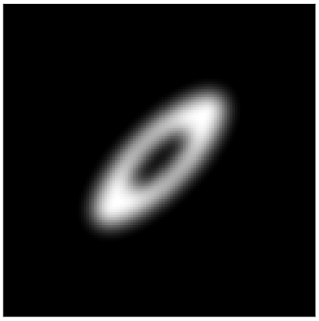
(a) x-coma, $c_x = 4$ um



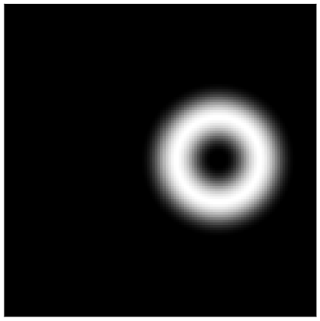
(b) y-coma, $c_y = 4$ um



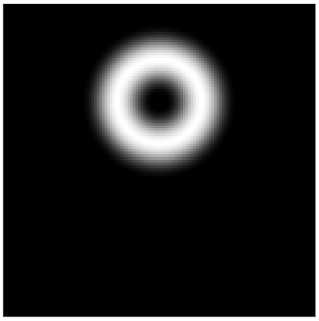
(c) x-astigmatisms, $a_x = 10$ um



(d) y-astigmatism, $a_y = 10$ um



(e) x-tilt, $t_x = 60$ um



(f) y-tilt, $t_y = 60$ um

Figure 3-1: Defocused images exhibiting $d \approx 600$ μm and exactly one of the other aberration parameters nonzero, with 1 arcsec blurring. The aberration parameters in these images are very large so as to clearly exhibit the types of distortion that they produce. In practice, the images analyzed by the fitting routine are much less distorted.

Suppose that we have estimates for the centroid of the image $t_{x,n}$ and $t_{y,n}$, the constant background value b_n, and the radial size of the star Δ_n, we can then update these estimates by computing the following moments:

$$t_{x,n+1} = \langle x - t_{x,n} \rangle_{\Delta_n, b_n}$$

$$t_{y,n+1} = \langle y - t_{y,n} \rangle_{\Delta_n, b_n}$$

$$\Delta^2_{n+1} = \langle (x - t_{x,n})^2 + (y - t_{y,n})^2 \rangle_{\Delta_n, b_n}$$

$$b_{n+1} = \frac{\sum (I(x,y) - b_n) \left(1 - e^{-\tilde{r}^2/\Delta_n^2}\right)}{\sum (1 - e^{-\tilde{r}^2/\Delta_n^2})}$$

where $\tilde{x} = x - t_{x,n}$, $\tilde{y} = y - t_{y,n}$, $\tilde{r} = \sqrt{\tilde{x}^2 + \tilde{y}^2}$, and the moment $\langle f(x,y) \rangle_{\Delta,b}$ is defined to be:

$$\langle f(x,y) \rangle_{\Delta,b} = \frac{\sum f(x,y) \, (I(x,y) - b) \, e^{-\left(x^2+y^2\right)/\Delta^2}}{\sum (I(x,y) - b) \, e^{-(x^2+y^2)/\Delta^2}}$$

This procedure is iterated until the change in the centroid values and the change in Δ is less than one pixel. Denoting the final values of this iteration with a subscript f, the initial guesses for defocus and total flux are set to be:

$$d = \sqrt{\frac{\langle (x - t_{x,f})^2 + (y - t_{y,f})^2 \rangle_{\Delta_f, b_f}}{2 \left(R^2_{out} + R^2_{in}\right)}}$$

$$F = \frac{\sum (I(x,y) - b_f) \, L^2}{\pi \left(R^2_{out} - R^2_{in}\right)}$$

where R_{out} and R_{in} are the telescope pupil radii and L is the pixel width. Finally, the tilt and background values are simply set to be the final values of the above iteration, and the coma and astigmatism guesses are set to zero.

Chapter 4

Performance of the Fit Routine

4.1 Simulated Data

Producing Simulated Data

Simulated data was produced using the algorithm given in Section 3.1. This data exactly follows our model, and so there should be rapid convergence to the exact parameter values. The images measure 120 by 120 pixels and were produced using values for the pixel width, telescope radii and defocus parameter to match those of the VISTA low-order wavefront sensor, with a focal length of 12.072 m, inner and outer pupil radii of 0.8251 m and 1.85 m, respectively, and a defocus of about 280 microns of wavefront error, which corresponds to a 1.0 mm displacement from the focal plane. [2] Test images were generated with all possible combinations of zero and nonzero aberration parameters, with the values of the nonzero parameters chosen randomly from a predetermined range of values. The range of values for each aberration parameter was chosen empirically such that aberration values within the range produced roughly one or two pixels of distortion in the image, which is what is seen in the VISTA data sample.

Fits to Simulated Data

The fitting algorithm described in Section 3.2 was tested on simulated data, and the resulting fits converged rapidly and accurately. Initial guesses were generated using the routine in Section 3.3. After fitting a random sample of 500 simulated images, the longest convergence time was 20 iterations with an average of 14.53 iterations, which was about 520 ms, to convergence. The relative error between the randomly generated parameter values and the best fit values was on average $9.17 \cdot 10^{-7}$, and the worst-case fit value from the entire set of images had a relative error of $4.83 \cdot 10^{-6}$.

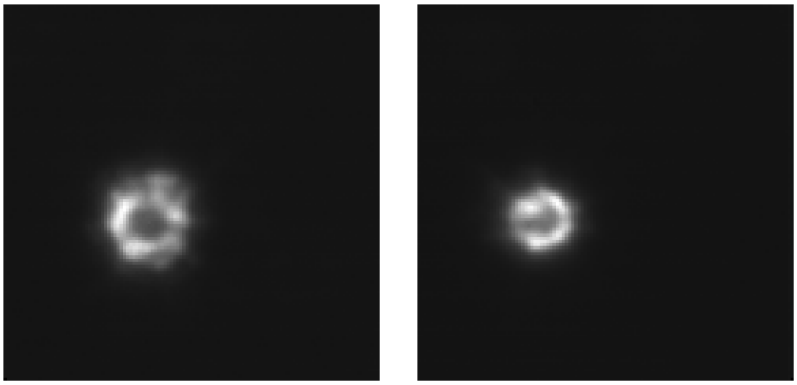
4.2 VISTA Data

Fit Speed and Quality

The fitting procedure was tested on 10 nights worth of VISTA wavefront-sensing data provided by the European Southern Observatory. [1] An example image from this dataset, as well as a best-fit image and residual, is given in Figure 4-1. This image and the resulting fit are very typical for the VISTA dataset. The fit routine has matched the large-scale size and shape of the image well, however there remains small-scale variations in the brightness internal to the star. This is likely due to the presence of unaccounted for higher-order terms in Equation 2.3. This causes a poor reduced chi-squared, $\chi^2_{red} = 8.57$. The fit was rapid, however, converging in 26 total iterations, with 21 belonging to the initial unblurred fit and 5 to the blurred fit. These are typical values; over the entire dataset, the average reduced chi-squared was 5.73 and the average total number of iterations was 34. The best-fit values of the aberration parameters for one day (about 70 images) of the full ten day dataset (700 images) are given in the following plots, and any notable features of the data are discussed in the captions.

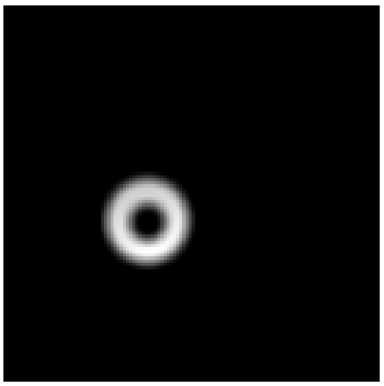
[1] We thank Thomas Szeifert, Andreas Kaufer and Magda Arnaboldi for arranging for the VISTA data to be recorded on our behalf.

Figure 4-1: An example fit to VISTA data. While the large-scale fit matches the images overall shape and size well, there are clear higher-order variations in the residual that the model does not account for. Also, note the very high intensity in the "hole" of the annulus on the residual image; the cause of this is currently unknown. This drives up the χ^2_{red} value to 8.6. The χ^2_{red} was computed using the square root of the photon count as the error. For this fit, the number of iterations before convergence was 26. The fit was taken over the entire 120x120 image, all of which is pictured.

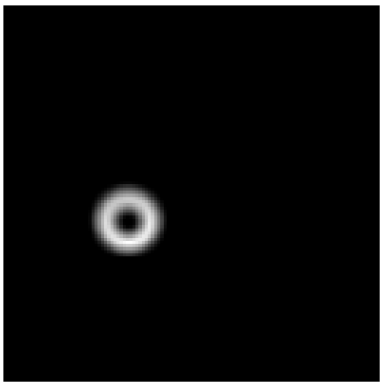


(a) Data, Positive Defocus

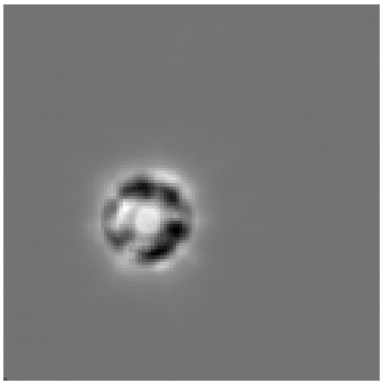
(b) Data, Negative Defocus



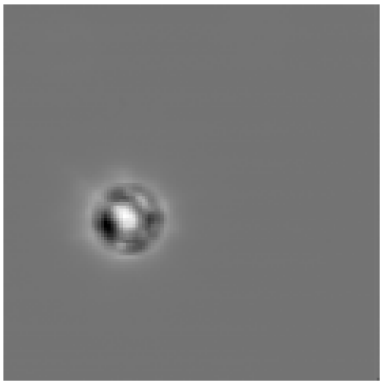
(c) Best Fit, Positive Defocus



(d) Best Fit, Negative Defocus



(e) Residual, Positive Defocus



(f) Residual, Negative Defocus

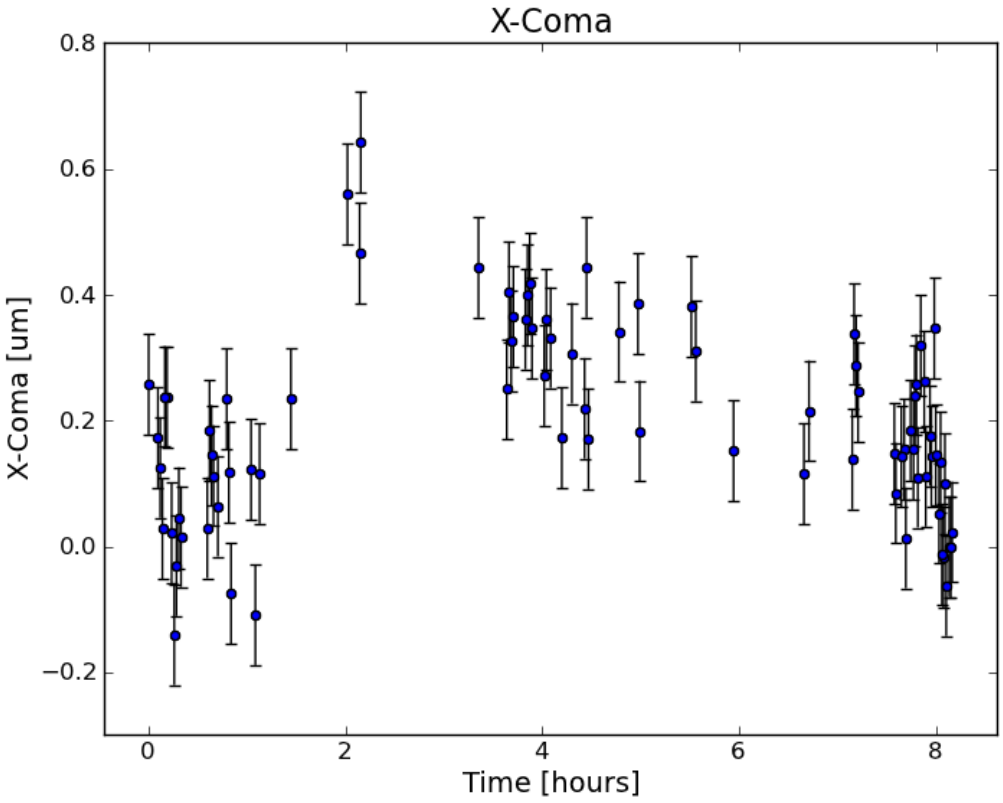


Figure 4-2: This is the best fit x-coma, plotted over time for 1 night of observing. This time period correspond to about $t = 4.6$ days on the 10-day plots. The error bars are taken to be the median discrepancy taken from Table 4.2.

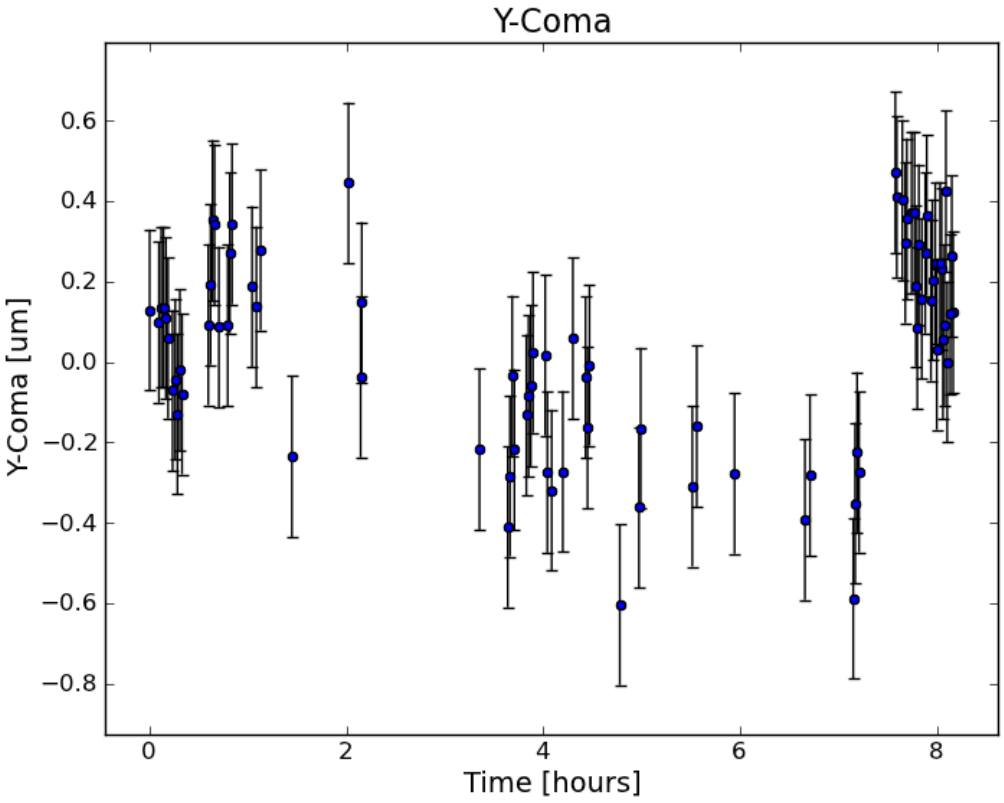


Figure 4-3: This is the best fit y-coma, plotted over time for 1 night of observing. This time period correspond to about $t = 4.6$ days on the 10-day plots. The error bars are taken to be the median discrepancy taken from Table 4.2.

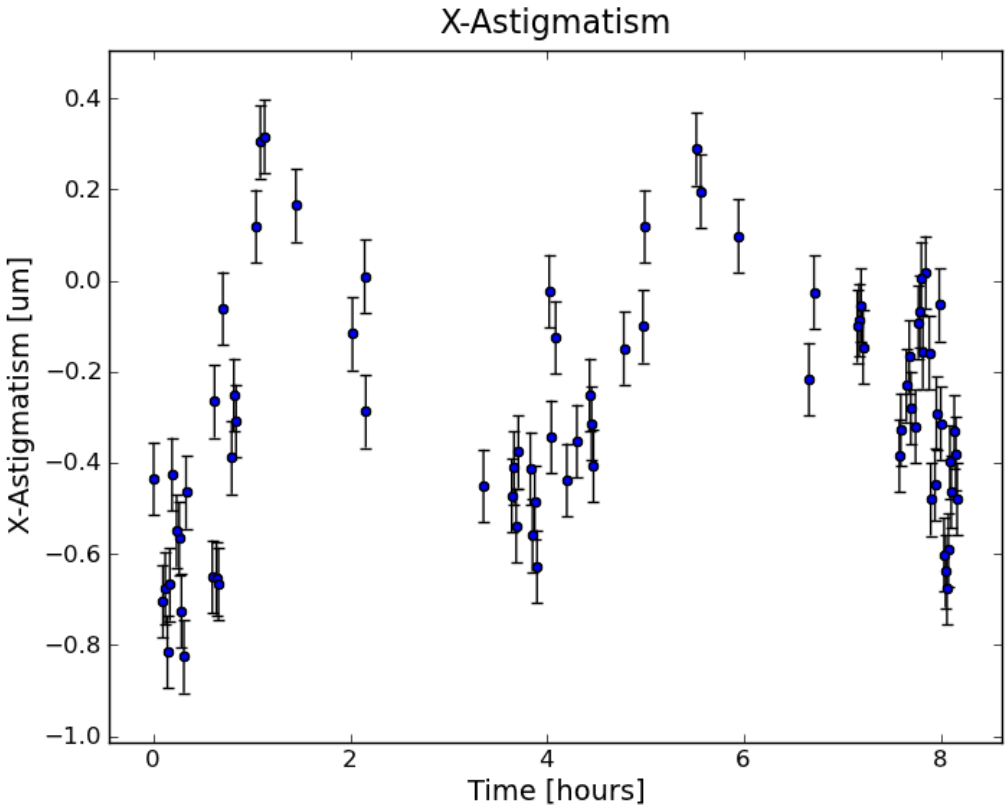


Figure 4-4: This is the best fit x-astigmatism, plotted over time for 1 night of observing. This time period correspond to about $t = 4.6$ days on the 10-day plots. The error bars are taken to be the median discrepancy taken from Table 4.2.

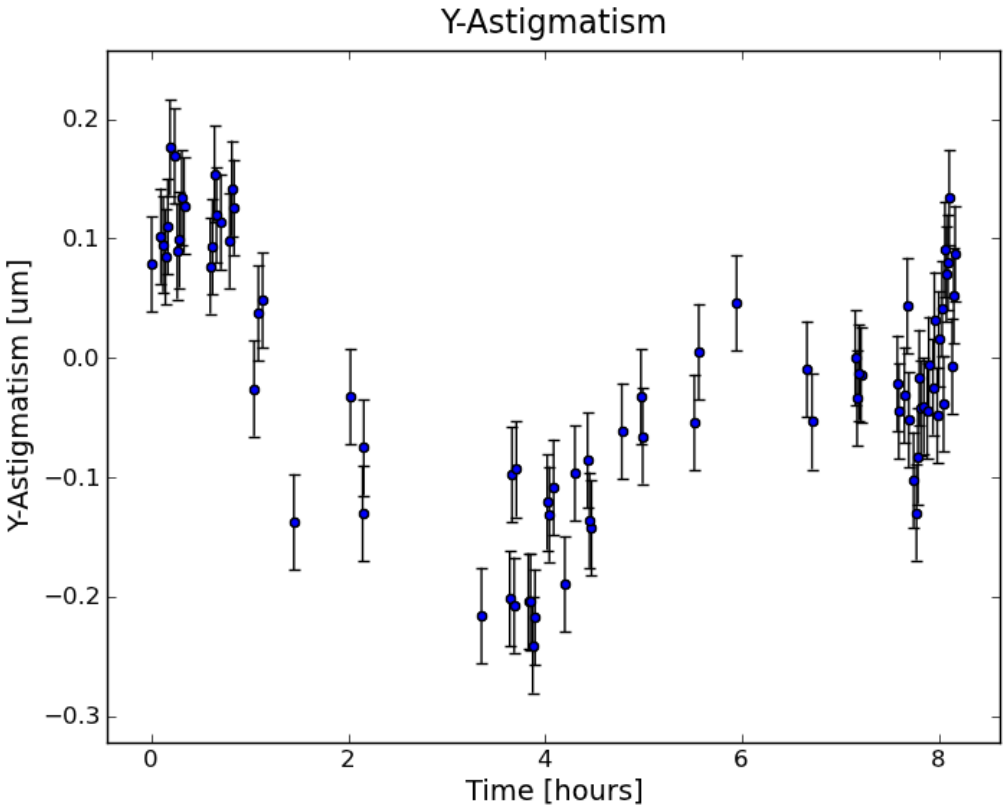


Figure 4-5: This is the best fit y-astigmatism, plotted over time for 1 night of observing. This time period correspond to about $t = 4.6$ days on the 10-day plots. The error bars are taken to be the median discrepancy taken from Table 4.2.

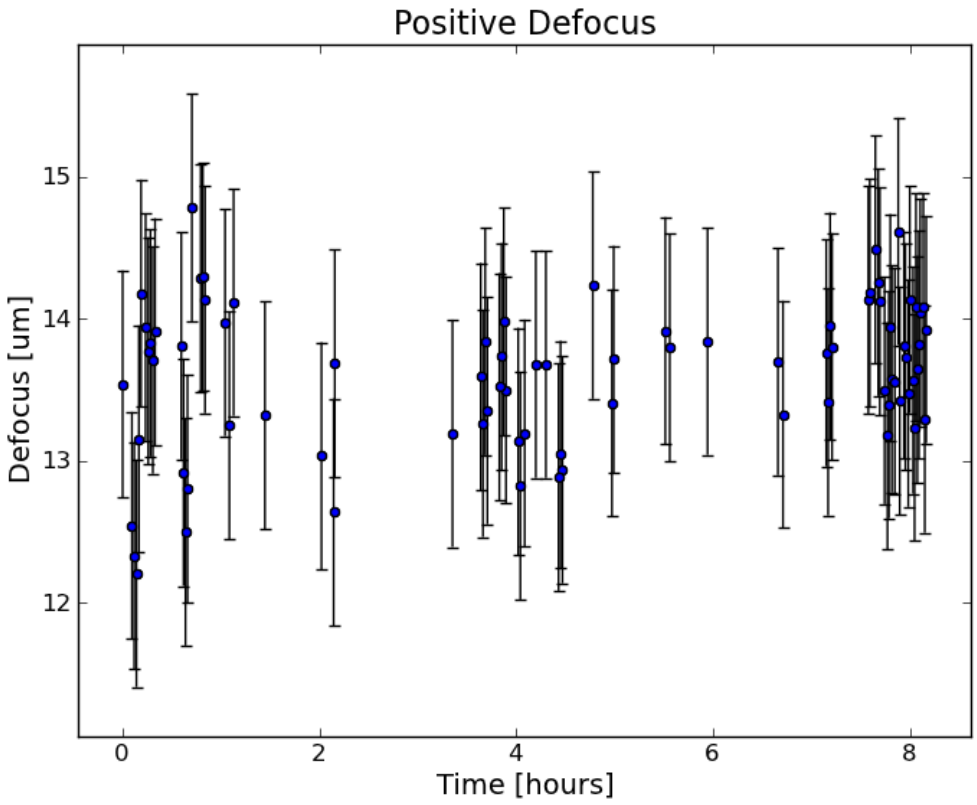


Figure 4-6: This is the best fit defocus for the pre-focal image, plotted over time for 1 night of observing. This time period correspond to about $t = 4.6$ days on the 10-day plots. The error bars are taken to be the median discrepancy taken from Table 4.2.

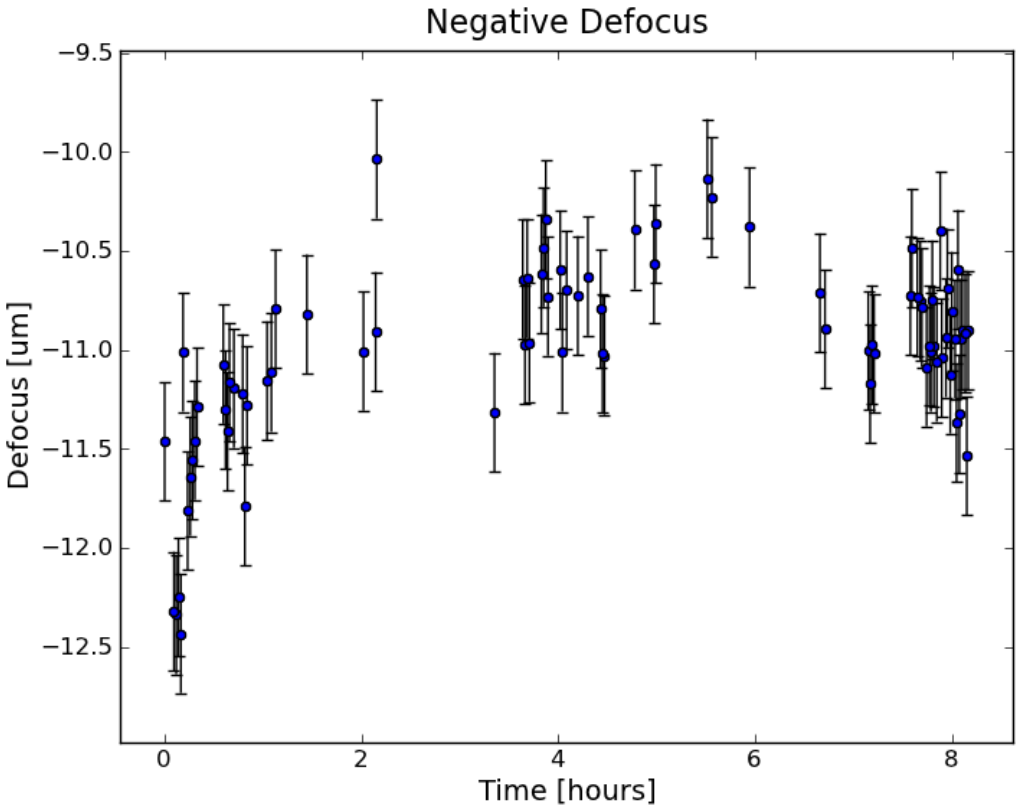


Figure 4-7: This is the best fit defocus for the post-focal image, plotted over time for 1 night of observing. This time period correspond to about $t = 4.6$ days on the 10-day plots. The error bars are taken to be the median discrepancy taken from Table 4.2.

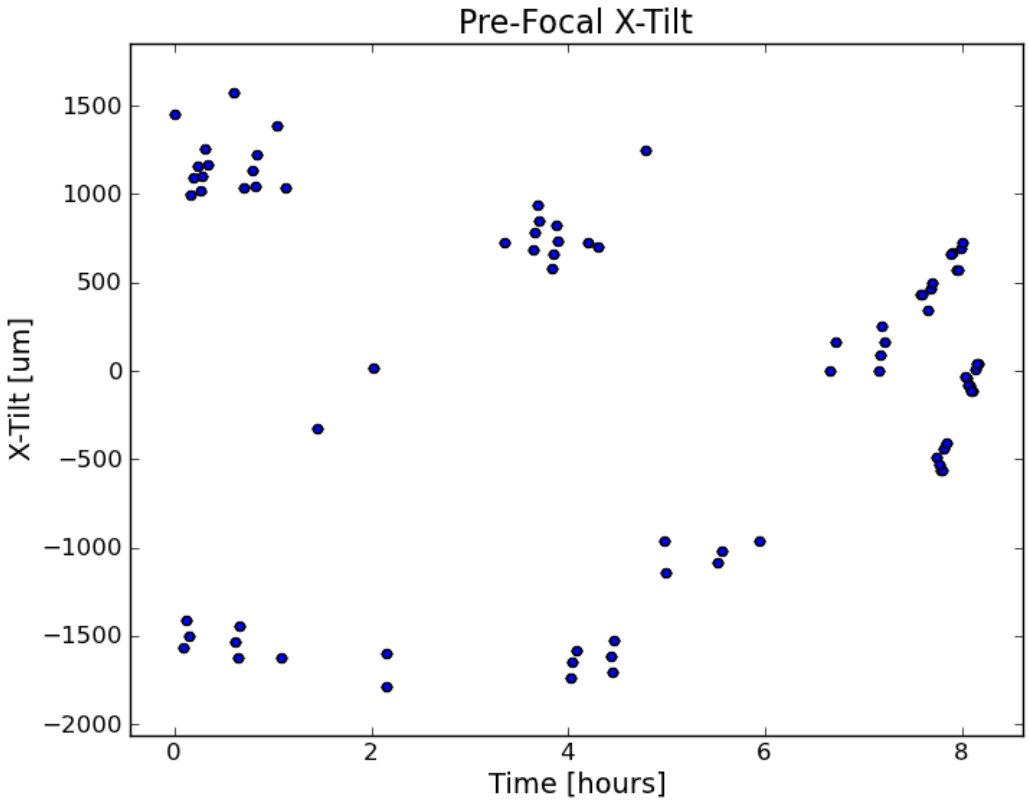


Figure 4-8: This is the best fit x-tilt for the pre-focal image, plotted over time for 1 night of observing. This time period correspond to about $t = 4.6$ days on the 10-day plots. The error bars are taken to be the median discrepancy taken from Table 4.2. Note the clumping of the data; this corresponds to the 6 dithers used by VISTA for a given pointing. [2]

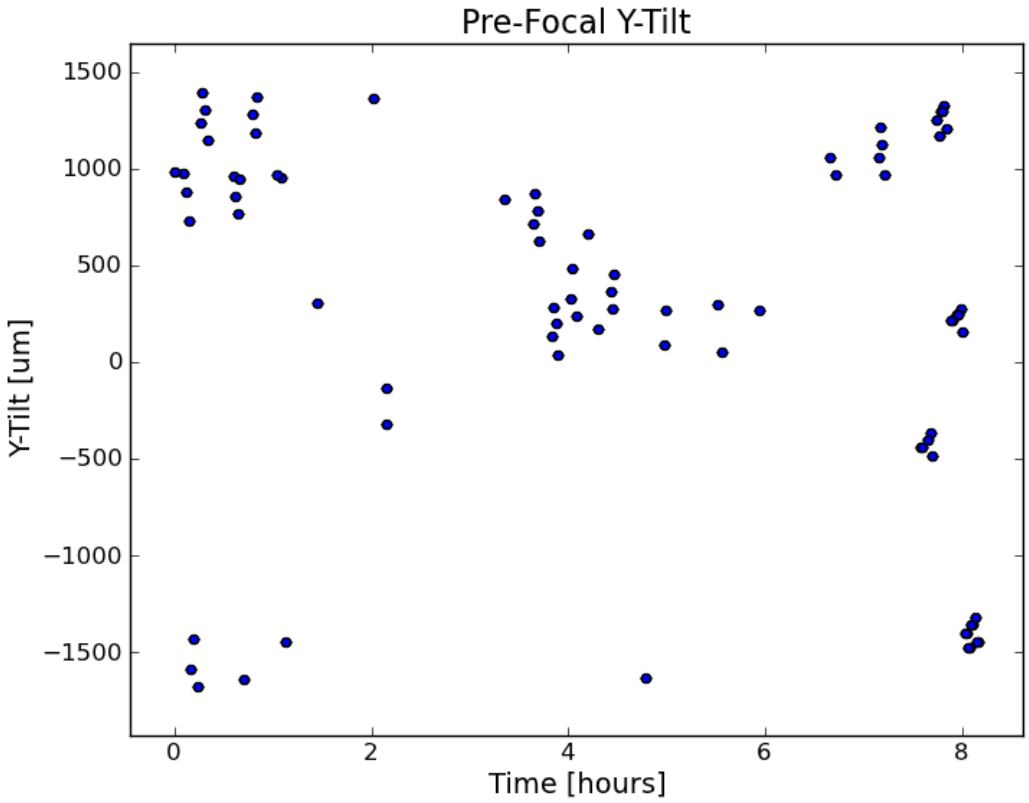


Figure 4-9: This is the best fit y-tilt for the pre-focal image, plotted over time for 1 night of observing. This time period correspond to about $t = 4.6$ days on the 10-day plots. The error bars are taken to be the median discrepancy taken from Table 4.2. Note the clumping of the data; this corresponds to the 6 dithers used by VISTA for a given pointing. [2]

Fit Errors

Since the χ^2_{red} values for these fits are poor, the error estimates produced by the fit routine are not reliable and underestimate the error by a factor of 4 to 8, when compared to the errors given in Table 4.2. Instead, I will estimate the error by looking at the variation in the fit values extracted from very similar images. The VISTA data consists of 120x120 pixel pieces taken from a 2Kx2K detector, and the header includes the pixel coordinates of the lower-left pixel of the 120x120 image. From the dataset, I selected images for which this pixel was located in exactly the same spot on the full detector, and further, from this set I selected all of the pairs of images that were observed consecutively. Generally, these consecutive observations were separated by an interval of about 50 seconds, and had an integration time of 25.0 seconds. Since these images were taken at nearly the same time, and the star did not change positions on the detector between the images, it is reasonable to assume that the star in both images is the same, and that the telescope did not move appreciably between the images. Thus, the difference in aberrations should be very small and any discrepancy between the fitted aberration values for these images can be attributed to the fitting routine. The maximum and median of the magnitude of this discrepancy, taken over all such pairs of images, is reported in Table 4.2 and Table 4.2. The median values can be taken to give a typical uncertainty and the maximum values give the worst-case uncertainty.

Another way to estimate the fitting errors is to consider the variation in time of combinations of parameters that should be constant. The defocus parameter for each image in a pair consists of a component due to the alignment state of the telescope and another component due to the position of the wavefront sensor. The wavefront sensor components for the VISTA images should be nearly equal in magnitude but of opposite sign and it should remain fixed in time, whereas the telescope components will be equal for each image and varying with time. Thus, the difference of these two values will be constant. Any variation in this quantity, which is plotted in Figure 4.2, should be attributable to the uncertainties of our measurement. From the figure, it is

Table 4.1: This table gives statistics of the magnitudes of the discrepancies of fit values extracted from very similar images. Aberration units are in microns of wavefront error, flux and background are given in photon counts, and the seeing diameter is in arc seconds. The median value gives a typical error bound, and the maximum values an upper error bound. Note the very small discrepancies for the total flux parameters, seeing diameters, and background level (the absolute scale of the background is about 750 photon counts), which affirms that we are looking at the same star in both images. Also, the values for y-coma are about 3 times larger than the other coma and astigmatism parameters. The cause of this is not yet understood, however it warns that the reported error for x-coma is likely too small. Finally, note that the astigmatism parameters differ by a factor of 2; the cause of this is also not currently understood.

Parameter	Minimum	Median	Maximum
Coma x	0.0009207	0.07955	0.523
Coma y	0.001261	0.179	1.413
Astig x	0.01432	0.08325	0.4041
Astig y	0.002016	0.04104	0.1978
Seeing Diameter	0.00669	0.1539	1.37
Tilt x+	0.007977	0.6695	4.725
Tilt y+	0.07267	0.5656	2.57
Defocus+	0.0102	0.2826	2.286
Flux+	2.487e-08	4.168e-07	2.865e-06
Bkgnd+	0.0494	4.688	43.53
Tilt x-	0.02365	0.8693	5.39
Tilt y-	0.01017	0.7276	2.593
Defocus-	0.02416	0.2809	2.286
Flux-	2.723e-08	4.176e-07	2.147e-06
Bkgnd-	0.2406	5.331	48.72

clear that the scatter is consistent with the uncertainty estimates given above, which lends more credibility to the estimates given in Table 4.2. The same reasoning applies to the x and y components of tilt, whose difference is plotted in Figures 4.2 and 4.2. These plots are less consistent than the defocus difference, but not alarming so.

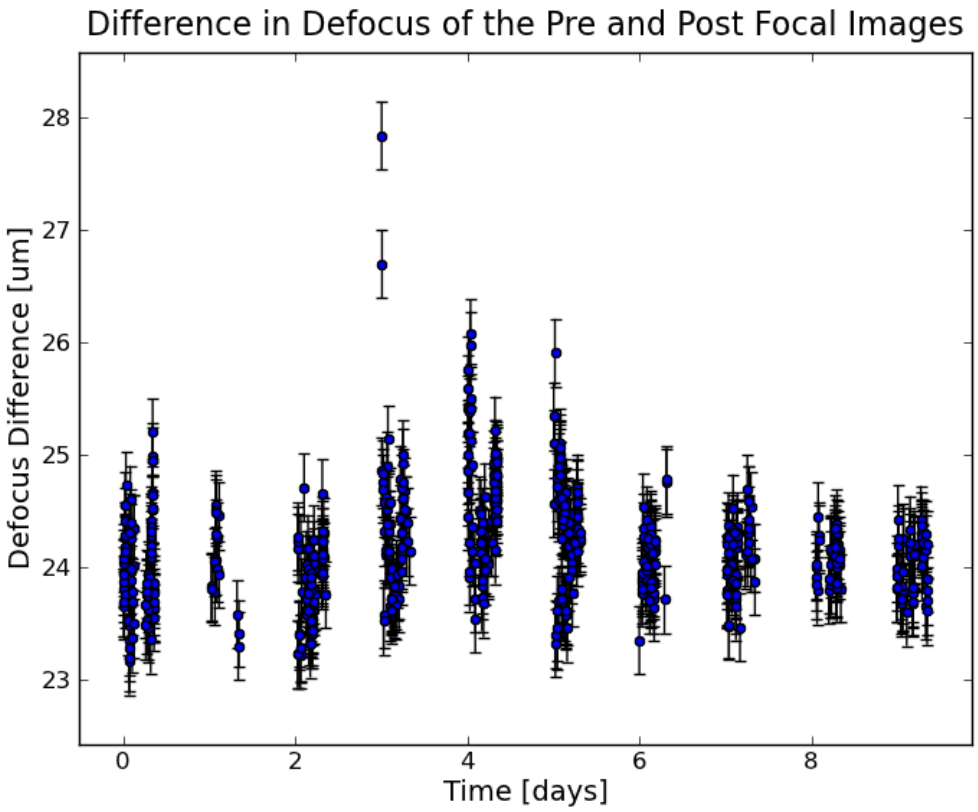


Figure 4-10: A plot of the defocus difference, which in principle should be constant in time. I have plotted this data with error bars set the median discrepancy taken from Table 4.2, in order to show that the scatter over time in the defocus difference is consistent with the median errors from Table 4.2. While there are a few obvious outliers, the data is clearly consistent with a constant function. The units of defocus are given in microns of wavefront error.

Table 4.2: These are the same results as in Table 4.2, but have been converted to errors in the Zernike coefficients using the conversions in Section 2.5. All of the values are given in microns of wavefront error.

Zernike Number	Minimum	Median	Maximum
1	0.04304	0.7554	2.675
2	0.07267	0.6947	2.574
3	0.02416	0.2809	2.286
4	0.01432	0.08325	0.4041
5	0.002016	0.04104	0.1978
6	0.0003069	0.02652	0.1743
7	0.0004204	0.05968	0.4709

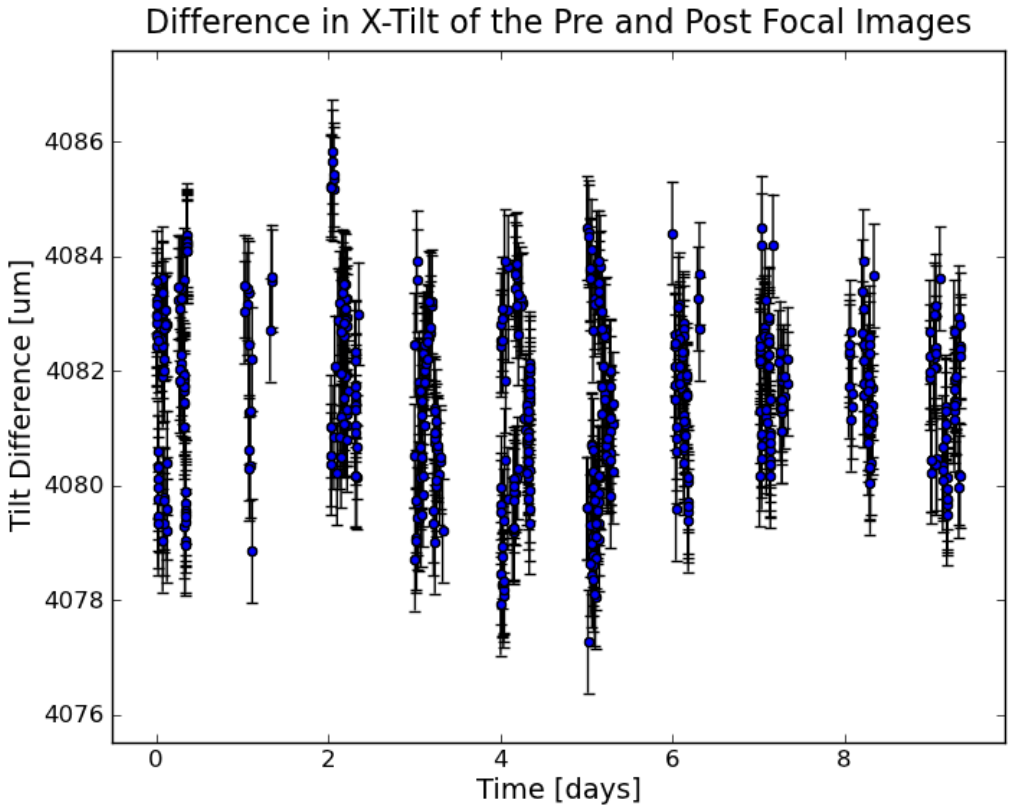


Figure 4-11: A plot of the x-tilt difference, which in principle should be constant in time. I have plotted this data with error bars set the median discrepancy taken from Table 4.2. While the the data is not rigorously consistent with a constant function, it is very close. The units of defocus are given in microns of wavefront error.

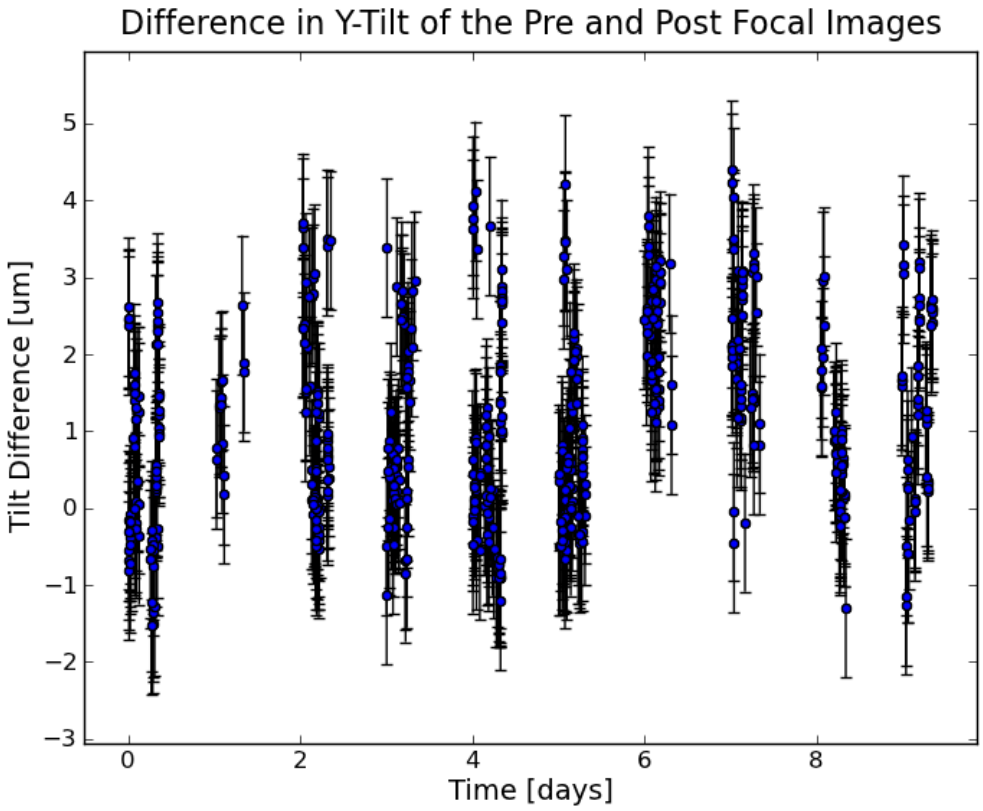


Figure 4-12: A plot of the y-tilt difference, which in principle should be constant in time. I have plotted this data with error bars set the median discrepancy taken from Table 4.2. As with the x-tilt difference (Figure 4.2), the the data is not rigorously consistent with a constant, it is very close The units of defocus are given in microns of wavefront error.

Chapter 5

Conclusion

In this work I have described a new algorithm for extracting measurements of optical aberrations from a pair of oppositely defocused images. The model was verified using simulated data, and also applied with some success to VISTA data. The VISTA fits were able to fit the data rapidly and provided a consistent estimate of the fitting errors, at a reasonable scale of 0.08 um for coma, 0.08 um for astigmatism, 0.9 um for tilt, and 0.3 um for defocus. However, but the model is also clearly unable to account for higher-order aberrations, and probably other effects as well, that are present in the VISTA data. As such, there is certainly room for improvement, and expanding the model from its present state to account for at least some subset of these currently un-modeled effects will likely result in a much more accurate determination of the aberrations and thus a more applicable model.

Appendix A

Formulas

In this section I will give the exact formulas that are used in the software implementations of the algorithms given in Chapter 3. In most cases, these results follow directly from the definitions in Chapter 2 and some tedious but straight-forward computation, in which case I will simply report the final result. However, there is some subtlety in a few of these computations, and for those cases I provide a derivation.

A.1 Illuminated Fraction

To find the illuminated fraction of a defocus-only pixel, we need to solve the following problem: Given a square pixel of length l centered on the point (x_{df}, y_{df}), what fraction ϕ of its area is enclosed by a circle of radius R centered on the origin? The illuminated fraction f is given by the enclosed fraction $\phi(\vec{x}_{df}, R)$ as:

$$f(\vec{x}_{df}) = \begin{cases} \phi(\vec{x}_{df}, 2dR_{out}) & |r_{df} - 2dR_{out}| < \frac{l}{\sqrt{2}} \\ 1 - \phi(\vec{x}_{df}, 2dR_{in}) & |r_{df} - 2dR_{in}| < \frac{l}{\sqrt{2}} \\ 0 & \text{Otherwise} \end{cases} \tag{A.1}$$

where $r_{df} = \sqrt{x_{df}^2 + y_{df}^2}$. In this formula, I have assumed that the three cases are mutually exclusive, which will be true in any realistic telescope.

To determine ϕ, note first that because of circular symmetry, the enclosed frac-

tion of the pixel will be invariant if we reflect the pixel across any line through the origin. Thus, I will restrict attention to pixels with coordinates $0 \leq x_{df} \leq y_{df}$, which significantly reduces the number of special cases that we need to consider. Second, in any realistic telescope we will have that $R \gg l$, and so I will approximate the circle by its tangent line in the vicinity of (x_{df}, y_{df}). The tangent is given by

$$y = -\frac{x_{df}}{y_{df}}x + \frac{r_{df}}{y_{df}}R \tag{A.2}$$

There are five special cases to consider, two of which are trivial:

1. The pixel is inside the circle: $\phi = 1$.
2. The pixel is outside the circle: $\phi = 0$.
3. The circle crosses the left side and the bottom of the pixel.
4. The circle crosses both the left and right sides of the pixel.
5. The circle crosses the top and right side of the pixel.

We need a computational way of differentiating between these cases. To do this, consider the quantity y_*, which gives the y-coordinate at which the tangent line intersects the vertical line coincident with the right edge of the pixel:

$$y_* = -\frac{x_{df}}{y_{df}}(x_{df} + l/2) + \frac{r_{df}}{y_{df}}R \tag{A.3}$$

This expression can be made easier to work with by defining a dimensionless quantity γ, which gives the distance between the top of the pixel and the intersection point y_* in units of the pixel length:

$$\gamma = \frac{1}{l}\left(y_{df} + \frac{l}{2} - y_*\right) \tag{A.4}$$

Then the above conditions become:

1. The pixel is inside the circle: $\gamma < 0$

2. The pixel is outside the circle: $\gamma > 1 + x_{df}/y_{df}$

3. The circle crosses the left side and the bottom of the pixel: $1 < \gamma < 1 + x_{df}/y_{df}$

4. The circle crosses both the left and right sides of the pixel: $x_{df}/y_{df} < \gamma < 1$

5. The circle crosses the top and right side of the pixel: $0 < \gamma < x_{df}/y_{df}$

All that remains is to calculate the fractional area of the pixel that lies below the tangent line. This is straightforward geometry, and the results are:

1. The pixel is inside the circle:

$$\phi = 1$$

2. The pixel is outside the circle:

$$\phi = 0$$

3. The circle crosses the left side and the bottom of the pixel:

$$\phi = \frac{x_{df}}{2y_{df}} \left(1 - (\gamma - 1)\frac{x_{df}}{y_{df}}\right)^2$$

4. The circle crosses both the left and right sides of the pixel:

$$\phi = 1 - \gamma + \frac{x_{df}}{2y_{df}}$$

5. The circle crosses the top and right side of the pixel:

$$\phi = 1 - \frac{1}{2}\frac{y_{df}}{x_{df}}\gamma^2$$

A.2 Derivative of Illuminated Fraction

Directly from Equation A.1, the derivative of the illuminated fraction with respect to some parameter α is:

$$\frac{\mathrm{d}f}{\mathrm{d}\alpha} = \begin{cases} \frac{\mathrm{d}\phi}{\mathrm{d}\alpha} & |r_{df} - 2dR_{out}| < \frac{l}{\sqrt{2}} \\ -\frac{\mathrm{d}\phi}{\mathrm{d}\alpha} & |r_{df} - 2dR_{in}| < \frac{l}{\sqrt{2}} \\ 0 & \text{Otherwise} \end{cases}$$

The enclosed fraction ϕ is a function of three variables: the defocus-only pixel coordinates x_{df} and y_{df}, and the radius of the circle. The derivative with respect to α is therefore:

$$\frac{\mathrm{d}\phi}{\mathrm{d}\alpha} = \frac{\partial\phi}{\partial x_{df}}\frac{\partial x_{df}}{\partial\alpha} + \frac{\partial\phi}{\partial y_{df}}\frac{\partial y_{df}}{\partial\alpha} + \frac{\partial\phi}{\partial R}\frac{\partial R}{\partial\alpha} \tag{A.5}$$

The defocus-only position partials are computed in Section A.5. The three partial derivatives of ϕ appearing in Equation A.5 are given by simply differentiating the expressions for ϕ derived in Section A.1. These are most clearly expressed as functions of x_{df}, y_{df}, and partial derivatives of the parameter γ with respect to defocus-only-positions. This gives the formulas:

1. The pixel is inside the circle:

$$\frac{\partial\phi}{\partial x_{df}} = \frac{\partial\phi}{\partial y_{df}} = \frac{\partial\phi}{\partial R}\frac{\partial R}{\partial\alpha} = 0$$

2. The pixel is outside the circle:

$$\frac{\partial\phi}{\partial x_{df}} = \frac{\partial\phi}{\partial y_{df}} = \frac{\partial\phi}{\partial R}\frac{\partial R}{\partial\alpha} = 0$$

3. The circle crosses the left side and the bottom of the pixel:

$$\frac{\partial \phi}{\partial x_{df}} = \frac{1}{2y_{df}} \left(1 - \left[(\gamma - 1)\frac{y_{df}}{x_{df}}\right]^2\right) - \left(1 - (\gamma - 1)\frac{y_{df}}{x_{df}}\right)\frac{\partial \gamma}{\partial x_{df}}$$
$$\frac{\partial \phi}{\partial y_{df}} = -\frac{x_{df}}{2y_{df}^2} \left(1 - \left[(\gamma - 1)\frac{y_{df}}{x_{df}}\right]^2\right) - \left(1 - (\gamma - 1)\frac{y_{df}}{x_{df}}\right)\frac{\partial \gamma}{\partial y_{df}}$$
$$\frac{\partial \phi}{\partial R}\frac{\partial R}{\partial \alpha} = \delta_{\alpha,d}\left(1 - (\gamma - 1)\frac{y_{df}}{x_{df}}\right)\frac{r_{df}}{y_{df}}\frac{R}{ld}$$

4. The circle crosses both the left and right sides of the pixel:

$$\frac{\partial \phi}{\partial x_{df}} = \frac{1}{2y_{df}} - \frac{\partial \gamma}{\partial x_{df}}$$
$$\frac{\partial \phi}{\partial y_{df}} = -\frac{x}{2y_{df}^2} - \frac{\partial \gamma}{\partial y_{df}}$$
$$\frac{\partial \phi}{\partial R}\frac{\partial R}{\partial \alpha} = \delta_{\alpha,d}\frac{r_{df}}{y_{df}}\frac{R}{ld}$$

5. The circle crosses the top and right side of the pixel:

$$\frac{\partial \phi}{\partial x_{df}} = -\frac{y_{df}}{x_{df}}\gamma\left(\frac{\partial \gamma}{\partial x_{df}} - \frac{\gamma}{2x_{df}}\right)$$
$$\frac{\partial \phi}{\partial y_{df}} = -\frac{y_{df}}{x_{df}}\gamma\left(\frac{\partial \gamma}{\partial y_{df}} + \frac{\gamma}{2y_{df}}\right)$$
$$\frac{\partial \phi}{\partial R}\frac{\partial R}{\partial \alpha} = \delta_{\alpha,d}\gamma\frac{r_{df}}{x_{df}}\frac{R}{ld}$$

where:

$$\frac{\partial \gamma}{\partial x_{df}} = \frac{1}{y_{df}}\left[\frac{1}{2} + \frac{x}{l}\left(2 - \frac{R}{r_{df}}\right)\right]$$
$$\frac{\partial \gamma}{\partial y_{df}} = \frac{1}{y_{df}^2}\left[\left(\frac{R - r_{df}}{l}\right)^2 - \frac{x}{2}\right] + \frac{1}{l}\left(2 - \frac{R}{r_{df}}\right)$$

A.3 Hessian Matrix

The Hessian matrix H is the Jacobian of the transformation in Equation 2.7. This is given by:

$$H = \begin{pmatrix} \frac{3C_x}{2d^2}x_{df} + \frac{C_y}{2d^2}y_{df} + \frac{A_x}{d} + 1 & \frac{C_y}{2d^2}x_{df} + \frac{C_x}{2d^2}y_{df} + \frac{A_y}{d} \\ \frac{C_y}{2d^2}x_{df} + \frac{C_x}{2d^2}y_{df} + \frac{A_y}{d} & \frac{C_x}{2d^2}x_{df} + \frac{3C_y}{2d^2}y_{df} - \frac{A_x}{d} + 1 \end{pmatrix}$$

It is useful to have an expression for $\det H^{-1}$, for use in computing H^{-1} and for computing the brightness. As a polynomial in the pupil position, it is given by:

$$\begin{aligned} \frac{1}{\det H} = &\frac{3C_x^2 - C_y^2}{4d^2}x_{df}^2 + \frac{3C_y^2 - C_x^2}{4d^2}y_{df}^2 + \frac{2C_xC_y}{d^4}x_{df}y_{df} \\ &+ \frac{1}{d^2}\left[C_x(2 - \frac{A_x}{d}) - \frac{C_yA_y}{d}\right]x_{df} + \frac{1}{d^2}\left[C_y(2 + \frac{A_x}{d}) - \frac{C_xA_y}{d}\right]y_{df} \\ &+ 1 - \frac{A_x^2 + A_y^2}{d^2} \end{aligned}$$

A.4 Derivative of the Brightness

In order to use this model to rapidly fit for the aberration parameters of an image, it is necessary to have an analytic expression for the gradient of a given pixel's brightness with respect to the set of aberration parameters and the total flux F.

The brightness of a pixel is given by Equation 2.9:

$$I(\vec{x}_i) = \frac{F}{4d^2}\frac{1}{|\det(H(\vec{x}_{df}))|}f(\vec{x}_{df}) + b$$

Differentiating this with respect to a parameter α gives four terms:

$$\begin{aligned} \frac{\partial I}{\partial \alpha} = &\frac{F}{4d^2}\frac{1}{|\det(H)|}\frac{\partial f}{\partial \alpha} + \frac{F}{4d^2}\frac{\partial}{\partial \alpha}\left(\frac{1}{|\det(H)|}\right)f \\ &- \delta_{\alpha,d}\frac{F}{2d^3}\frac{1}{|\det(H)|}f + \delta_{\alpha,F}\frac{1}{4d^2}\frac{1}{|\det(H)|}f + \delta_{\alpha,b} \end{aligned}$$

The determinant derivative is:

$$\begin{aligned}
\frac{\partial}{\partial\alpha}\left(\frac{1}{|\det(H)|}\right) &= -\frac{1}{|\det(H)|^2}\frac{\partial}{\partial\alpha}\left(|\det(H)|\right) \\
&= -\frac{1}{|\det(H)|^2}\frac{|\det(H)|}{\det(H)}\frac{\partial}{\partial\alpha}\left(\det(H)\right) \\
&= -\frac{1}{|\det(H)|^2}\frac{|\det(H)|}{\det(H)}\det(H)\operatorname{Tr}\left(H^{-1}\frac{\partial H}{\partial\alpha}\right) \\
&= -\frac{1}{|\det(H)|}\operatorname{Tr}\left(H^{-1}\frac{\partial H}{\partial\alpha}\right)
\end{aligned}$$

and so the full brightness derivative is:

$$\begin{aligned}
\frac{\partial I}{\partial\alpha} &= \frac{F}{4d^2}\frac{1}{|\det(H)|}\frac{\partial f}{\partial\alpha} - \frac{F}{4d^2}\frac{1}{|\det(H)|}\operatorname{Tr}\left(H^{-1}\frac{\partial H}{\partial\alpha}\right)f \\
&\quad - \delta_{\alpha,d}\frac{F}{2d^3}\frac{1}{|\det(H)|}f + \delta_{\alpha,F}\frac{1}{4d^2}\frac{1}{|\det(H)|}f + \delta_{\alpha,b} \\
&= \frac{1}{4d^2}\frac{1}{|\det(H)|}\left[\frac{\partial f}{\partial\alpha}F - \operatorname{Tr}\left(H^{-1}\frac{\partial H}{\partial\alpha}\right)Ff - \delta_{\alpha,d}\frac{2}{d}Ff + \delta_{\alpha,F}f\right] + \delta_{\alpha,b}
\end{aligned} \tag{A.6}$$

Now, if f is zero then the last two terms vanish. The term $\partial f/\partial\alpha$ will also vanish (see Equation A.2). This makes sense, as $f = 0$ corresponds to an unilluminated background pixel. In addition, F represents the total flux through the pupil, and so for any image it must be nonzero. Since we will therefore only evaluate the above expression when $f \neq 0$ and $F \neq 0$, it does no harm to factor out a factor of f and F to give the final formula:

$$\frac{\partial I}{\partial\alpha} = I\left[-\operatorname{Tr}\left(H^{-1}\frac{\partial H}{\partial\alpha}\right) + \frac{1}{f}\frac{\partial f}{\partial\alpha} + \frac{2}{d}\delta_{\alpha,d} + \frac{1}{F}\delta_{\alpha,F}\right] + \delta_{\alpha,b} \tag{A.7}$$

This is the form of the derivative that will be used by the fitting routine. Note that while there are five terms, it is rarely necessary to evaluate all of them. Of the four bracketed terms, the first encodes the change in brightness of an interior pixel and is always nonzero for illuminated pixels. The second accounts for the large change in brightness that occurs near the edge of the image due to the shifting boundaries of the illuminated region, and it is only nonzero for border pixels. The third and fourth

terms include the overall change in brightness that occurs when the defocus or total flux is varied, respectively. These terms come from the fact that the brightness of the defocus-only image changes when the total flux or defocus is varied.

A.5 Derivative of the Defocus-Only Position

To compute the derivative of the brightness of a pixel with respect to an aberration parameter, we need to know how the defocus-only position changes if we fix the image position and vary an aberration parameter. The defocus-only position and the image position are related by Equation 2.7. I will rewrite this in the more compact form:

$$\vec{x}_i = \nabla_{df} W(\vec{x}_{df}) \tag{A.8}$$

Since the image coordinates x_i and y_i are fixed, differentiating with respect to an aberration parameter α gives zero on the left-hand side. Applying the chain rule to the right-hand side gives:

$$0 = \frac{d}{d\alpha} \left[\nabla_{df} W(\vec{x}_{df})\right] \tag{A.9}$$

$$= \frac{\partial}{\partial \alpha} \left[\nabla_{df} W(\vec{x}_{df})\right] + \frac{\partial \vec{x}_{df}}{\partial \alpha} \cdot J_{df} \left(\nabla_{df} W(\vec{x}_{df})\right) \tag{A.10}$$

where $J_{df}(\vec{f}(\vec{x}_{df}))$ denotes the Jacobian matrix of $\vec{f}(\vec{x}_{df})$ with respect to $\vec{x}_{df}$. But, $J_{df} \left(\nabla_{df} W(\vec{x}_{df})\right)$ is just the Hessian $H(\vec{x}_{df})$, so:

$$0 = \frac{\partial}{\partial \alpha} \left[\nabla_{df} W(\vec{x}_{df})\right] + \frac{\partial \vec{x}_{df}}{\partial \alpha} \cdot H(\vec{x}_{df})$$

Thus the derivative of the defocus-only position with respect to α is:

$$\frac{\partial \vec{x}_{df}}{\partial \alpha} = -\frac{\partial}{\partial \alpha} \left[\nabla_{df} W(\vec{x}_{df})\right] \cdot H^{-1}(\vec{x}_{df})$$

Let $\vec{V}_\alpha = \frac{\partial}{\partial \alpha}\left[\nabla_{df} W(\vec{x}_{df})\right]$. This vector is given by:

$$\vec{V}_\alpha = -\begin{pmatrix} \frac{\partial}{\partial \alpha}\left[x_{df} + \frac{3C_x}{4d^2}x_{df}^2 + \frac{C_x}{4d^2}y_{df}^2 + \frac{C_y}{2d^2}x_{df}y_{df} + \frac{A_x}{d}x_{df} + \frac{A_y}{d}y_{df} + T_x\right] \\ \frac{\partial}{\partial \alpha}\left[y_{df} + \frac{C_y}{4d^2}x_{df}^2 + \frac{3C_y}{4d^2}y_{df}^2 + \frac{C_x}{2d^2}x_{df}y_{df} + \frac{A_y}{d}x_{df} - \frac{A_x}{d}y_{df} + T_y\right] \end{pmatrix}$$

Working this out, the defocus-only position derivative is:

$$\frac{\mathrm{d}\vec{x}_{df}}{\mathrm{d}\alpha} = \vec{V}_\alpha \cdot H^{-1}(\vec{x}_{df})$$

where:

$$\vec{V}_\alpha = \begin{cases} \begin{pmatrix} -\frac{3}{4d^2}x_{df}^2 - \frac{1}{4d^2}y_{df}^2 \\ -\frac{1}{2d^2}x_{df}y_{df} \end{pmatrix} & \text{if} \quad \alpha = C_x \\ \begin{pmatrix} -\frac{1}{2d^2}x_{df}y_{df} \\ -\frac{1}{4d^2}x_{df}^2 - \frac{3}{4d^2}y_{df}^2 \end{pmatrix} & \text{if} \quad \alpha = C_y \\ \begin{pmatrix} -\frac{x_{df}}{d} \\ \frac{y_{df}}{d} \end{pmatrix} & \text{if} \quad \alpha = A_x \\ \begin{pmatrix} -\frac{y_{df}}{d} \\ -\frac{x_{df}}{d} \end{pmatrix} & \text{if} \quad \alpha = A_y \\ \begin{pmatrix} -1 \\ 0 \end{pmatrix} & \text{if} \quad \alpha = T_x \\ \begin{pmatrix} 0 \\ -1 \end{pmatrix} & \text{if} \quad \alpha = T_y \\ \begin{pmatrix} \frac{3C_x}{2d^3}x_{df}^2 + \frac{C_x}{2d^3}y_{df}^2 + \frac{C_y}{d^3}x_{df}y_{df} + \frac{A_x}{d^2}x_{df} + \frac{A_y}{d^2}y_{df} \\ +\frac{C_y}{2d^3}x_{df}^2 + \frac{3C_y}{2d^3}y_{df}^2 + \frac{C_x}{d^3}x_{df}y_{df} + \frac{A_y}{d^2}x_{df} + \frac{A_x}{d^2}y_{df} \end{pmatrix} & \text{if} \quad \alpha = d \end{cases}$$

A.6 Derivative of the Hessian

The Hessian matrix is:

$$H = \begin{pmatrix} \frac{3C_x}{2d^2}x_{df} + \frac{C_y}{2d^2}y_{df} + \frac{A_x}{d} + 1 & \frac{C_y}{2d^2}x_{df} + \frac{C_x}{2d^2}y_{df} + \frac{A_y}{d} \\ \frac{C_y}{2d^2}x_{df} + \frac{C_x}{2d^2}y_{df} + \frac{A_y}{d} & \frac{C_x}{2d^2}x_{df} + \frac{3C_y}{2d^2}y_{df} - \frac{A_x}{d} + 1 \end{pmatrix}$$

Differentiating this is straightforward. Note that the derivative matrix will contain some terms that will be present for any differentiation parameter α and some terms that depend on α. It is useful to separate these two pieces, as the term that is constant in α only needs to be evaluated once per pixel, and so I express the derivative as:

$$\frac{\mathrm{d}H}{\mathrm{d}\alpha} = \begin{pmatrix} \frac{3C_x}{2d^2}\frac{\mathrm{d}x_{df}}{\mathrm{d}\alpha} + \frac{C_y}{2d^2}\frac{\mathrm{d}y_{df}}{\mathrm{d}\alpha} & \frac{C_y}{2d^2}\frac{\mathrm{d}x_{df}}{\mathrm{d}\alpha} + \frac{C_x}{2d^2}\frac{\mathrm{d}y_{df}}{\mathrm{d}\alpha} \\ \frac{C_y}{2d^2}\frac{\mathrm{d}x_{df}}{\mathrm{d}\alpha} + \frac{C_x}{2d^2}\frac{\mathrm{d}y_{df}}{\mathrm{d}\alpha} & \frac{C_x}{2d^2}\frac{\mathrm{d}x_{df}}{\mathrm{d}\alpha} + \frac{3C_y}{2d^2}\frac{\mathrm{d}y_{df}}{\mathrm{d}\alpha} \end{pmatrix} + \eta_\alpha$$

where:

$$\eta_\alpha = \begin{cases} \begin{pmatrix} \frac{3}{2d^2}x_{df} & \frac{1}{2d^2}y_{df} \\ \frac{1}{2d^2}y_{df} & \frac{1}{2d^2}x_{df} \end{pmatrix} & \text{if} \quad \alpha = C_x \\ \begin{pmatrix} \frac{1}{2d^2}y_{df} & \frac{1}{2d^2}x_{df} \\ \frac{1}{2d^2}x_{df} & \frac{3}{2d^2}y_{df} \end{pmatrix} & \text{if} \quad \alpha = C_y \\ \begin{pmatrix} \frac{1}{d} & 0 \\ 0 & -\frac{1}{d} \end{pmatrix} & \text{if} \quad \alpha = A_x \\ \begin{pmatrix} 0 & \frac{1}{d} \\ \frac{1}{d} & 0 \end{pmatrix} & \text{if} \quad \alpha = A_y \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} & \text{if} \quad \alpha \in \{T_x, T_y\} \\ \begin{pmatrix} -\frac{3C_x}{d^3}x_{df} - \frac{C_y}{d^3}y_{df} - \frac{A_x}{d^2} & -\frac{C_y}{d^3}x_{df} - \frac{C_x}{d^3}y_{df} - \frac{A_y}{d^2} \\ -\frac{C_y}{d^3}x_{df} - \frac{C_x}{d^3}y_{df} - \frac{A_y}{d^2} & -\frac{C_x}{d^3}x_{df} - \frac{3C_y}{d^3}y_{df} + \frac{A_x}{d^2} \end{pmatrix} & \text{if} \quad \alpha = d \end{cases}$$

Appendix B

Software

B.1 Fitting Program

The program `abr_fit_image`, written in C++, will sequentially fit any number of .FITS whose filenames are passed as command-line arguments. The results of the fit will be printed to stdout in the following order:

1. filename of target .FITS file
2. the initial guesses used by the routine
3. best fit parameter values
4. covariance matrix entires, starting with the upper left entry and working down to the lower right entry
5. total number of iterations before convergence
6. final chi-squared value
7. total degrees of freedom
8. final value of the Levenberg-Marquardt damping factor

The convention for ordering the parameters is: c_x, c_y, a_x, a_y, s, t_x+, t_y+, $d+$, $F+$, $b+$, t_x-, t_y-, $d-$, $F-$, $b-$, where the + and − symbols indicate a parameter that is

applied to only either the positive or negative defocus image. Calling `abr_fit_image` with the `-v` option will cause it to print the status of each iteration.

When `abr_fit_image` runs, it will first call `unpack_fits_data` to read from the .FITS file, and then call `compute_iniital_guesses` and `compute_error` to generate initial guesses and error estimates. These are used to call `lev_mar_fit`, which carries out the fitting procedure. The following list gives all of the functions used in the fitting procedure, along with a brief description.

- `void grad_wavefront_residual(int n, double x[], double f[], double params[], double x_im, double y_im)`

  Given the coordinates of a defocus-only point in the two-element array `x`, a set of aberration parameters in `params`, and the coordinates of an image plain point in `x_im` and `y_im`, `grad_wavefront_residual` returns in the two-element array `f` the residual vector between the given image point and the image point produced by mapping the given defocus-only point to the image plane using the given aberration parameters. The argument `n` is the dimension of `x` and `f`, which should always be 2.

- `void unpack_fits_data(char* file_name, double flat_image[], double image_pos[], double image_neg[], int num_data_pts)`

  This function reads the .FITS file with name `file_name`, containing a positive defocus image as the primary data array, the negative defocus image as the first extension, and no other extensions. The images must be 120x120 arrays with 16 integer values. This is the format of the data produced by the VISTA low-order wavefront sensor. If `filename` is not formatted correctly, an error message will be printed. The primary data array is stored in the one-dimensional array `image_pos`, and similarly the first extension data is stored in `image_neg`. `flat_image` is a concatenation of these arrays, with `image_pos` first. `num_data_pts` is the total number of pixels in both images.

- `void compute_error(double flat_image[], double flat_errors[],`

`int num_data_pts)`

Computes the statistical error in each pixel, assuming a Poisson counting experiment. The number of counts reported by the VISTA low-order wavefront sensors is 4 times the number of hits detected, [2] so the error in a pixel with reported intensity I is given as $\sigma = 2\sqrt{I}$. `flat_image` should contain the observed data and `num_data_point` should be the length of `flat_image`. The errors in each pixel are returned in `flat_errors`.

- `void lev_mar_fit(double pixel_number[], double flat_image[],`
  `double errors[], int num_data_pts, double fit_params[],`
  `int num_fit_params, double **covar_matrix, double *chisq,`
  `double chisq_tol, int num_successes, int verbose)`

  This runs the *Numerical Recipes* Levenberg-Marquardt routine to fit the data in `flat_image`. This array should contain the data from a pair of defocused images concatenated into a single one-dimensional array. The data will be fit for the parameters in `fit_params`, which should contain the initial guesses when `lev_mar_fit` is called. The fitting algorithm will iterate until the change in chi-squared is less than 0.01 for `num_successes` consecutive iterations.

- `void blurred_image_model(double x_im, double y_im,`
  `double abr_params[], int num_abr_params,`
  `double *brightness, double partials[], double pixel_width)`

  This function preforms the convolution. Given a pair of image coordinates `x_im` and `y_im`, and a set of aberration parameters `abr_params`, the function `image_model` computes the convolution kernel as in Section 2.4, calls `fat_image_model` to get the unblurred brightnesses and derivatives for pixels nearby (`x_im`, `y_im`), and then performs the weighted averages. It also computes the derivative with respect to the seeing diameter parameter. To save time, this function will store unblurred model values as it is iterated through an image, and will then only call `flat_image_model` if necessary.

- `void flat_image_model(double pixel_number, double fit_params[], double *brightness, double derivatives[], int num_fit_params)`

  This is a wrapper for the function `image_model` that allows `image_model` to be accessed as though it were a one-variable function. Given a `pixel_number`, this function determines which of the two images this pixel is located in, and it determines the pixel's coordinate in the image plane of that image. It then selects only those parameters from `fit_params` that pertain to this image, and uses them to call `image_model`. The derivatives with respect to the parameters that pertain to the other image are set to zero before the brightness and full set of derivatives are returned. Pixel numbers for the positive defocus image are assigned starting with 0 in the lower left corner of the image and increase as you move across to the right and upwards, with pixel number $N^2 - 1$ located in the upper right corner for an NxN image. For the negative defocus image, pixel numbers start with N^2 in the lower left and end with $2N^2 - 1$ in the upper right.

- `void image_model(double x_im, double y_im, double abr_params[], int num_abr_params, double *brightness, double partials[], double pixel_width)`

  Given a pair of image coordinates `x_im` and `y_im`, and a set of aberration parameters `abr_params`, the function `image_model` computes the brightness and the partial derivatives with respect to each parameter in `abr_params` of the pixel centered on that coordinate (`x_im`, `y_im`), according to the model described above.

- `int project_to_defocus_only(double x_im, double y_im, double params[], double* x_df, double* y_df)`

  Using the aberration parameters in `pramas`, `project_to_defoucs_only` computes the defocus-only point that maps to the image point (`x_im`, `y_im`). This is done by finding the zero of `grad_wavefont_residual` using the *Numerical Recipes* Newton's method routine `nr_newt`. The solution will be stored in `x_df`

and `y_df`, and the return value will indicate a failed convergence with: 0 → convergence, and 1 → no convergence.

- `double inverse_hessian_det(double x_df, double y_df, double params[])`

  Returns $1/\det H(\vec{x}_{df})$ evaluated at the given coordinates and aberration parameters.

- `void inverse_hessian(double x_df, double y_df, double params[], double Hinv[][2], double Hinv_determinant)`

  Computes the inverse of the Hessian matrix evaluated at (`x_df`, `y_df`) with the aberration parameters given in `params`, and stores it in `Hinv`. `Hinv_determinant` should contain the inverse of $\det H(x_{df}, y_{df})$ as computed by `inverse_hessian_det`.

- `void grad_df_position(double x_df, double y_df, double params[], double Hinv[][2], double grad_x_df[], double grad_y_df[])`

  Assuming a fixed image plane point, this returns the gradient of the projected x and y defocus-only coordinates, with respect to the aberration parameters. This is computed using the procedure described in Section A.5.

- `void grad_bulk_term(double x_df, double y_df, double grad_x_df[], double grad_y_df[], double Hinv[][2], double params[], double bulk_term[])`

  This computes the bulk term of the derivative of the pixel brightness (the first term in Equation A.7) with respect to all of the parameters in `params`.

- `void grad_edge_term(double x_df, double y_df, double grad_x_df[], double grad_y_df[], double params[], double l, double Rin, double Rout, double edge_term[])`

  This computes the edge term of the derivative of the pixel brightness (the second term in Equation A.7) with respect to all of the parameters in `params`. This function calls `partial_frac_below_circle_wrt_defocus` and

`grad_frac_below_circle_wrt_dfposition` in order to compute the derivative of the illuminated fraction f,

- `double ill_frac(double x_df, double y_df, double Rin, double Rout, double pixel_length)`

  Returns the fraction of the area of a pixel with length `pixel_length`, centered on (`x_df`, `y_df`), that is illuminated in the defocus-only image. `Rin` and `Rout` are the inner and outer radii of the illuminated annulus of the defocus-only image, not the radii of the pupil. This is computed as described in Section A.1, where ϕ is computed by calling `frac_below_circle`.

- `double frac_below_circle(double x_df, double y_df, double radius, double pixel_length)`

  Returns the fraction of the area of a pixel with length `pixel_length`, centered on (`x_df`, `y_df`), that is located inside a circle centered on the origin with the given radius. This is computed using the procedure described in Section A.1 and uses the approximation that `pixel_length` $\ll$ `radius`.

- `double partial_frac_below_circle_wrt_defocus(double x, double y, double R, double l, double defocus)`

  Computes the partial derivative of the `frac_below_circle` with respect to the defocus parameter. `frac_below_circle` depends on the defocus parameter in two ways: changing defocus will change the defocus-only position that corresponds to a given fixed image pixel and it causes the radius of the defocus-only illuminated annulus to change. This function computes the term in the derivative of `frac_below_circle` corresponding only to the change in defocus-only radius. `grad_frac_below_circle_wrt_dfposition` will compute the terms corresponding to the change in the defocus-only position of the pixel.

- `void grad_frac_below_circle_wrt_dfposition(double x, double y, double R, double frac_below_grad[2], double l)`

Computes the gradient of the `frac_below_circle` with respect to the defocus-only image coordinates. The function `grad_edge_term` multiplies these by the derivatives of the defocus-only coordinates with respect to the aberration parameters in order to get the derivative of `frac_below_circle` with respect to the aberration parameters.

- `void compute_initial_guesses(double flat_image[], int num_fit_params, int num_common_fit_params, int num_x_pixels, int num_y_pixels, double param_guesses[])`

  Computes initial guesses for all of the parameters except coma and astigmatism, whose guesses are set to zero. These are computed using the procedure described in Section 3.3. The guesses returned by this `compute_iniital_guesses` are used to call `lev_mar_fit`.

- `void newt(double x[], int n, int *check, double parameters[], double x_image, double y_image, void (*vecfunc)(int, double [], double [], double [], double, double));`

  This is a *Numerical Recipes* Newton's method solver that is used by `project_to_defocus_only` to find the zeros of `grad_wavefront_residual`. It has been slightly modified to allow the aberration parameters to be passed as additional arguments, as they are needed to evaluate `grad_wavefront_residual`. This function employs the following *Numerical Recipes* routines: `fdjac`, `lubksb`, `ludcmp`, `lnsrch`, and `fmin`. [8]

- `void mrqmin(double x[], double y[], double sig[], int ndata, double a[], int ia[], int ma, double **covar, double **alpha, double *chisq, void (*funcs)(double, double [], double *, double [], int), double *alamda, int verbose)`

  This is a *Numerical Recipes* routine which executes one iteration of a Levenberg-Marquardt fitting algorithm. This is called repeatedly by `lev_mar_fit`. This function employs the following *Numerical Recipes* routines: `mrqcof`, `gaussj`,

and `covsrt`. [8]

B.2 Image Program

The program `abr_draw_image`, written in C++, will read from stdin 9 aberration parameters and 5 telescope parameters, and then compute the image formed by these parameters according to the model. Note that this does not produce a pair of defocused images, but rather a single image. The primary use of this program is to test the reasonableness of the model after making modifications, as well as to generate synthetic data with which to test the fitting routine `abr_fit_image`.

`abr_draw_image` accepts aberration values in *polar* format, as opposed to the Cartesian format used by `abr_fit_image` and which I have used throughout the rest of this paper. This was done to make it easier to visualize what images will be produced when calling `abr_draw_image` with a given set of parameters. The convention for ordering the parameters is: number of pixels along x-direction, number of pixels along y-direction, inner pupil radius, outer pupil radius, width of a pixel, F, c, θ_c, a, θ_a, d, t, θ_t, and b. The resulting image will be written to stdout, starting with the upper left pixel, then working across to the right and downwards, ending with the lower right pixel.

The operation of `abr_draw_image` is simply to loop over all pixels in the output image, and at each pixel call the *Numerical Recipes* routine `newt` to compute the corresponding defocus-only points, and then call the functions `inverse_hessian_determinant` and `ill_frac` to compute the remaining pieces of Equation 2.9. These functions operate in exactly the same manner as their counterparts from `abr_fit_image`.

Bibliography

[1] N. Bissonauth, P. Clark, Gavin B. Dalton, R. Myers, and William J. Sutherland. Image analysis algorithms for critically sampled curvature wavefront sensor images in the presence of large intrinsic aberrations. *Proc. SPIE 5496, 738*, 2004.

[2] P. Cark, P. Berry, R. Bingham, N. Bissonauth, M. Caldwell, C. Dunlop, D. Henry, P. Luke, and D. Robertson R. Myers. Wavefront sensing within the vista infrared camera. *Proc. SPIE 5499*, 2004.

[3] LSST Science Collaboration. Lsst science book, version 2.0. arXiv:0912.0201v1, 2009.

[4] K. Honscheid and D. L. DePoy for the DES Collaboration. The dark energy camera (decam). arXiv:0810.3600v1, 2008.

[5] M. James Jee and J. Anthony Tyson. Toward precision lsst weak-lensing measurement. i. impacts of atmospheric turbulence and optical aberration. *Publications of the Astronomical Society of the Pacific*, 2011.

[6] V. N. Mahajan. *Optical imaging and aberrations, part I.* SPIE, Bellingham, WA, 2001.

[7] Lothar Noethe. *Active Optics in Modern, Large Optical Telescopes.* Progress in Optics. North Holland, first edition, 10 May 2002.

[8] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, New York, NY, second edition.

[9] Paul L. Schechter, Gregory S. Burley, Charles L. Hull, Matthew Johns, Hubert M. Martin, Skip Schaller, Stephen A. Shectman, and Steve C. West. Active optics on the baade 6.5-m (magellan i) telescope. *Proc. SPIE 4837, 619*, 2003.

[10] R. N. Wilson. *Reflecting Telescope Optics, Part I.* Springer, Berlin, 2007.