Seeing Clearly with KAPAO: Measuring Performance with Data Analysis Tools for Adaptive Optics

Joseph Daniel Long

Submitted in partial fulfillment of the degree of Bachelor of Arts in Physics and Astronomy

April 23, 2014
Acknowledgments

First of all, I would like to acknowledge the support of Dr. Philip Choi, who provided guidance without which this thesis could not have been written, and whose advice is indispensable. Secondly, the breadth of knowledge of Dr. Scott Severson on the subject of optics (adaptive and otherwise) was an invaluable resource while undertaking this research. Many thanks are due to Dalton Bolger, whose skill with hardware no doubt prevented thousands of dollars in damage I could have done if I had started fiddling with components.

I would also like to thank the previous generation of KAPAO researchers: Daniel Contreras, Lorcan McGonigle, and Erik Littleton, for moving the project forward and leaving behind such valuable documentation.
Contents

1 Background: Turbulence and Telescopes 1
  1.1 The Limitations of Ground-based Observing . . . . . . . . . . . . . . . . . . . 1
  1.2 Solutions to the Turbulence Problem . . . . . . . . . . . . . . . . . . . . . . 3
  1.3 Basics of Adaptive Optics . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
  1.4 KAPAO: A Pomona Adaptive Optics project . . . . . . . . . . . . . . . . . . 5

2 Performance Measurements from Images 7
  2.1 Upper Limits on Imaging Performance . . . . . . . . . . . . . . . . . . . . . . 8
  2.2 Creating a Synthetic Point-Spread Function . . . . . . . . . . . . . . . . . . 12
  2.3 Calculating Strehl Ratios from Observation Data . . . . . . . . . . . . . . . 17

3 Turbulence Characterization 19
  3.1 Turbulence Power Spectra . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  3.2 Telemetry Format Details . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
  3.3 Spatial Power Spectrum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
    3.3.1 An Initial Power Spectrum Computation . . . . . . . . . . . . . . . . . 23
    3.3.2 Refining the Power Spectrum Computation . . . . . . . . . . . . . . . . 26
    3.3.3 Tilt-subtraction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
  3.4 Turbulence Power Spectra . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

4 Performance Measurements 35
  4.1 Introduction and Summary of Observing Run . . . . . . . . . . . . . . . . . . 35
  4.2 Curves of Growth . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
  4.3 Strehl Ratios . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

Appendices 41
  A Code Listings 43
    A.1 Fourier Transform Explanation . . . . . . . . . . . . . . . . . . . . . . . . 43
    A.2 Spatial Frequencies in One Direction and the 2D Fourier Transform . . 44
## A.3 Spatial Frequencies in Two Directions and the 2D Fourier Transform

### B aotools PyRAF Package

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Installation</td>
<td>49</td>
</tr>
<tr>
<td>B.2 Usage</td>
<td>50</td>
</tr>
<tr>
<td>B.2.1 aoavgcube</td>
<td>50</td>
</tr>
<tr>
<td>B.2.2 cubeflatfield</td>
<td>50</td>
</tr>
<tr>
<td>B.2.3 cubemedian</td>
<td>50</td>
</tr>
<tr>
<td>B.2.4 cubestack</td>
<td>51</td>
</tr>
<tr>
<td>B.2.5 cubetoframes</td>
<td>51</td>
</tr>
<tr>
<td>B.2.6 findbright</td>
<td>52</td>
</tr>
<tr>
<td>B.2.7 photstrehl</td>
<td>52</td>
</tr>
<tr>
<td>B.2.8 photstrehlframe</td>
<td>53</td>
</tr>
<tr>
<td>B.2.9 pngtocube</td>
<td>54</td>
</tr>
<tr>
<td>B.2.10 pngtofits</td>
<td>55</td>
</tr>
<tr>
<td>B.2.11 removeband</td>
<td>55</td>
</tr>
<tr>
<td>B.2.12 strehlcube</td>
<td>55</td>
</tr>
<tr>
<td>B.2.13 strehlframe</td>
<td>56</td>
</tr>
<tr>
<td>B.3 Overview</td>
<td>57</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Types of KAPAO Telemetry Files ........................................ 23
3.2 Power Law Dependence in Initial Spatial Power Spectrum ............ 25
# List of Figures

1.1 A cartoon view of KAPAO and the paths light travels through the system.

2.1 Example periodic signal to be Fourier transformed

2.2 1-D Fourier transform output for the example signal

2.3 Visualizations of the two dimensional Fourier transform
   (a) 2D Fourier Transform showing spatial frequencies in horizontal direction
   (b) 2D Fourier Transform showing spatial frequencies in horizontal and vertical direction

2.4 Central region of a numerical aperture array for a telescope like TMO

2.5 Output of 2D FFT showing engineering/computing convention for frequency ordering

2.6 A synthetic PSF image for an unobscured circular aperture

2.7 Plot of the central row of the PSF image, showing profiles for obscured and unobscured apertures

2.8 Several example PSFs with their corresponding numerical apertures

3.1 Schematic plot of predicted power law dependencies of the spectrum of wavefront phase variations due to atmospheric turbulence (Hardy, 1998)

3.2 Turbulence power spectrum for the ViLLaGEs instrument at Lick Observatory (Morzinski et al., 2010)

3.3 The arrangement and numbering of KAPAO’s 97 subapertures

3.4 Turbulence power spectrum of slope values averaged across all subapertures (from December 2013 observations of β Pegasi)

3.5 Power spectrum of x-direction slope values averaged across all subapertures from Figure 3.4, over-plotted with power law fits detailed in Table 3.2

3.6 Plot of measured slope values across an entire run for a good and bad subaperture

3.7 Plot showing timestamps jumping backwards in time intermittently

3.8 Visualizations of the subaperture include mask and intensity map
   (a) Subaperture mask visualization from the intensity map
(b) Single frame from intensity map .............................................. 28
3.9 Power spectrum of x-direction slope values averaged across the “good” sub-
 apertures shown in Figure 3.8a. .................................................. 29
3.10 Image showing the fraction of timestamps for which each subaperture had
 sufficient intensity for a slope to be measured. A value near unity means
 that few or no timesteps contained artificial zero values in the X and Y
 slopes for that subaperture. ....................................................... 29
3.11 Subtracting the overall tip/tilt in subaperture slopes ............................... 32
   (a) Overall trends in subaperture slope timeseries ............................... 32
   (b) Subaperture slope timeseries with overall tip/tilt removed ............... 32
3.12 Scatter plot of measured speckle pattern centroids showing tracking drift . . 33
3.13 The power in the overall slope at different frequencies for X and Y slopes
 overplotted on the power spectrum for a single (good) subaperture. ......... 33
3.14 Power spectrum of the tilt-subtracted residual slopes averaged over good
 subapertures. ................................................................. 34
3.15 Power spectrum of the tilt-subtracted residual slopes averaged over good
 subapertures. ................................................................. 34
4.1 Images of η Piscium in near-infrared (H-band) and optical (Sloan z′ band)
 from December 2013 observations. ............................................ 36
   (a) Comparison of open/seeing-limited and AO lucky images of η Piscium
      in H-band ................................................................. 36
   (b) Comparison of open/seeing-limited and AO lucky images of η Piscium
      in z′-band ................................................................. 36
4.2 Visualizations of imaging performance on η Piscium from December 2013
 observations. ................................................................. 38
   (a) Curve of growth comparison for H-band images of η Piscium ........ 38
   (b) Curve of growth comparison for z′-band images of η Piscium ....... 38
4.3 Plot of near-IR (H-band) Strehl ratios computed frame-by-frame for observ-
 ations of η Piscium. ............................................................. 40
4.4 Plot of optical (Sloan z′ band) Strehl ratios computed frame-by-frame for
 observations of η Piscium. ........................................................... 40
Chapter 1

Background: Turbulence and Telescopes

Most humans, and certainly most astronomers, are familiar with the sight of a point of starlight embedded in the night sky, twinkling. What many do not know, however, is that the twinkling we observe is an effect that occurs in the final few hundred miles of the starlight’s trillion-mile journey. This distortion is entirely due to Earth’s atmosphere interfering with light on its way to our eyes. To an astronomer, this twinkling is an undesirable side-effect of observing the stars from the ground. For example, the ability to distinguish between stars in a crowded field is negatively affected by the distorting effects of the atmosphere. For most of history, astronomers were unable to overcome this distortion in any meaningful way. Only very recently have advances in computer controlled adaptive optics systems made high resolution imaging of the stars from the ground feasible.

1.1 The Limitations of Ground-based Observing

Classical optics predicts a theoretical maximum for an optical system’s resolution: the diffraction limit, which determines the minimum angular separation between two sources that one can resolve. For a circular aperture, the diffraction limit for angular resolution $\theta$ depends on $\lambda$, the wavelength of light you are imaging, and $D$, the diameter of the lens’ aperture:

$$\theta \propto \frac{\lambda}{D}$$

As telescope builders increased the size of their primary optics in hopes of capturing more light and observing dimmer objects, they noticed that as $D$ increased they were not measuring a corresponding decrease in $\theta$, the smallest resolvable angular separation. For example,
our Table Mountain Observatory telescope \((D = 1\, \text{meter})\) should be able to resolve angular separations of 0.2013\(^{''}\) when observing 800 nm light. When completed, the Giant Magellan Telescope \((D \approx 22\, \text{meters})\) ought to be able to resolve features only 0.004\(^{''}\) apart. However, once the diameter of the primary optic surpasses about 0.2 m, angular resolution for ground-based telescopes stops increasing. Why don’t we get any additional angular resolution by increasing the diameter of our mirror by five times (at TMO) or ten (at GMT)? The culprit is the atmosphere: the air currents and cells of differing air density that cause twinkling are distorting starlight in dynamic and unpredictable ways, smearing the light from a distant target across a larger region of the field of view.

Astronomers use the term “seeing” (or equivalently “astronomical seeing”) for any measurement of local atmospheric conditions. A common quantity to measure for the seeing is the full width at half maximum (FWHM, or colloquially “full-width half-max”) for some star in the image over the course of an observing run. (However, FWHM is not the only metric, and other quantities can be cited as measurements of seeing conditions.) For a given observatory, it is common to reference typical seeing values when planning an observing campaign in order to gauge how well one is able to resolve stars or other objects. Seeing is also calculated during the run from the FWHM of stars being imaged to monitor the effective resolving power of the telescope as conditions change. It’s important to note that atmospheric distortion, while it can be broadly characterized by a single number, is changing from moment to moment. Any system that attempts to correct for it will also have to measure it repeatedly, hundreds or thousands of times each second.

When explaining exactly how a distant source is distorted, it is common to refer to a **point spread function (PSF)**. If we model our distant source as a point (e.g. a Dirac delta function), its projection onto the image plane seems like it should also be a point. In reality, even the most ideal optical system must spread out a point source into a pattern on the image plane. This could be described by a predicted point spread function, in which all the specifications of the optical system are used to calculate a theoretical prediction for how the system affects light, or an observed point spread function, where we have data from a camera and need to fully describe how the pattern differs from a point source. (A famous example of the benefits of understanding one’s PSF well is the Hubble Space Telescope: by analyzing the distortion of stars in the images sent back to Earth, astronomers were able to determine that the contractor that supplied the mirror, Perkin-Elmer, had polished the mirror very precisely to the wrong shape!)

Seeing conditions vary with weather, but also with the amount of air being looked through and consequently the location of the observatory. For example, looking at a target near the horizon will necessitate cutting through the atmosphere at an angle, giving the photons of

---

\(^1\)We will use \(''\) and arcseconds interchangeably, as well as \(\text{' }\) and arcminutes. One arcminute is \(1/60^{\text{th}}\) of a degree, and one arcsecond is \(1/3600^{\text{th}}\) of a degree. (There are 206265 arcseconds, or 3438 arcminutes, in one radian.)
1.2. SOLUTIONS TO THE TURBULENCE PROBLEM

interest more chances to be disturbed by the atmosphere. This is one reason that observatories are frequently sited on mountain-tops: not only are mountain-tops remote and likely less light-polluted, but there is simply less atmosphere to look through at higher altitudes. Pomona’s own Table Mountain Observatory averages seeing of $\approx 1'' - 1.5''$ FWHM in the mountains of Wrightwood, CA. The best seeing conditions on Earth for an observatory, e.g. at Mauna Kea, approach $0.4'' - 0.7''$ FWHM.

1.2 Solutions to the Turbulence Problem

There exist technological solutions to mitigate or eliminate the effects of a turbulent atmosphere on incoming light. As previously discussed, atmospheric distortion decreases as the amount of atmosphere through which one is looking decreases. Taken to the logical extreme, this is essentially the argument for space telescopes: they reside outside the atmosphere, and are not subject to its distortions. For example, since the atmosphere strongly attenuates radiation in wavelengths outside of the visible and radio bands, one must go to space to conduct X-ray or infrared observations. However, since we cannot do anything about this particular atmospheric characteristic from the ground, we will leave the discussion of the benefits of space-based observatories to others.

In recent years, ground-based observatories have been outfitting their telescopes with adaptive optics (AO) instruments to correct for atmospheric distortion. As suggested by the title of this thesis, adaptive optics instruments will be discussed in detail shortly. Adaptive optics instruments work by measuring the distortion in incoming light, then deforming a reflective surface by very small amounts such that when light bounces off of it, the measured distortion is “un-done” by the mirror. Adaptive optics instruments have the advantage of being (comparatively) easy to install on existing telescopes: one does not have to re-engineer the telescope itself, merely bolt on an instrument that contains all of the necessary measurement and correction tools. The goal of AO is to recover the angular resolution lost to atmospheric distortion by iteratively measuring and correcting the collected light, hopefully approaching the diffraction limit for the telescope.

A third option for improving effective angular resolution is “lucky imaging”. Rather than applying some active correction, it is effectively a data analysis technique. In brief, the observer takes many exposures with shorter integration times (fractions of a second, rather than the seconds- or minutes-long exposures typical in astronomy), which “freezes” turbulence in the atmosphere such that it will change negligibly over the course of the exposure. Then, the astronomer discards those exposures where the image is badly distorted by the atmosphere and combines those conditions were briefly less turbulent (the “lucky” images) to form a high-resolution image. There is a related technique called “speckle imaging” that takes similarly takes advantage of short exposures, and uses computer-based image
analysis to undo the effects of atmospheric seeing on short timescales.

1.3 Basics of Adaptive Optics

Adaptive optics systems can be discussed as two main parts: the wavefront sensing component, and the wavefront correction component. The wavefront correction is carried out by a tip-tilt mirror and a deformable mirror (DM) that are controlled by a computer with information from the wavefront sensing leg of the instrument. The tip-tilt mirror provides low-order corrections, while the deformable mirror provides higher-order corrections. As mentioned previously, the control computer deforms the DM such that it assumes the conjugate shape to the incoming light wavefronts. Thus, when light reflects off of it, the differences in optical path length through the atmosphere that distorted the light are corrected by the slightly different distances to the DM that light must travel before reflecting.

In our instrument, the wavefront sensing component is situated after the light has already been corrected by the DM. This allows the instrument to operate in a feedback loop: if the correction is good, the wavefront sensor will see a perfectly corrected wavefront and the computer will not instruct the DM to move. If conditions change, however, the computer will compute a differential change to the DM based on the new displacements on the wavefront sensor. Our instrument for the Table Mountain 1-meter telescope uses a Shack-Hartmann wavefront sensor, which takes collimated light in and divides it up into “subapertures” spatially. By focusing the light in these subapertures, the sensor can measure the amount and direction of distorting effects in a region of the wavefront. A perfect correction (and flat wavefront) would result in perfectly centered spots of light in each subaperture on our wavefront sensing camera. Residual wavefront error will result in off-center spots, which are reported by the control loop software as “slopes” calculated by taking the ratio of intensities on the two halves of the subaperture in the horizontal and vertical direction.

The wavefront sensor uses some of the light collected by the telescope, but most of it is destined for the science cameras. (We use dichroic filters to send different wavelengths to the two science cameras and the WFS camera.) Both the WFS and the science camera lie after the DM correction, so the science camera receives the corrected wavefronts and the WFS sensor can iteratively measure and improve the correction.

In some adaptive optics instruments, the wavefront sensing component does not see AO-corrected wavefronts. Instead, the wavefront sensor measures the distorted wavefront coming in from the atmosphere, without the iterative feedback loop effect that our instrument takes advantage of. This approach allows the instrument to provide a direct measurement of atmospheric conditions, at the expense of not being able to verify the accuracy of the
correction within the system itself. This is true of the ViLLaGEs instrument built by Morzinski et al., for example [6].

1.4 KAPAO: A Pomona Adaptive Optics project

KAPAO is the name of the instrument being designed and constructed for the Table Mountain Observatory 1-meter telescope. Officially, KAPAO is a recursive acronym (like “GNU’s Not Unix”, or “PHP Hypertext Preprocessor”) for KAPAO: A Pomona Adaptive Optics project.\(^2\) The end goal is to construct a natural guide star adaptive optics instrument for permanent installation on the telescope, capable of remote operation and imaging in visible and near-infrared bands.

![Diagram of KAPAO system](image)

Figure 1.1: A cartoon view of KAPAO and the paths light travels through the system.  
1. Light from the telescope comes in, distorted by the atmosphere.  
2. It bounces off a MEMS deformable mirror, which restores a near-planar wavefront.  
3. The wavefront sensor measures residual uncorrected distortion.  
4. The control software moves actuators behind the deformable mirror to compensate.  
5. The corrected light is imaged by the science cameras in near-IR and visible.

\(^2\)Another possible interpretation from Dr. Severson: “Kid-Assembled Pomona Adaptive Optics”.
Over the past three years, a rotating cast of undergraduate students from Pomona College, Harvey Mudd College, and Sonoma State University has developed the optical and mechanical design for the system, as well as constructed a proof of concept instrument with off-the-shelf components called KAPAO Alpha. We are now implementing an improved design created by Lorcan McGonigle (Pomona ’13), KAPAO Prime, which incorporates the lessons learned from Alpha’s construction and operation. In the following chapters, I will explain my contribution to the performance characterization of KAPAO Prime, as well as data analysis tools I have developed for observers using the instrument.
Chapter 2

Measuring Instrument Performance from Science Images

The current and final stage of KAPAO is the construction of a facility-class instrument for permanent installation. As we work towards this goal, it is very important to understand the theoretical limits on our system’s performance as well as how close we are to achieving them. When imaging a point source at infinity with our telescope, our ability to resolve two sources is dictated by \( \theta \simeq \frac{\lambda}{D} \), or, more precisely

\[ \sin \theta = 1.22 \frac{\lambda}{D} . \]

This is a mathematical statement of the Rayleigh criterion, which states that two sources are “just resolved” if the maximum value of one source’s diffraction pattern lies at the first minimum of the other source’s diffraction pattern. The leading coefficient is specific to the case where incoming light passes through a circular aperture, and is approximately the first zero of the Bessel function \( J_1 \) divided by \( \pi \). (In any case, astronomers omit the leading coefficient and make use of the small angle approximation, giving the approximate expression: \( \theta \sim \frac{\lambda}{D} \).)

This gives us a measure of our theoretical minimum angular separation, but that is not much good to us as a measurement of instrument performance unless we are able to measure how close we are getting to said theoretical minimum. After all, not every observing target for the instrument will lie in a field with optimally-separated “just resolved” sources. In this chapter we discuss computing an ideal point spread function for our 1-meter telescope
at Table Mountain Observatory and the development of code that compares it to measured PSFs from science images.

2.1 Upper Limits on Imaging Performance

The goal of an adaptive optics system is to recover resolving ability lost to atmospheric distortion and form an image of a distant object as if we were viewing it through still air or vacuum. There is an equation in optics, the Fraunhofer diffraction equation, that describes the pattern the intensity on an image plane in terms of an integral over an arbitrary aperture through which incident waves of light passed. The Fraunhofer diffraction equation allows us to predict what our distant point source will look like, purely as a function of the shape of our aperture.

We begin by considering a source such as a distant star, emitting light approximately equally in all directions, which creates spherical waves radiating outwards from the star. By the time these waves reach Earth, it is a good approximation to treat them as a plane waves, as the region of the spherical wave intersecting with our telescope is such a small angular region of the whole sphere that it is approximately flat.

Fraunhofer diffraction is described in detail in Optics by Hecht, specifically section 11.3.3 [3]. The takeaway can be summarized in this quote: “[...] the field distribution in the Fraunhofer diffraction pattern is the Fourier transform of the field distribution across the aperture (i.e. the aperture function).” The nuances of this statement take the better part of a chapter to elaborate, but suffice to say that we can make use of this fact to compute a PSF for our telescope numerically.

Suppose our plane waves are traveling along the $x$ axis, and are diffracted by an aperture to form a pattern on a $y$-$z$ plane located some distance $R$ beyond the aperture. In mathematical notation, this can be expressed as:

$$E(k_y, k_z) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} A(y, z)e^{i(k_y y + k_z z)} \, dy \, dz$$

$$= \mathcal{F}\{A(y, z)\}.$$ (2.1) (2.2)

Here we have defined the spatial frequencies at each point in the image plane by $k_y \equiv ky/R$ and $k_z \equiv kz/R$, and $A(y, z)$ is a function defined to be 1 where the aperture admits light and 0 otherwise. The notation $\mathcal{F}\{f(x, y)\}$ denotes the continuous Fourier transform of $f(x, y)$ in two dimensions.
### 2.1. UPPER LIMITS ON IMAGING PERFORMANCE

#### The Fourier Transform

As a brief refresher, the Fourier transform is a tool frequently employed in signal processing (and physics) that transforms a 1D signal from the time-amplitude domain to the frequency-power domain. For two-dimensional inputs like our aperture function, it produces a spatial frequency spectrum, which we will discuss later.

Suppose we had a signal consisting of a sine wave with angular frequency $\omega_a$ and a sine wave with angular frequency $\omega_b$, added together such that $\text{Signal}(t) = A\sin(\omega_a t) + B\sin(\omega_b t)$. From looking at a plot of the signal vs. time (Figure 2.1), we can guess that the signal in red is the sum of two periodic functions, but we cannot easily obtain the values of $\omega_a$ and $\omega_b$ (except, of course, that in this case I have plotted the components for the signal for clarity).

![Figure 2.1: The combined signal (in red) that we are analyzing. The two components added to create the signal are plotted with dashed lines. (See Code Listing A.1.)](image)

However, if we compute the Fourier transform of our $\text{Signal}(t)$ function, the values of $\omega_a$ and $\omega_b$ are easily obtained. Furthermore, in Figure 2.2 I have plotted the normalized Fourier transform, which shows us that the component at $\omega = 4(2\pi)$ is $1/4$ as strong as the component at $\omega = 2\pi$. If you consult Code Listing A.1, you will see that the input signal in Figure 2.1 is $\text{Signal}(t) = 1\sin(8\pi t) + \frac{1}{4}\sin(2\pi t)$.

Above, we have used a sampled (discrete) signal in the computation of the Fourier transform. There are mathematical functions for which it is straightforward to compute a Fourier transform analytically, but many scientific applications involve a discrete Fourier

---

1For an excellent conceptual overview of the Fourier transform (albeit not a mathematically rigorous one), I recommend Stuart Riffle’s “Understanding the Fourier Transform” at #AltDevBlog: http://www.altdevblogaday.com/2011/05/17/understanding-the-fourier-transform/.
CHAPTER 2. PERFORMANCE MEASUREMENTS FROM IMAGES

In the two-dimensional Fourier transform, the output represents the strength in different spatial frequencies in the original image. An image with a single periodic component (say, vertical bands shaded in a gradient that varies sinusoidally) results in a Fourier transformed image with pixels whose values encode the frequency and amplitude of the underlying sine waves. For example, see the image and Fourier transform shown in Figure 2.3a. Here, the image was created by summing a sine wave that completes 2 periods across the image and a sine wave that completes 8 periods across the image from left to right. This created periodicity in the horizontal direction at two frequencies, which shows up in the Fourier-transformed image as four dark points: two for the positive frequency representation, and two for the negative. A more complex example, which adds periodic components in the perpendicular direction, is shown in Figure 2.3b. Here, I have added a vertical periodic component in the form of a sine wave that completes 2 periods across the image from top to bottom. In the resulting Fourier-transformed image, this is represented by the dark points above and below the center point.

The expression for a two-dimensional discrete Fourier transform, as defined in the NumPy
(a) The image on the left varies from left to right as the sum of two sine functions with different periods. The image on the right is the 2D DFT of the image on the left, showing four high-valued pixels (black in the color scale used) representing the positive and negative frequencies present in the gradient.

(b) Introducing a vertical periodic component to the image in Figure 2.3a produces dark points above and below the central point in the Fourier transformed image.

Figure 2.3: Visualizations of the two dimensional Fourier transform
The form of the Fraunhofer equation suggests a convenient way to generate the PSF numerically, using the Fast Fourier Transform, a specialized computer algorithm for computing the Fourier transform numerically. The Fast Fourier Transform (FFT) is a kind of discrete Fourier transform (DFT) that is optimized to run much faster than the so-called “naïve” implementation that translates the formula into code directly. The NumPy library for Python provides an implementation of the FFT in one or more dimensions as part of the `numpy.fft` package. Fortunately for us, the Fast Fourier Transform (FFT) is one of the most common and best-optimized common algorithms for scientific computing.

### 2.2 Creating a Synthetic Point-Spread Function

The procedure to compute a point spread function from an aperture is just a two-dimensional Fourier transform of the aperture function, where our aperture function is defined to be 1 where light is admitted and 0 elsewhere. We construct a numerical aperture array, compute its Fourier transform, compute the absolute value squared for each element of the Fourier transform output array, and scale the resulting image to real, physical dimensions. This gives us an ideal point-spread function that we are trying to recover with our adaptive optics corrections; deviations from this ideal PSF can be measured in our science images.

To implement this algorithm, and handle the inevitable subtleties that arise when trying to put theory into practice, I have used the de-facto standard tools for scientific computing in Python: NumPy and SciPy [4]. The numerical aperture provided as input to the FFT is simply an array with its element values set such that a circular or annular region in the center is filled with ones, while the rest are initialized to zero (Figure 2.4).

There are multiple subtly different definitions of the Fourier transform. In physics and mathematics, it often makes sense to interpret the results of the Fourier transform as running from the smallest (most negative) frequency evaluated through the zero frequency term to the largest (most positive) frequency evaluated. However, the convention in computing is to place the zero frequency term in the lowest order position in the array, then successively greater positive frequencies up to the Nyquist frequency, followed by negative frequencies in order of decreasingly negative frequency. This makes sense for ease of determining constant offset for the input signal (“DC offset”) by merely reading the first element of the output array, but does not result in an image we would recognize as a PSF (Figure
2.2. CREATING A SYNTHETIC POINT-SPREAD FUNCTION

Figure 2.4: Central region of a numerical aperture array for a telescope like TMO. \( r_{\text{primary}} = 40.9 \) pixels, \( r_{\text{obscuration}} = 11.5 \) pixels (Since only the ratio of \( r_{\text{primary}} \) to \( r_{\text{obscuration}} \) is important, we adopt a 1 pixel = 1 inch scale.)

2.5). The solution is then to shift the output array elements into the configuration we use in physics. It is easy to see how this is the appropriate configuration: we would expect the greatest intensity in the constant offset term, so we’d expect it to be in the middle of our image. NumPy provides a function, \texttt{numpy.fft.fftshift}, that does this shift for us. The final, shifted PSF for an on-axis source is shown in Figure 2.6. Several examples of PSFs for different numerical apertures are shown in Figure 2.8.

When moving from analytical solutions to numerical ones, two new concerns are introduced: sampling errors and numerical precision. Fortunately, numerical precision is much less of an issue with modern 64-bit computers. Any error in computing a Fourier transform on an array of 64-bit floating point numbers will be small relative to the uncertainties in our optical parameters like wavelength, \( f \)-number, etc. Sampling presents a problem when trying to ensure we are obtaining enough detail in the region of interest in our PSF. The radius of the first minimum from the central pixel is large for numerical apertures with radii that are small relative to the dimension of the array. The effects of undersampling the central region can be mitigated by choosing a large enough dimension for the input array. (We also tried spline interpolation of the resulting PSF when finding the first minimum, but found negligible differences in first minima from the two methods for a finely-sampled PSF.) For our 1600 \( \times \) 1600 aperture array with \( r_{\text{primary}} = 40.9 \) px and \( r_{\text{obscuration}} = 11.5 \) px, we find a first minimum of \( r = 22 \) px and sufficient detail in the region around the first minimum to be confident we are sampling the PSF finely.
CHAPTER 2. PERFORMANCE MEASUREMENTS FROM IMAGES

Figure 2.5: The output of a 2D FFT of the aperture in Figure 2.4 as produced by \texttt{numpy.fft.fft2} with the engineering/computing conventional frequency ordering, before it is shifted to place the highest intensity in the middle.

Figure 2.6: Computing the $|\text{FFT(Aperture)}|^2$ for an aperture array generates a point spread function array of the same dimensions. Left: The FFT output for an unobscured circular aperture of $r = 40.9$ pixels in a $1600 \times 1600$ array. Right: a detail view of the central region. The image has been square-root scaled to show detail.
2.2. CREATING A SYNTHETIC POINT-SPREAD FUNCTION

Figure 2.7: The central row of the calculated ideal PSF, with the region inside the first minimum \((r = 22\, \text{px})\) shaded. The second, darker shaded region encloses the core of the PSF calculated for an aperture with central obscuration. While the core narrows and loses intensity when central obscuration is introduced, some of the flux is noticeably pushed out to the first Airy ring.
Figure 2.8: Several examples of diffraction patterns produced by different aperture masks. The first row shows an unobscured circular aperture, followed by an aperture with central obscuration, followed by an aperture with central obscuration and obscuration from an (exaggerated) support structure. The image has been square-root scaled to show detail.
2.3 Calculating Strehl Ratios from Observation Data

Simply put, the Strehl ratio is the ratio of peaks between an image of a real point source and an ideal point spread function. The values of the ideal point spread function array are rescaled such that the array contains the same total integrated flux as the real point source at a large radius. (The Strehl ratio calculation is treated in detail in “Is That Really Your Strehl Ratio?” by Roberts, et al. [7]) In addition to scaling the values of the pixels in our PSF array, we also need to rescale it to the physical pixel dimensions of our detector. We set our scale by the first minimum of the computed PSF, the angular size given by Equation 2, and our knowledge of the physical size of each pixel on our detector. As the angular size is dependent on the wavelength, and the physical size of pixels differs between our optical and near-IR cameras, these are configurable options to the aotools code developed as part of this thesis.

Given an image from one of the KAPAO science cameras, one computes a Strehl ratio using the following procedure. First, one uses the pixel scale for the camera to figure out the radius at which to normalize the ideal PSF such that the ideal and observed PSFs have the same total flux. From Roberts, et al. we chose to normalize at a radius of 2.5″, which corresponds to \( r \approx 33 \) pixels for the Andor optical camera (with pixel scale 0.076″/pixel). For the near-IR camera from Xenics, which has pixel scale 0.133″/pixel, an angular radius of 2.5″ corresponds to \( r \approx 19 \) pixels.

We also use the pixel scale to resize our ideal PSF such that one pixel in the ideal PSF array represents one pixel on the detector. We generally choose the dimensions of our aperture and PSF arrays such that we finely sample the region near the first minimum in our computed PSF. However, this results in a PSF array with pixels that are too small compared to pixels on the detector. Resizing the ideal image makes it possible to compare it with science images, and ensures that our photometry routines are treating the ideal PSF image identically to a science image (including any over or underestimation due to inclusion of partial pixels). To avoid the difficulty of accounting for fractional pixels in our simple aperture photometry, the Strehl calculator delegates photometry to phot through PyRAF automation. There is a version of the Strehl calculator that does simple aperture photometry in Python (strehlframe), but in order to approximate the flux in fractional pixels we needed phot. The improved single-frame Strehl calculator is found under the name photstrehlframe in the aotools package.

Rather than develop a Strehl calculator alone, this analysis code ended up including a curve of growth calculator for ideal and observed sources. Ultimately, it is only necessary to compute enclosed flux at two radii (the pixel radius of the first minimum, and the 2.5″ cutoff), but it can be instructive to examine curves of growth for science images. We have set our cutoff for the “core” in the Strehl calculation to the first minimum of the ideal PSF. As this varies with detector and wavelength, it is computed for every invocation of
the Strehl calculator tool.

This code also spawned a “Strehl series” tool called `photstrehlcube`, which automates the process of computing a Strehl ratio for many frames at once. By leveraging DAOfind, the tool centroids the star in each frame, computes a Strehl ratio as described above, and outputs a series of Strehl measurements: one for each frame of a FITS data cube (or movie). This produced data that rather dramatically showed the effects of closed loop operation versus seeing limited observation. For further discussion of these Strehl time-series, see Chapter 4.
Chapter 3

Turbulence Characterization from Wavefront Sensor Telemetry

The atmosphere above Table Mountain Observatory is constantly in flux. (If it were not, building KAPAO would be much easier.) While moment-to-moment measurements and correction are the domain of the KAPAO control loop, it is still important to characterize the turbulence at our site more broadly. KAPAO uses control loop software shared with us by the RoboAO project at Caltech. Using the telemetry recorded by this software as the instrument runs, we attempt to characterize the behavior of the atmosphere at the telescope site.

3.1 Turbulence Power Spectra

In normal operation, wavefront error is measured by finding the slope of the wavefront over each subaperture in both an X and Y direction. Each subaperture is divided into a top half and bottom half (or left half and right half), and the ratio of intensities in the halves is used to calculate the slope of the wavefront over the subaperture. Using the slope measurements made during operation, we can compute a power spectral density of the time-variation in different subapertures. That is to say, by taking the discrete Fourier transform of the slope values recorded over a run, we can see the relative prominence of different frequency components in the variability. Plotting this power spectral density versus the frequencies sampled in the Fourier transform gives us a power spectrum, which we use to look for power law dependencies in the turbulence spectrum that are predicted by John W. Hardy in *Adaptive Optics for Astronomical Telescopes* [2], a classic adaptive optics text.

In chapter nine, “Adaptive Optics Performance Analysis and Optimization”, Hardy derives
a set of power law dependencies for the strength of different frequency components of atmospheric turbulence. Based on Hardy’s analysis, we should observe a power spectrum dominated at low frequencies by a $f^{-2/3}$ frequency dependence, and at higher frequencies by a $f^{-8/3}$ frequency dependence called the Kolmogorov asymptote.

A key component of the Hardy model is that fitting out the tip/tilt component for each frame and looking at the power spectral density for the residual tip/tilt-removed phase should reveal a sharply different power law dependence of $f^{4/3}$ at low frequencies, as well as a “crossover” frequency $f_c$ where the power spectrum is once again dominated by a $f^{-8/3}$ dependence. A schematic plot of these power law dependencies is provided in Figure 3.1, with the caveat that the value of $f_c$ is not a constant but will vary based on local conditions. As $f_c$ allows us to determine the local wind velocity through the relationship $v = f_c D / 0.705$ (Hardy 9.21) and the Greenwood frequency through $f_G = 0.427 v / r_0$ (Hardy 9.55), it is a desirable quantity to calculate for our site.

![Figure 3.1: Schematic plot of predicted power law dependencies of the spectrum of wavefront phase variations due to atmospheric turbulence. Power spectral density is in arbitrary units. (Adapted from Hardy, 1998, Fig. 9.3.)](image)

To collect information on the atmosphere alone, as opposed to information on our instrument’s ability to correct, we must operate the instrument in “zero-gain” mode. This means setting the values of “scalar gain” and “tip/tilt gain” to zero, and operating the system as normal. (The gain values control how aggressively the instrument tries to correct distortion. Very high gain values lead to resonant behavior where the instrument overshoots
the optimal correction.) Functionally, this means that the deformable mirror is configured
to an initial position using a “flat map” and then left alone while the control software
measures the wavefront hundreds or thousands of times each second, writing the resulting
measurements to telemetry files.

![Turbulence power spectrum for the ViLLaGEs instrument at Lick Observatory. The \( f_c \) is estimated to be 7 Hz. (From Morzinski et al., 2010)](image)

We sought to use slope telemetry data to determine \( f_c \), the critical frequency at which
atmospheric turbulence transitions from a 4/3 power law (in tilt-removed phase) to a -8/3
power law. The power spectrum analysis that follows takes its inspiration from a 2010 AO
performance paper for the ViLLaGEs instrument by Morzinski, et al. [6] In their analysis,
they determined the critical frequency for the atmosphere above Lick Observatory using
time-series wavefront measurements from their AO instrument. The plot from which they
estimated \( f_c \) is reproduced here as Figure 3.2.

### 3.2 Telemetry Format Details

Before we discuss the analysis procedure, it is important to state some pertinent facts about
our system and the data products it creates. The RoboAO loop control software creates
several types of telemetry files, as described in Table 3.1. KAPAO measures the incoming
wavefront in 97 subapertures that cover the non-vignetted portion of our wavefront sensor
camera’s image plane (Figure 3.3). These values are used to reconstruct a wavefront and move a subset of the actuators on our MEMS deformable mirror (DM). Our DM has a $12 \times 12$ array of actuators behind a continuous face-sheet mirror surface. Of these, we use only the 121 actuators corresponding to the circular pupil of our telescope.

![Figure 3.3: The arrangement and numbering of KAPAO’s 97 subapertures.](image)

The files themselves are named according to the format “[type]_YYYYMMDD_HHMMSS.tel”, where everything after [type] is a timestamp in 24 hour format in the control computer’s local time zone. Every line begins with a timestamp like “2013-12-17 02:43:22.089911”, followed by the corresponding data at that timestamp.\(^\text{1}\) Each timestep is one line in the file, and the individual fields are separated by ASCII tab characters.

### 3.3 Spatial Power Spectrum

When looking for the strength of different frequency components in turbulence, we start with the measured slopes in each subaperture. The individual subaperture slopes are a measurement of the local wavefront deformation within a region of the pupil plane imaged by our Shack-Hartmann wavefront sensor (WFS). For the details of KAPAO’s wavefront sensor design, see the 2013 senior thesis of Daniel Contreras [1]. Slope measurements are dependent on sufficient intensity in the corresponding subaperture to form a “spot” image on the WFS camera. A configurable parameter (MIN\_LIGHT) in the RoboAO software controls the minimum intensity required in a subaperture before it will calculate slope values for the subaperture. If there is not sufficient light to measure a slope, the control

\(^1\)This timestamp can be parsed by using the format string %Y-%m-%d %H:%M:%S.%f for strftime implementations that support %f for microseconds.
Table 3.1: Types of KAPAO Telemetry Files

<table>
<thead>
<tr>
<th>Filename pattern</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_pos YYYYMDD_HHMSS.tel</td>
<td>121 actuators + tip + tilt + piston in DM units in the range [0, 65535]</td>
</tr>
<tr>
<td>dm_pos YYYYMDD_HHMSS.tel</td>
<td>Internal DM position information. 160 values in the range [0, 65535]. (Only 121 are used.)</td>
</tr>
<tr>
<td>intensity_map YYYYMDD_HHMSS.tel</td>
<td>Intensity values for 97 subapertures</td>
</tr>
<tr>
<td>slope_x YYYYMDD_HHMSS.tel</td>
<td>X slope measurements for 97 subapertures from the WFS, linearized with the configured table between [-1, 1]</td>
</tr>
<tr>
<td>slope_y YYYYMDD_HHMSS.tel</td>
<td>Y slope measurements for 97 subapertures from the WFS, linearized with the configured table between [-1, 1]</td>
</tr>
<tr>
<td>wfs_image YYYYMDD_HHMSS.tel</td>
<td>flattened 26 × 26 (i.e. 1 × 676) image from the WFS camera</td>
</tr>
<tr>
<td>wfs_info YYYYMDD_HHMSS.tel</td>
<td>WFS frame number, dropped frames, bad frame count, and loop timing information (see src/control/WFS/wfs_control.cpp:166-177)</td>
</tr>
</tbody>
</table>

Software will record a zero for that subaperture’s slope value in both the x and the y direction. (The intensity map will still record the system’s measured intensity for the subaperture at that timestep, even if it was too low to calculate slopes.)

### 3.3.1 An Initial Power Spectrum Computation

An initial analysis of the slopes is, in fact, quite straightforward. In the below code listing, we omit the necessary (but not particularly illuminating) process of reading in telemetry files beforehand and producing plots afterwards. Here we compute the discrete Fourier transform of 97 time series (for 97 subapertures) using Python and NumPy. We are assuming we have already read in data and computed dt. For a run with N samples, lasting a total duration of \((N \times \Delta t)\) seconds, the data variable holds an array with dimensions \(N \times (\text{Number of Subapertures})\) and dt is the \(\Delta t\) between samples in seconds, generally computed from the timestamps in the telemetry files. (See Figure ?? for an illustration of this glitch.) Two lines of code are sufficient\(^2\) to compute the power spectral density for all 97 subapertures at once, and one more line averages the power spectrum over the subapertures.

\(^2\)The trick is knowing which two.
import numpy as np

freqs = np.fft.fftfreq(data.shape[0], dt)
power = np.abs(np.fft.fft(data, axis=0))
power_avg = np.average(power, axis=1)

The outputs are \texttt{freqs}, a 1-D array of length \( N \) containing the centers of the FFT frequency bins in Hertz, and \texttt{power}, an array with dimensions \( N \times \text{(Number of Subapertures)} \) containing the magnitudes of the powers in different frequency bins. The array \texttt{power_avg} contains the average power at each frequency, computed by averaging across all subapertures at each timestep.

This simple-but-straightforward computation produces a power spectrum like that seen in Figure 3.4.

![Figure 3.4: Power spectrum of x-direction slope values averaged across all subapertures. The \( f^{-2/3} \) dashed line is not a fit to the data, but is plotted for reference. (Created using data from observations of \( \beta \) Pegasi on December 17, 2013. The data contained 67 seconds of measurements, totaling 13,426 samples at \( \Delta t = 5 \text{ ms} \).) Referring back to Figure 3.1, our goal is to determine the regions in which different power law dependencies hold. This is not a straightforward linear fit to log-scaled data, however, as these fits are expected to hold in different “windows” of frequency. It may also be the case, whether due to unknown systematics or unanticipated difficulties applying the turbulence model, that the power spectrum does not exhibit the predicted dependence for our observations.}
Leaving to a future researcher a more rigorous statistical line-fitting approach, I wrote a routine that attempted a linear fit to the power spectrum by taking 500 points at a time from the log-scaled power distribution and computing the slope of a line fitting these points. In regions with slopes promisingly close to the power law dependencies from Hardy, I adjusted the ranges of points included and refined my fit.

The data exhibited a strong power law dependence. Unfortunately, the powers in the fit were not the characteristic $f^{-2/3}$ and $f^{-8/3}$ predicted. Ruling out a programming error, this points to a deficiency in either model or analysis. To eliminate possible sources of error in the analysis, I placed more stringent criteria on the subapertures and data ranges used, which I will describe.

However, before I discuss refining the analysis procedure, it is worth quantifying the power law dependencies present in the data from this initial analysis. As we eliminate potential sources of systematic error, we may change the power law behavior of our power spectrum. Frequency ranges and associated exponents for the data in Figure 3.4 are shown in Table 3.2. The fits over-plotted on the initial power spectrum are shown in Figure 3.5.

![Figure 3.5: Power spectrum of x-direction slope values averaged across all subapertures from Figure 3.4, over-plotted with power law fits detailed in Table 3.2.](image)

### Table 3.2: Power Law Dependence in Initial Spatial Power Spectrum

<table>
<thead>
<tr>
<th>Frequency range</th>
<th>Scale factor $a$</th>
<th>Exponent $k$</th>
<th>Correlation coefficient $(r$-value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 5 Hz</td>
<td>17.9</td>
<td>-0.44</td>
<td>-0.97</td>
</tr>
<tr>
<td>5 – 30 Hz</td>
<td>24.9</td>
<td>-0.61</td>
<td>-0.96</td>
</tr>
<tr>
<td>30+ Hz</td>
<td>48.5</td>
<td>-0.83</td>
<td>-0.97</td>
</tr>
</tbody>
</table>
3.3.2 Refining the Power Spectrum Computation

The first and most obvious step to improve our ability to extract a measurement from our power spectra is to select our subapertures more carefully before averaging. Our data set suffers from a wandering of the intensity pattern. The RoboAO software zeros out slopes when there is insufficient light to make a measurement, but this determination is made on a frame by frame basis. As a result, the slopes for certain subapertures may be locked to zero for most of a run, but take a few seconds to vary wildly between -1 and 1.

This “zeroing out” is apparent when plotting slope values for a particular subaperture over the course of a whole run. In Figure 3.6, we plot the measured slope values for a subaperture we believe to be well-illuminated over the whole run, as well as the slope values for a subaperture that is inconsistently illuminated over the run. The latter exhibits stretches where the slope value is set to zero artificially, which indicates that we should exclude it from the power spectral density calculations.

Figure 3.6: Plot of measured slope values across an entire run for a good (#49, in green) and bad (#22, in red) subaperture. The RoboAO loop software sets slopes to zero when there is insufficient intensity, resulting in the horizontal lines in parts of the timeseries for the excluded subaperture.

Mysteriously, plotting these slope values with their associated timestamps reveals an issue with timestamps in the RoboAO telemetry. Apparently, time does not always increase: intermittently, lines of telemetry will be recorded that are offset backwards in time by approximately one second from the previous timestep. The line following will recover, showing the expected timestamp by moving $\approx 1$ second plus one timestep $\Delta t$ forward in time. This appears in plots like Figure 3.6 as kinks in the timeseries where points are connected in an order other than left to right. However, as the timestamps are not used for
calculation of the power spectrum (only to find $\Delta t$ between iterations), and only very few such defects in the data exist (see Figure 3.7) we can safely ignore them for the purposes of this analysis. Even if the problem is not limited to timestamp recording (that is, if it affects slope values recorded), there are only $\sim 10$ such defects in this data set, which makes it unlikely that they would affect our power spectrum. We will notify RoboAO and attempt to diagnose the problem with them.

Figure 3.7: Plot showing timestamps jumping backwards in time intermittently. Timestep index on the X axis is just the line number from the telemetry file. Time elapsed shows the difference between the time at the start of the run and that line’s timestamp.

Among the telemetry listed in Table 3.1 can be found the intensity map. These are the measured per-subaperture intensities at each iteration of the loop. By filtering subapertures such that you only retain those whose intensities are high enough for good measurements, one can eliminate those subapertures that report artificial zero slopes.

Figure 3.8a shows the subset of “good” subapertures for our $\beta$ Pegasi data, and Figure 3.8b shows an example frame of data from the intensity map. Here we have identified a set of “good” subapertures by calculating the subaperture’s median intensity over the whole run and taking the top 50. An example frame from the intensity map (Figure 3.8b) shows the poor alignment of the central obscuration with the center of the wavefront sensor. This is a result of misalignment that happened during instrument transport, and was constant throughout the instrument’s operation. Future observations will benefit from improved alignment between these components.

Figure 3.9 shows a comparison between the naive power spectrum including all subapertures and the refined calculation that averages only power in the top 50 subapertures by median
CHAPTER 3. TURBULENCE CHARACTERIZATION

(a) The top 50 subapertures by median intensity over the whole run are part of an “include mask” that is used in the refined PSD calculation. The subapertures shaded in green are included in the calculation, while the rest are excluded.

(b) A single frame from the intensity map showing the telescope central obscuration and the alignment issue described in the text.

Figure 3.8: Visualizations of the subaperture include mask and intensity map.

intensity values. Evidently, the unexpected flatness in our low frequency power spectrum was not due to biasing from incompletely illuminated subapertures included in the power spectrum, as comparing the power spectrum to the naive power spectrum in Figure 3.4 shows very little difference.

By determining our mask for good subapertures simply by taking the top 50 subapertures by median intensity, we risk sweeping fluctuations in intensity over the run under the rug. Suppose, for instance, that a subaperture had higher than average intensity, but telescope drift resulted in occasional gaps where the slope values were zeroed out. This subaperture might not be excluded by the original cutoff, even though it produced bad data. To get a better picture of how well our median intensity cutoff served to exclude bad subapertures, we returned to the control software to get the configured MIN_LIGHT value for this run. Using this cutoff and the intensity map at each timestep, we could compute for each subaperture the fraction of timesteps that produced accurate slopes. If our median masking worked well, plotting the fraction of timesteps that yielded accurate slopes over the subaperture grid should show a fraction near unity for those subapertures included in the mask. Comparing Figure 3.10 to Figure 3.8a shows that, with the exception of a couple of marginal subapertures in the central obscuration pattern, our median intensity-based threshold did a good job of rejecting subapertures with low intensity.

At this point, we went back to our code to ensure we were not miscalculating the power
3.3. SPATIAL POWER SPECTRUM

Figure 3.9: Power spectrum of x-direction slope values averaged across the “good” sub-apertures shown in Figure 3.8a.

Figure 3.10: Image showing the fraction of timestamps for which each subaperture had sufficient intensity for a slope to be measured. A value near unity means that few or no timesteps contained artificial zero values in the X and Y slopes for that subaperture.
spectrum. This was verified in two ways: first, the creation of a synthetic power spectrum exhibiting a power law frequency dependence; and second, smoothing the time series data by convolving it with a boxcar kernel in order to remove higher frequency components and ensuring that the effects are apparent in the power spectrum. Both of these tests confirmed the correctness of our power spectrum computation code.

### 3.3.3 Tilt-subtraction

A key part of Hardy’s power laws for turbulence is the presence of a dominant tip/tilt component at low frequencies. This makes sense from consideration of the sizes of turbulent cells of air in the atmosphere: larger cells take longer to move across the sky, and because they are larger we see only part of them and thus pick up a low-order (tip/tilt) contribution from them. The overall slope is determined by either averaging or taking the median slope value at each timestep across the “good” subapertures. The expectation is that a time series of median or mean slopes will roughly track the long-timescale behavior of the slope fluctuations in all subapertures.

Shown in Figure 3.11a is a plot of the top 40 good subapertures’ slope values over time with the median slope value for each timestep overplotted. Subtracting the median slope from each subaperture at each timestep produces the flattened slope data shown in Figure 3.11b. Cross-referencing the median slopes with a scatter plot of the measured optical speckle pattern centroids across the run (Figure 3.12) shows that the abrupt jumps in Y are likely an artifact of telescope tracking. Here the Y direction for both the slopes and the science images from the Andor camera means the declination direction in terms of telescope pointing.

Referring to Figure 3.2 reproduced from Morzinski et al., we expect the tilt-subtracted power spectrum to exhibit a “flattened” behavior. However, when subtracting the median (or mean, we did not find a notable difference) slope across good subapertures at each time step and computing the power spectral density of the residuals, we see an almost-identical power spectrum. The power in the overall slope at different frequencies for X and Y slopes is shown in Figure 3.13, and the power spectrum of the residual slopes averaged over good subapertures is shown in 3.14. Though the overall slope varies at low frequencies in a power-law-dependent manner, it is not the dominant component of the power spectral density, and subtracting overall slopes at each timestep does not noticeably change the average power spectrum for good subapertures.
3.4 Turbulence Power Spectra

Our take on Morzinski’s turbulence power spectrum plot (Figure 3.2) is given in Figure 3.15, which should look familiar from the previous chapter. Unfortunately, it is readily apparent from the plot that there is little hope of fitting a $4/3$ power law to any portion of the residual power spectrum. Thus, we are forced to conclude that the data we have in hand does not exhibit the power law dependence predicted by Hardy. Unfortunately, this places a roadblock in the way of computing turbulence parameters ($r_0, v, f_G, \tau_0$) for the TMO site, several of which depend on knowing the critical frequency at which tip-tilt effects from turbulence no longer dominate the power spectrum.

Our analysis showed no correlation in the low-frequency behavior of wavefront fluctuations across subapertures. Without further work it is impossible to say whether our results are due to faults in our telemetry-taking, the peculiarities of the one zero-gain run we have in hand for analysis, or some other aspect of the TMO site and KAPAO instrument that make Hardy’s model a poor model for our case. Future researchers searching for the predicted behavior are advised to first and foremost take additional zero-gain data on-sky, and analyze it as described in Chapter 3 to find power law dependencies. Should additional on-sky data not be sufficient to constrain the problem, a possible laboratory troubleshooting technique has been proposed.

To recreate tip/tilt-dominated power spectra in lab and verify that the RoboAO telemetry will show the expected correlation across subapertures at low frequencies, one can manually drive the tip-tilt mirror around by hand at low frequencies while introducing distortion with a phase screen. In the resulting telemetry, low frequency power in slope variations should be correlated across all subapertures, and should subtract out nicely by removing the overall tip-tilt component as we attempted for our power spectra.
(a) In color, the slopes of 40 “good” subapertures at each timestep are plotted with transparency to reveal overall trends. In white, the median slope at each timestep is plotted.

(b) Here we have subtracted the median slope plotted in white above, flattening the slope data.

Figure 3.11: Subtracting the overall tip/tilt in subaperture slopes
3.4. TURBULENCE POWER SPECTRA

Figure 3.12: Scatter plot of measured speckle pattern centroids from the Andor (optical) science camera. The scatter in the Y direction on these science images corresponds to telescope tracking jitter in the declination direction. (Compare to Y slope values in Figure 3.11a.)

Figure 3.13: The power in the overall slope at different frequencies for X and Y slopes overplotted on the power spectrum for a single (good) subaperture.
Figure 3.14: Power spectrum of the tilt-subtracted residual slopes averaged over good subapertures.

Figure 3.15: Power spectrum of the tilt-subtracted residual slopes averaged over good subapertures.
Chapter 4

Performance Measurements

4.1 Introduction and Summary of Observing Run

KAPAO Prime had first light during Summer 2013, and on-sky operation continued with a run last December. First light observations were intended to validate the summer engineering work and get initial observations with closed-loop AO correction with the updated instrument.\(^1\)

Our second KAPAO Prime observing run took advantage of our Fall engineering work, and was conducted over several days in December 2013. Our observations from this run contained simultaneous Sloan \(z'\)-band and H-band observations with our visible and near-IR cameras (first light observations were unfiltered). This observing run also allowed us to refine the procedures for target acquisition and guiding, which must necessarily be worked out while the instrument is installed on the telescope. During this run we observed the following targets: \(\beta\) Andromedæ, \(\beta\) Gemini, \(\alpha\) Gemini, Io, and \(\eta\) Piscium. The analysis below was performed with the image analysis tools developed for KAPAO called aotools (see additional documentation in Appendix B).

For the image analysis below, we used exposures of \(\eta\) Piscium (V=3.62), a double star in Pisces. The original goal was to verify the plate scale of KAPAO Prime through imaging of \(\eta\) Piscium’s faint companion at a separation of \(\approx 1''\). However, our science images do not show the fainter companion, so we instead used the data to characterize the PSF with and without adaptive optics correction, as well as measuring instrument performance through frame-by-frame Strehl ratio time series.

\(^1\)This summary and some of the material in this chapter originally appeared as part of a poster by the author presented at AAS 223.
(a) **Near-infrared**: Side-by-side comparison of open/seeing-limited and AO lucky images of η Piscium in near-IR H-band.

(b) **Optical**: Side-by-side comparison of open/seeing-limited and AO lucky images of η Piscium in optical Sloan $z'$ band.

Figure 4.1: Images of η Piscium in near-infrared (H-band) and optical (Sloan $z'$ band) from December 2013 observations.
4.2 Curves of Growth

Curves of growth were calculated for near-IR and optical band images using the aotools photstrehlframe task. Using different subsets of frames from our observations, we were able to compare open-loop, closed-loop, and AO lucky PSFs with an ideal PSF. The ideal point spread function was computed with the procedure discussed in Chapter 2 for wavelengths in the middle of the near-IR H and Sloan $z'$ passbands (913 nm and 1650 nm, respectively).

“Open” images simulated a seeing limited observation by coadding the uncorrected frames from the beginning of the FITS data cube (before the observer closed the AO loop). The “AO Closed” image is a single frame chosen from the AO-corrected frames of the data cube. (We found that our tip-tilt correction performance was not by itself sufficient to prevent smearing of the PSF; simply coadding a series of AO-corrected frames actually reduced the measured Strehl ratio compared to the Strehl ratio calculated for any single AO-corrected frame.)

In order to push the limits of the Strehl achievable with KAPAO Prime, we performed post-processing on images that combined “lucky imaging” techniques with AO correction. Rather than the long exposures more typical of astronomical imaging, our science cameras were configured to take images with very short integration times. As in lucky imaging, this gave us snapshots where the atmosphere was more quiescent. The difference is that our AO system corrected the wavefronts in these more quiescent periods, giving us images that are both “lucky” and AO corrected. By splitting the data cube and sorting frames by their peak flux values, a simple cutoff can be employed to take only the $N$ luckiest images from a particular observation. These lucky frames are then shifted and coadded, correcting any residual tip and tilt variation from frame to frame.

In the text and figures, when we refer to “AO lucky” frames, we mean frames processed in this manner: captured with short integration times while the AO loop is running, selected by peak flux values, and aligned to remove residual tip-tilt error. A demonstration of the dramatic effects of the adaptive optics correction is visible in the side-by-side image comparisons of Figure 4.1a (near-IR) and Figure 4.1b (optical). Despite the preceding excursion into the specifics of AO lucky images, it is worth noting that the vast majority of the improvement in resolution is due to the AO system (not our post-processing). For confirmation of this, look no further than the discussion of curves of growth in Section 4.2.

The bulk of the Strehl improvement is, as expected, due to the AO loop. However, it is interesting to note that AO lucky processing gives a greater boost to the Sloan $z'$ exposures, despite the individual AO closed frame Strehl being lower in $z'$ than H-band. This could be due to a combination of a larger PSF core in the near infrared and larger pixels in
(a) **Near-infrared:** Curve of growth for an ideal PSF, compared to an open loop/seeing-limited frame, a typical closed frame, and an AO lucky frame from the near-IR camera in H-band.

(b) **Optical:** Curve of growth for an ideal PSF, compared to an open loop/seeing-limited frame, a typical closed frame, and an AO lucky frame from the visible light camera in Sloan $z'$ band.

Figure 4.2: Visualizations of imaging performance on $\eta$ Piscium from December 2013 observations.
4.3 STREHL RATIOS

the infrared science camera resulting in individual frames that are more forgiving of short-
timescale tip-tilt drift. Though AO lucky performance in near-IR lags slightly behind optical, Strehl ratios for near-IR images pre-lucky-processing are generally higher.

4.3 Strehl Ratios

Using the Strehl ratio and Strehl time-series tools described in Chapter 2, we analyzed the data we took on η Piscium to characterize instrument performance. The Strehl measurements made gave us an idea of how AO-corrected performance compared to both open-loop operation (i.e. seeing-limited observation) and speckle imaging, where one uses short integration times to freeze turbulence but does not use an adaptive optical correction. Figures 4.3 and 4.4 show Strehl values over time, clearly depicting the point where the instrument’s adaptive optical correction is switched on.

As in the curve of growth analysis, the open loop Strehl ratios noted on the plots are obtained by analyzing a coadded image made from the frames taken without any AO correction. The speckle pattern mean Strehl is the mean instantaneous Strehl measurement from the individual frames taken without AO correction. The Strehl timeseries for these first frames provides an illustration of the value of lucky imaging in the absence of AO: without any sophisticated correction, random changes in the atmosphere can deliver a Strehl ratio of 0.2 or more. The beginning of the AO loop is clearly visible as a step up in Strehl values, and the noted AO closed loop mean Strehl is simply the mean over those frames taken after the loop closes. (The frames that were taken at the very beginning of the loop closing are not included in either average.)
CHAPTER 4. PERFORMANCE MEASUREMENTS

Figure 4.3: Plot of near-IR (H-band) Strehl ratios computed frame-by-frame for observations of η Piscium. Integration time was 0.02 seconds per frame.

Figure 4.4: Plot of optical (Sloan z′ band) Strehl ratios computed frame-by-frame for observations of η Piscium. Integration time was 0.05 seconds per frame.
Appendices
Appendix A

Code Listings

Unless otherwise noted, these code listings depend on NumPy and Matplotlib. Some code listings may additionally depend on the Python interface to PyRAF (the STScI extension for the IRAF CL), or scientific computing routines from SciPy.

A.1 Fourier Transform Explanation

This code computes the Fourier transform of a signal we create by summing two sine waves of differing angular frequencies, and generates Figures 2.1 and 2.2.

```python
from matplotlib import pyplot
import numpy

# Number of sample points
N = 1200
# sample spacing
dt = 1.0 / 80.0
t = numpy.linspace(0.0, N*dt, N)

# create our two periodic components, a and b
a_amplitude, a_angular_freq = 1.0, 2*numpy.pi*1.0
b_amplitude, b_angular_freq = 0.25, 2*numpy.pi*4.0
a = a_amplitude * numpy.sin(a_angular_freq * t)
b = b_amplitude * numpy.sin(b_angular_freq * t)

# create the signal by summing the contributions of the two components
```

43
signal = a + b

# plot the signal
pyplot.plot(t, a, '−−', alpha=0.5, label="$A\sin(\omega_a t)\$")
pyplot.plot(t, b, '−−', alpha=0.5, label="$B\sin(\omega_b t)\$")
pyplot.plot(t, signal)
pyplot.legend()
pyplot.xlim(0, 2)
pyplot.ylim(-1.8, 1.8)
pyplot.xlabel('Time(t)')
pyplot.ylabel('Amplitude')
pyplot.savefig('fft_explain_signal.pdf', bbox_inches='tight')

# compute the Fourier transform of the signal
signal_fft = numpy.fft.fft(signal)
signal_freqs = numpy.linspace(0.0, 1.0/(2.0*dt), N/2)

# plot the Fourier transform of the signal
pyplot.figure()
pyplot.plot(signal_freqs, 2.0/N * numpy.abs(signal_fft[0:N/2]))
pyplot.ylim(0, 1.05)
pyplot.yticks(numpy.arange(0, 1.25, 0.25))
pyplot.xlim(0, 10)
pyplot.grid()
pyplot.xlabel("Angular\ Frequency_c (in units of \ 2\pi $)")
pyplot.ylabel("Power")
pyplot.savefig('fft_explain_transform.pdf', bbox_inches='tight')

A.2 Spatial Frequencies in One Direction and the 2D Fourier Transform

This code creates Figure 2.3a by generating a 64 × 64 array with a sum of two sine waves repeated on every row. Much of the code is taken up with arranging the plot attractively; the key lines are all before the first call to plt.subplot.

import numpy as np
from matplotlib import pyplot as plt
from matplotlib.axes_grid1 import make_axes_locatable
img = np.zeros((64, 64))
# sine wave going from 0 to 4pi in 64 steps
img[:] = np.sin(np.linspace(0, 4*np.pi, 64))
# add sine wave going from 0 to 16pi in 64 steps
img[:] += np.sin(np.linspace(0, 16*np.pi, 64))

# Fourier transform and shift zero-frequency pixel to center
shifted = np.fft.fftshift(np.abs(np.fft.fft2(img)))

# Plot image and its DFT using gray_r (darker is greater)
ax = plt.subplot(121)
plt.imshow(img, cmap="gray_r")

ax2 = plt.subplot(122)
plt.imshow(shifted, "gray_r")

# Add schematic plot above showing underlying sine wave
divider = make_axes_locatable(ax)
topax = divider.append_axes("top", size="10\%", pad=0.05)
topax.plot(img[32])
topax.set_xlim(0, 63)
topax.set_xticklabels([])
topax.set_yticks([])
topax.set_ylim(-3.5, 3.5)

# Add schematic plot above showing power spectrum
divider2 = make_axes_locatable(ax2)
topax2 = divider2.append_axes("top", size="10\%", pad=0.05)
topax2.plot(shifted[32])
topax2.set_xlim(0, 63)
topax2.set_xticklabels([])
topax2.set_yticks([])
A.3 Spatial Frequencies in Two Directions and the 2D Fourier Transform

This code creates Figure 2.3b by generating a $64 \times 64$ array with a sum of two sine waves repeated on every row, and a vertical periodic component that contains a single sine wave. As before, much of the code is taken up with arranging the plot attractively; the key lines are all before the first call to `plt.subplot`.

```python
import numpy as np
from matplotlib import pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

img = np.zeros((64,64))
# add horizontal sine wave going from 0 to 4pi in 64 steps
img[:] = np.sin(np.linspace(0, 4*np.pi, 64))
# transpose the image, making that a vertical component
img = img.T

# add the horizontal components as before
img[:] += np.sin(np.linspace(0, 4*np.pi, 64))
img[:] += np.sin(np.linspace(0, 16*np.pi, 64))

# Fourier transform and shift zero-frequency pixel to center
shifted = np.fft.fftshift(np.abs(np.fft.fft2(img)))

# Plot image and its DFT using gray_r (darker is greater)
ax = plt.subplot(121)
plt.imshow(img, cmap="gray_r")

ax2 = plt.subplot(122)
plt.imshow(shifted, cmap="gray_r")

# Add schematic plot above showing underlying sine wave
divider = make_axes_locatable(ax)
topax = divider.append_axes("top", size="10%", pad=0.05)
topax.plot(img[32])
topax.set_xlim(0,63)
topax.set_xticklabels([])
topax.set_yticks([])
topax.set_ylim(-3.5, 3.5)
```
# Add schematic plot above showing power spectrum

divider2 = make_axes_locatable(ax2)
topax2 = divider2.append_axes("top", size="10\%", pad=0.05)
topax2.plot(shifted[32])
topax2.set_xlim(0,63)
topax2.set_xticklabels([])
topax2.set_yticks([])
Appendix B

aotools PyRAF Package

While conducting this research, it was necessary to develop several new pieces of software to perform the science image-based Strehl calculation. These were created originally as Python functions in IPython notebooks, but for reusability it was decided to encapsulate them as a PyRAF package.

The most up-to-date documentation for them can be found at: https://github.com/josephoenix/aotools. For completeness’ sake, I have included a copy of the documentation below (current as of May 7, 2014).

B.1 Installation

To install and load by default, create a loginuser.cl file (in the folder with your login.cl file) with the following content:

```plaintext
reset aotools = /Users/josephoenix/Dropbox/Software/Joseph/aotools/
task aotools.pkg = aotools$aotools.cl
        aotools
```

Edit the path on the `reset aotools` line to point to this folder in your installation. Don’t forget the trailing slash!

49
B.2 Usage

B.2.1 aoavgcube

Note: should really be named aosumcube.

Uses imcombine and aotools.util.combine_cube_frames to recombine a datacube in such a way that it creates a new cube with frames of a greater integration time. For example, 100 frames of 0.1 seconds could be processed to create a 10 frame cube where each frame covers 1 second of integration time.

Developer note: Under the hood, this is just calling imcombine to break up and reassemble the cube. Could be reimplemented using only PyFITS, since we aren’t using any fancy rejection features.

Parameters:

- cubefile: Path to a FITS file with a data cube in the first extension (prompted every time)
- outfile: Name of output file (prompted every time)
- fromidx: Frame number (1-indexed) at which to start combining (default: 1, remembered between invocations)
- toidx: Frame number (1-indexed) at which to end combining (if < fromidx, use whole cube) (default: -1, remembered between invocations)

B.2.2 cubeflatfield

Divides every frame of a FITS data cube by a given single-frame FITS image.

Parameters:

- infile: Path to FITS cube
- flatfile: FITS file containing image to divide by
- outfile: Path to FITS cube output

B.2.3 cubemedian

Generates a median frame from a FITS data cube and outputs it as a FITS file with a user-specified EXPOSURE time header.
B.2. USAGE

Parameters:
- `infile`: Path to FITS cube
- `outfile`: Path for output FITS file
- `exposure`: Exposure in seconds (to be stored in FITS header)

B.2.4 cubestack

Stacks one or more ranges of frames from a data cube using `imcombine` to sum them. Useful when you know, e.g., frames 10-40 are open loop operation and 60-90 are closed loop. Then you can run this task with the range specification “10-40,60-90” and get two FITS files containing the summed open and closed frames, respectively.

Developer note: Under the hood, this too is just calling `imcombine` to break up and reassemble the cube.

Parameters:
- `cubefile`: Path to a FITS file with a data cube in the first extension (prompted every time)
- `rangespec`: Comma-separated one-indexed ranges to combine (e.g. “1-3,6-10”) (prompted every time)
- `clobber`: currently unused (default: False, remembered between invocations)

B.2.5 cubetoframes

Explodes a FITS data cube into a folder with one .fit file for each frame. Useful in development, or when you are just sick and tired of data cubes and want to blink between two frames in ds9.

Parameters:
- `cubefile`: Path to a FITS file with a data cube in the first extension (prompted every time)
- `rangespec`: Comma-separated one-indexed ranges to split into frames (e.g. “1-3,6-10”) (prompted every time)
- `clobber`: When True, delete the `{cubefile}.frames/` directory if it exists before running. When False, exit with an error rather than clobber an existing directory. (default: False, remembered between invocations)
B.2.6 findbright

A quick and dirty wrapper around the daofind brightest function used in the strehlframe and strehlcube tasks. Prints out the location of the brightest source in the input image, as determined by daofind.

Parameters:

- **image**: Path to a FITS file to analyze
- **fwhmpsf**: FWHM in pixels (initial guess), passed on to daofind (default: 2.5 px, remembered between invocations)
- **threshold**: Threshold for detection in sigma, passed on to daofind (default: 20.0 sigma, remembered between invocations)

B.2.7 photstrehl

Computes a time series of Strehl measurements for an entire FITS data cube (or specified ranges). This process is described in more detail under strehlframe below. Aside from operating on data cubes, the other major difference is that there is (currently) no way to disable the daofind-powered auto-centroiding that detects the source in the frame. (On the plus side, that means that tracking / tip-tilt wander will not cause totally bogus Strehl measurements. On the down side, if daofind is way off on one or more of the frames, you can’t correct it for that frame.)

**Note**: This differs from strehlcube mainly in using phot for “accurate” sub-pixel photometry at small radii. True fractional pixel intersections could probably be implemented, but phot gets us closer than our previous Curve of Growth integration code.

Parameters:

- **cubefile**: Path to a FITS file with a data cube to analyze in the first extension (prompted every time)
- **rangespec**: Comma-separated one-indexed ranges to analyze (e.g. “1-3,6-10”) (prompted every time)
- **primary**: Primary mirror diameter (same units as secondary) (default: 40.9 inches, remembered between invocations)
- **secondary**: Secondary mirror diameter (same units as primary) (default: 11.5 inches, remembered between invocations)
• **dimension**: Dimension of intermediate PSF array (default: 1600, remembered between invocations)

• **f_number**: Adjusted f number for image (f_Andor: 34.875, f_Xenics: 46.5, remembered between invocations)

• **pixel_scale**: Pixel scale in mm (Andor: 0.013 mm, Xenics: 0.03 mm, remembered between invocations)

• **lambda_mean**: Mean wavelength in nm (default: 800 nm, remembered between invocations)

• **growth_step**: Pixel radius increment step for curve of growth (default: 1 px, remembered between invocations)

• **normalize_at**: Pixel radius at which all flux is contained (default: 0, meaning calculate 2.5” in px)

• **fwhmpsf**: FWHM in pixels (initial guess), passed on to daofind (default: 2.5 px, remembered between invocations)

• **threshold**: Threshold for detection in sigma, passed on to daofind (default: 20.0 sigma, remembered between invocations)

• **quiet**: Silence debugging messages (specifically curve of growth radius step information) (default: True, remembered between invocations)

### B.2.8 photstrehlframe

**Note:** This differs from strehlframe mainly in using phot for “accurate” sub-pixel photometry at small radii. True fractional pixel intersections could probably be implemented, but phot gets us closer than our previous Curve of Growth integration code.

**Parameters:**

• **image**: Path to a FITS image to analyze (prompted every time)

• **rangespec**: Comma-separated one-indexed ranges to analyze (e.g. “1-3,6-10”) (prompted every time)

• **primary**: Primary mirror diameter (same units as secondary) (default: 40.9 inches, remembered between invocations)

• **secondary**: Secondary mirror diameter (same units as primary) (default: 11.5 inches, remembered between invocations)
• dimension: Dimension of intermediate PSF array (default: 1600, remembered between invocations)

• f_number: Adjusted f number for image (f_Andor: 34.875, f_Xenics: 46.5, remembered between invocations)

• pixel_scale: Pixel scale in mm (Andor: 0.013 mm, Xenics: 0.03 mm, remembered between invocations)

• lambda_mean: Mean wavelength in nm (default: 800 nm, remembered between invocations)

• growth_step: Pixel radius increment step for curve of growth (default: 1 px, remembered between invocations)

• normalize_at: Pixel radius at which all flux is contained (default: 0, meaning calculate 2.5" in px)

• find_source: Detect and automatically centroid brightest source with daofind, ignoring xcenter and ycenter values (default: True, remembered between invocations)

• xcenter: Bright source X Center (column), if not auto-centroiding (default: 511.5 px, remembered between invocations)

• ycenter: Bright source Y Center (row), if not auto-centroiding (default: 511.5 px, remembered between invocations)

• fwhmpsf: FWHM in pixels (initial guess), passed on to daofind (default: 2.5 px, remembered between invocations)

• threshold: Threshold for detection in sigma, passed on to daofind (default: 20.0 sigma, remembered between invocations)

• quiet: Silence debugging messages (specifically curve of growth radius step information) (default: True, remembered between invocations)

B.2.9 pngtocube

Turn a sequence of numbered PNG files into a FITS cube. Xenics data are output as a series of PNGs with an incrementing integer index at the end of the filename. Specify a pattern for the filename, with $i$ in place of the integer index, and this task will collect all the matching frames and turn them into a data cube.

Parameters:
B.2. USAGE

- **directory**: Directory with grayscale PNG images (‘.’ for current directory)
- **filepattern**: Pattern for image filenames (Use $i for the index. Example: `target_foo$i.png`)
- **outfile**: Path and filename for destination FITS file
- **exposure**: Exposure in seconds (to be stored in FITS header)

B.2.10 pngtofits

Convert a grayscale PNG image (e.g. from the Xenics camera) to a FITS file. (Note: counts are not calibrated!)

**Parameters:**

- **infile**: Path to grayscale PNG image
- **exposure**: Exposure in seconds (to be stored in FITS header)

B.2.11 removeband

Computes an “average row” for the image and subtracts it off the image row-by-row to remove a banding artifact. The options `exclude_from` and `exclude_to` determine which rows are excluded from the averaging. (Generally you want to exclude the rows that contain the star.)

- **cubefile**: Path to a FITS frame or cube to analyze
- **outfile**: Path to write the corrected frame to
- **exclude_from**: Row at which to begin excluded range
- **exclude_to**: Row at which to end excluded range

B.2.12 strehlcube

Computes a time series of Strehl measurements for an entire FITS data cube (or specified ranges). This process is described in more detail under `strehlframe` below. Aside from operating on data cubes, the other major difference is that there is (currently) no way to disable the `daofind`-powered auto-centroiding that detects the source in the frame. (On the plus side, that means that tracking / tip-tilt wander will not cause totally bogus Strehl measurements. On the down side, if `daofind` is way off on one or more of the frames, you can’t correct it for that frame.)
Parameters:

- **cubefile**: Path to a FITS file with a data cube to analyze in the first extension (prompted every time)
- **rangespec**: Comma-separated one-indexed ranges to analyze (e.g. "1-3,6-10") (prompted every time)
- **primary**: Primary mirror diameter (same units as secondary) (default: 40.9 inches, remembered between invocations)
- **secondary**: Secondary mirror diameter (same units as primary) (default: 11.5 inches, remembered between invocations)
- **dimension**: Dimension of intermediate PSF array (default: 1600, remembered between invocations)
- **f_number**: Adjusted f number for image (f_Andor: 34.875, f_Xenics: 46.5, remembered between invocations)
- **pixel_scale**: Pixel scale in mm (Andor: 0.013 mm, Xenics: 0.03 mm, remembered between invocations)
- **lambda_mean**: Mean wavelength in nm (default: 800 nm, remembered between invocations)
- **growth_step**: Pixel radius increment step for curve of growth (default: 1 px, remembered between invocations)
- **fwhmpsf**: FWHM in pixels (initial guess), passed on to `daofind` (default: 2.5 px, remembered between invocations)
- **threshold**: Threshold for detection in sigma, passed on to `daofind` (default: 20.0 sigma, remembered between invocations)
- **quiet**: Silence debugging messages (specifically curve of growth radius step information) (default: True, remembered between invocations)

### B.2.13 strehlframe

Parameters:

- **image**: Path to a FITS image to analyze (prompted every time)
- **rangespec**: Comma-separated one-indexed ranges to analyze (e.g. "1-3,6-10") (prompted every time)
B.2. USAGE

- **primary**: Primary mirror diameter (same units as secondary) (default: 40.9 inches, remembered between invocations)
- **secondary**: Secondary mirror diameter (same units as primary) (default: 11.5 inches, remembered between invocations)
- **dimension**: Dimension of intermediate PSF array (default: 1600, remembered between invocations)
- **f_number**: Adjusted f number for image (f_Andor: 34.875, f_Xenics: 46.5, remembered between invocations)
- **pixel_scale**: Pixel scale in mm (Andor: 0.013 mm, Xenics: 0.03 mm, remembered between invocations)
- **lambda_mean**: Mean wavelength in nm (default: 800 nm, remembered between invocations)
- **growth_step**: Pixel radius increment step for curve of growth (default: 1 px, remembered between invocations)
- **find_source**: Detect and automatically centroid brightest source with daofind, ignoring xcenter and ycenter values (default: True, remembered between invocations)
- **xcenter**: Bright source X Center (column), if not auto-centroiding (default: 511.5 px, remembered between invocations)
- **ycenter**: Bright source Y Center (row), if not auto-centroiding (default: 511.5 px, remembered between invocations)
- **fwhmpsf**: FWHM in pixels (initial guess), passed on to daofind (default: 2.5 px, remembered between invocations)
- **threshold**: Threshold for detection in sigma, passed on to daofind (default: 20.0 sigma, remembered between invocations)
- **quiet**: Silence debugging messages (specifically curve of growth radius step information) (default: True, remembered between invocations)
Bibliography


