Computing human optical point spread functions

Andrew B. Watson NASA Ames Research Center, Moffett Field, CA, USA

There is renewed interest in the role of optics in human vision. At the same time there have been advances that allow for routine standardized measurement of the wavefront aberrations of the human eye. Computational methods have been developed to convert these measurements to a description of the human visual optical point spread function (PSF), and to thereby calculate the retinal image. However, tools to implement these calculations for vision science are not widely available or widely understood. In this report we describe software to compute the human optical PSF, and we discuss constraints and limitations.

Introduction

Vision begins with a retinal image formed by the optics of the eye. Many important aspects of visual performance, such as acuity and contrast sensitivity, are highly dependent upon the state of the visual optics. To model this performance, or spatial vision more generally, we must be able to compute the retinal image for a given source image and a given state of the eye optics (Watson & Ahumada, 2008, 2012).

The shape of the wavefront error at the exit pupil defines the optical performance of any optical system, including the human eye (Goodman, 2005). This error is usually described as a collection of wavefront aberrations, and standardized methods to describe these aberrations have been developed (Thibos, Applegate, Schwiegerling, & Webb, 2002). The wavefront error function (commonly "wavefront aberration") over the pupil is characterized as the sum of a set of weighted Zernike polynomials

$$w(x',y') = \sum_{n,m} c_n^m z_n^m(x',y') \quad (1)$$

where x' and y' describe coordinates relative to the pupil center and normalized by the pupil radius. Each polynomial z_n^m is indexed by an order n and a frequency m. The collection of weights, or Zernike coefficients c_n^m, usually expressed in micrometers, serve as the standard description of the aberrations.

The wavefront aberrations can be used to compute the optical point spread function (PSF; Dai, 2008; Goodman, 2005; Mahajan, 2013). To do this we first define the complex-valued generalized pupil function g,

$$g(x',y') = p(x',y')\exp\left[\frac{i2\pi}{\lambda} w(x',y')\right] \quad (2)$$

The real-valued function p is the pupil function that describes transmission through the pupil. It may be defined as 1 within the pupil area and 0 elsewhere, but may also be used to describe variable transmission as a function of position (apodization), such as due to the Stiles-Crawford effect (Atchison & Scott, 2002; Metcalf, 1965; Stiles & Crawford, 1933).

The PSF for incoherent light is then given by the squared modulus of the Fourier transform of the generalized pupil function,

$$h(x,y) = ||F[g(x',y')]||^2. \quad (3)$$

Equipped with the PSF it is then possible to compute the retinal image $r(x,y)$ as the convolution of the source image $s(x,y)$ and the PSF,

$$r(x,y) = h(x,y) * s(x,y). \quad (4)$$

These basic principles relating wavefront aberrations to the PSF are well known (Dai, 2008; Goodman, 2005; Mahajan, 2013) and were first introduced to computation of retinal images by Artal (1990). Methods to scale Zernike coefficients from one pupil size to another were developed by Schwiegerling (2002) and refined by others (Bara, Arines, Ares, & Prado, 2006; Dai, 2006; Díaz, Fernández-Dorado, Pizarro, & Arasa, 2009; Janssen & Dirksen, 2006; Mahajan, 2010). Methods to compute the polychromatic PSF from monochromatic data have also been developed (Artal, Santamaria, & Bescos, 1989; Coe, Bradley, & Thibos, 2014; Marcos, Burns, Moreno-Barriusop, & Navarro, 1999; Ravikumar, Thibos, & Bradley, 2008; Van Meeteren, 1974).

In this report we describe software to compute human optical PSFs from measurements of wavefront aberrations. The software is written in the Mathematica programming language (Wolfram Research, Inc., 2013). The software, in the form of a Mathematica

Citation: Watson, A. B. (2015). Computing human optical point spread functions. *Journal of Vision*, *15*(2):26, 1–25, http://www.journalofvision.org/content/15/2/26, doi:10.1167/15.2.26.

doi: 10.1167/15.2.26 Received October 15, 2014; published February 27, 2015 ISSN 1534-7362

notebook, is provided as a supplement to this report. We also provide in the notebook a database of wavefront aberrations of 200 healthy young eyes (Thibos, Hong, Bradley, & Cheng, 2002). Finally we also provide a demonstration that illustrates effects of parameter selection on computation of the PSF. These supplementary files can be viewed with the free Wolfram CDF player (http://www.wolfram.com/cdf-player/).

An example

To motivate the subsequent developments, we provide a very brief example of the use of this software. We begin with a list of 12 Zernike coefficients,

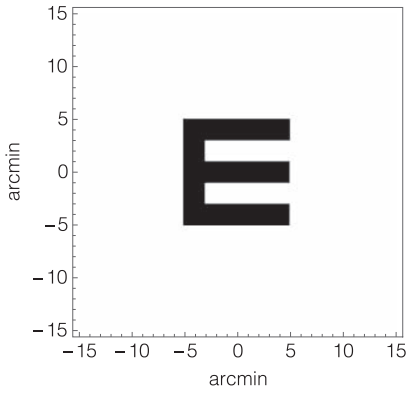
```
zc = {{2,−2,−0.0946},{2,0,0.0969},{2,
2,0.305},{3,−3,0.0459},{3,−1,
−0.121},{3,1,0.0264},{3,3,−0.113},
{4,−4,0.0292},{4,−2,0.03},{4,0,
0.0294},{4,2,0.0163},{4,4,0.064}};
```

We compute a PSF,

```
psf = ZernikePointSpread[zc];
```

We have an image of the letter "E" in the Sloan font with a height of 10 arcmin, corresponding to acuity of 20/40.

```
ImagePlot[letter]
```

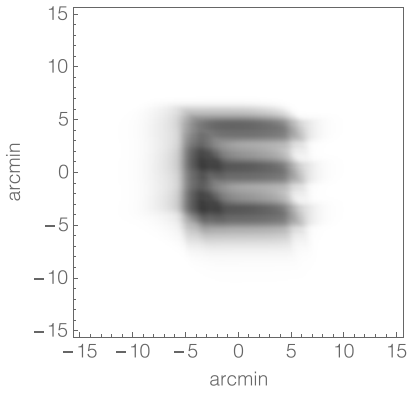


We convolve the PSF with the letter image, and display the result,

```
blurredletter = ImageConvolve[letter,
Wrap[psf]]
```

and display the result

```
ImagePlot[blurredletter]
```



Basics

We do not attempt a thorough introduction to Mathematica here, but instead introduce a few concepts and notations that will be useful in understanding the subsequent examples (Wolfram Research Inc., 2013). We assume the reader is familiar with the essentials of digital images and filtering. A readable introduction to the subject is provided by Bracewell (2003).

Mathematica notebooks, input, and output

Mathematica code usually exists in notebooks, which are digital documents much like a word-processing file. Mathematica input usually consists of typed expressions. When evaluated (by means of Shift-Enter), the output consists of printed numbers, expressions, graphics, or other actions. Here we show input as bold courier text, and printed output as normal courier.

Lists

In Mathematica lists are enclosed by curly braces. Arrays are represented by lists of lists. Arrays use "row major" ordering, meaning each sublist is a row. Here is an example of an array with two rows each of three elements.

```
array = {{0, .25, 1}, {.5, 1, .3}};
```

A value within an array is specified by indexing its row and column.

```
array[[1,2]]
```

```
0.25
```

Pure functions

We make frequent use of a Mathematica programming device known as a pure function. This allows the construction of a temporary function made from existing components. It is signaled by an & at the end of an expression and a dummy variable is denoted by #. In this example

```
(#+2)&[7]
```

```
9
```

the pure function just adds 2 to its argument.

Images

Images are represented here by two-dimensional (2-D) rectangular arrays of numbers, expressed as Mathematica arrays, as described above. We adopt the convention that the first row is the bottom of the image, and that dimensions are given in the order {width,height}. To map from pixel coordinates to degrees of visual angle, an image also has an implicit visual resolution, in horizontal and vertical pixels/degree. The implicit size {width, height} in degrees is the product of dimensions in pixels and the visual resolution. In the introductory example above, {width,height} = {64, 64} pixels and {width, height} = {1/8, 1/8} degrees. The spatial resolution of the image is thus 512 pixels/degree.

Image DFT

The Discrete Fourier Transform (DFT) of an image is also a rectangular array, of the same dimensions as the original image, in which each entry is a complex number that represents the magnitude and phase of a Fourier component of the image. Just as the image has a spatial resolution in samples/degree, the DFT has a spectral resolution in samples/cycle/degree. The spectral resolution is equal to the image size in degrees. The one-dimensional spacing between samples in the DFT is the inverse of the spectral resolution (cycles/degree/sample). Software to compute the DFT is widely available.

Wrapped images

In mapping between arrays and image or DFT coordinates, it is often necessary to represent both positive and negative coordinates. For example, the DFT must represent both positive and negative frequencies, and the PSF must represent both positive and negative coordinates relative to its center. In such cases it is common to shift (rotate or scroll or wrap) each row and column so that the sample corresponding to the origin, or coordinate {0,0}, is the first pixel in the array. For an even dimension, this means rotation to the left by $n/2$ where n is the width or height. For an odd dimension, it means rotating by $(n-1)/2$. This "wrapped" format is also expected as input, and produced as output, by most DFT software, so that the origin is located unambiguously (Voelz, 2011). We provide the **`Wrap`** function to perform this operation.

Image filtering

An optical filter may be represented by the PSF or by the optical transfer function (OTF), which is the DFT of the PSF. Filtering of an image may be achieved by convolution between the image and the PSF, or by multiplication of the image DFT and the OTF, followed by an inverse DFT (Bracewell, 2003). The OTF is generally complex. The absolute value, or modulus, of the OTF is the modulation transfer function (MTF).

Zernike coefficients

We represent the Zernike coefficients for an eye by a list of triples of the form $\{n, m, c\}$ where n is the order, m is the frequency, and c is the coefficient. The coefficients can also be indexed by a single integer mode (Thibos, Applegate, et al., 2002). This is useful when coefficients are to be plotted as a function of mode (see **`ZernikeMode`** and **`ZernikePlot`** below).

Computing Zernike polynomials

Example Zernike coefficients

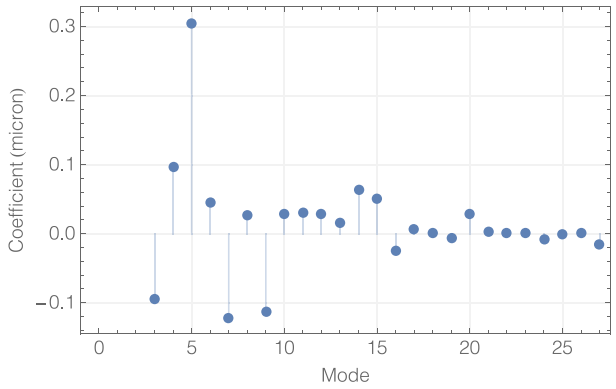
We begin with a set of Zernike coefficients. These are from right eye #1 at 6 mm from the database discussed below (**`ThibosHongBradleyChengData`**; Thibos, Hong et al., 2002), from order 2 to order 6.

`TestCoefficients`

```
{{2,-2,-0.0946},{2,0,0.0969},{2,2,
0.305},{3,-3,0.0459},{3,-1,-0.121},
{3,1,0.0264},{3,3,-0.113},{4,-4,
0.0292},{4,-2,0.03},{4,0,0.0294},
{4,2,0.0163},{4,4,0.064},{5,-5,
0.0499},{5,-3,-0.0252},{5,-1,
0.00744},{5,1,0.00155},{5,3,
-0.00686},{5,5,0.0288},{6,-6,
0.00245},{6,-4,0.00185},{6,-2,
0.00122},{6,0,-0.00755},{6,2,
-0.000693},{6,4,0.000551},{6,6,
-0.0148}}
```

These can be plotted against the single number index mode, with the function `ZernikePlot`.

```
ZernikePlot[TestCoefficients]
```



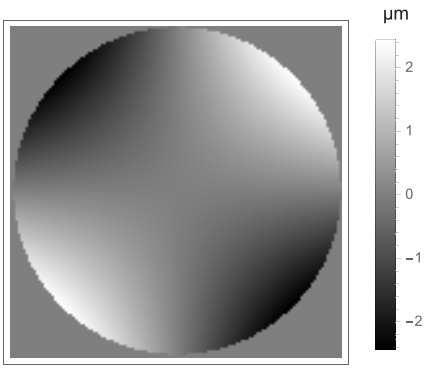
The Zernike image

Each Zernike coefficient describes the order, frequency, and magnitude of a single Zernike polynomial basis function defined over the unit pupil. The sampled image of a single Zernike polynomial can be computed with the function **`ZernikeImage`**.

```
zi = ZernikeImage[2, −2, 63.1];
```

The first two arguments are the order and frequency. The third argument is the radius of the pupil in pixels. This radius may be a real number. The implications of this choice will be discussed later, but in general a larger value will yield greater accuracy. The resulting image size (rows or columns) in pixels is **`2 Ceiling[radius]`**. Ceiling returns the smallest integer greater than or equal to its argument. Because we are assuming circular pupils, the result is always a square image. We display the image.

```
AberrationPlot[zi]
```



We can also display the Zernike image with pseudocolor

```
AberrationPlot[zi, PseudoColor→True]
```

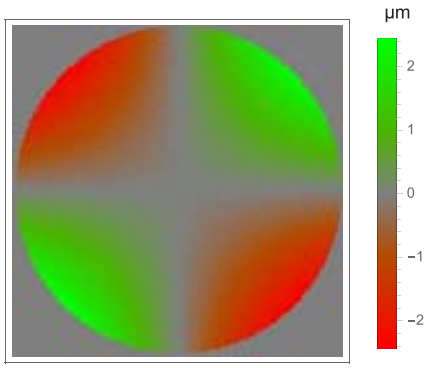
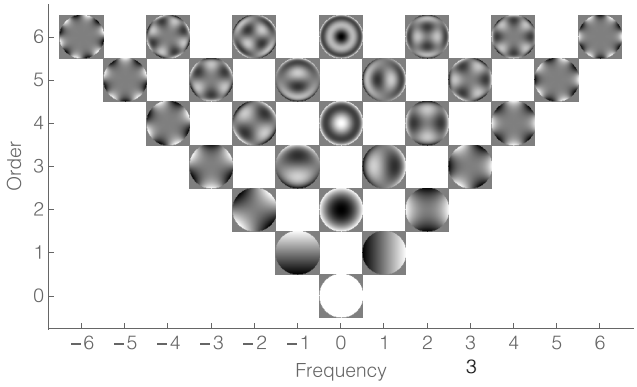


Table of Zernike images

We can also show a table of all Zernike images up to a specified order. The picture shows all the Zernike polynomial basis functions, up to order 6, arranged by order and frequency.

```
ZernikeTable[6]
```



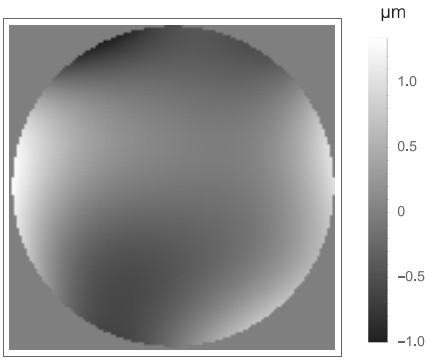
The Wavefront image

The total wavefront aberration image is the sum of Zernike polynomial images, weighted by their coefficients. This can be computed with **`WaveAberrationImage`**. We supply a list of coefficients and a pupil radius in samples.

```
wai = WaveAberrationImage
[TestCoefficients, 63.1];
```

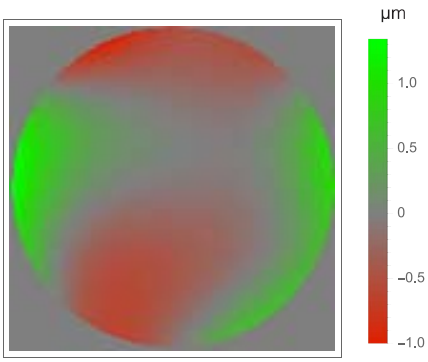
The image can be displayed as a grayscale image,

```
AberrationPlot[wai]
```



or by a pseudocolor image.

```
AberrationPlot[wai,
PseudoColor→True]
```



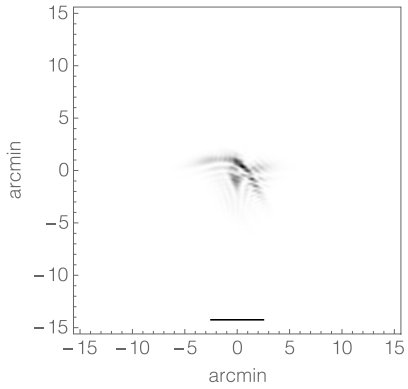
Computing the PSF and OTF

The PSF for a set of coefficients is created with the function **ZernikePointSpread**. We create a PSF for the example set of aberrations.

```
psf = ZernikePointSpread
[TestCoefficients];
```

The returned PSF is a 2-D array of real numbers in wrapped form. We plot the PSF using the function **PSFPlot**. This shows the dimensions of the PSF in arcmin. The small fiducial line at the bottom is 5 arcmin in length. This is the width of a test letter for an acuity of 20/20. We also note that by default **PSFPlot** stretches and reverses the contrast of the PSF: Largest values are plotted as black, the smallest values (usually zero) as white. This permits much easier visualization of the PSF.

```
PSFPlot[psf]
```



The optical PSF is always positive, and its integral is always 1, since it describes the redistribution of light.

```
Total[psf, 2]
```

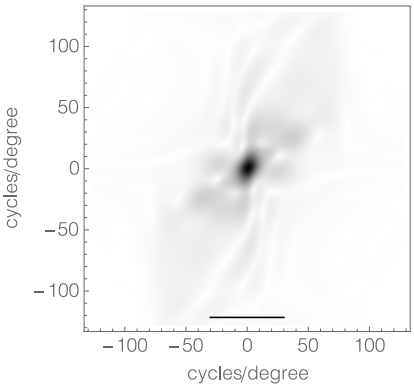
```
1.
```

By default the function **ZernikePointSpread** returns a PSF. If we specify the option **OTF→True**, it will instead return the OTF. If **OTF→"Both"**, it will return a list {PSF, OTF}. Here we illustrate computing both the PSF and the OTF.

```
{psf,otf} = ZernikePointSpread
[TestCoefficients, OTF→"Both"]
```

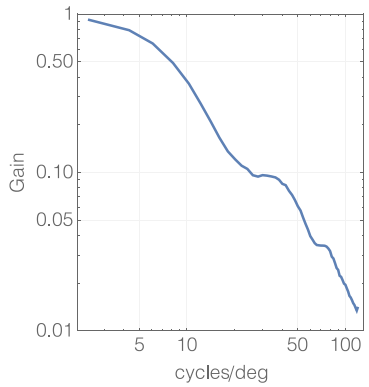
In general, the OTF is complex. The MTF is the absolute value of the OTF. We show the MTF, using the function **MTFPlot**. We magnify by 2 to see more detail. The black bar at the bottom of the plot indicates ±30 cycles/degree.

```
MTFPlot[otf, Magnification→2]
```



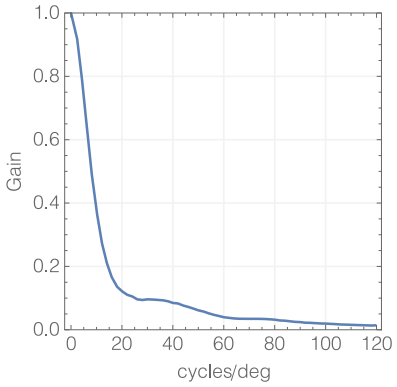
We can also view the radial MTF. The function **RadialMTF** provides a log-log plot of the MTF as a function of radial frequency, averaged over polar angle.

RadialMTF[otf]



The radial plot can also be produced with linear axes.

RadialMTF[otf, LinearPlot→True]



ZernikePointSpread options

The function `ZernikePointSpread` has a number of options.

Options[ZernikePointSpread]

```
{Verbose→False, Wavelength→555,
PupilDiameter→6, OTF→False,
PupilSamples→Automatic,
ImageSamples→256, Degrees→0.5,
Apodization→False,
DerivedParameters
→{PupilSamples→62.8947}}
```

In the previous example the options were assigned their default values. Here we provide some discussion of the most significant options.

Wavelength

This is the monochromatic wavelength in nm for which the coefficients are defined. Typically, this is the wavelength used to measure the aberrations. Below we will show how aberrations at other wavelengths can be computed.

PupilDiameter

This is the pupil diameter in mm for which the coefficients are defined. Usually this is the pupil diameter at which the coefficients were measured. Below we will show how aberrations may be scaled for smaller pupils.

PupilSamples

This is the diameter of the pupil in samples. It contributes to the accuracy of the computed PSF. This will be discussed further below.

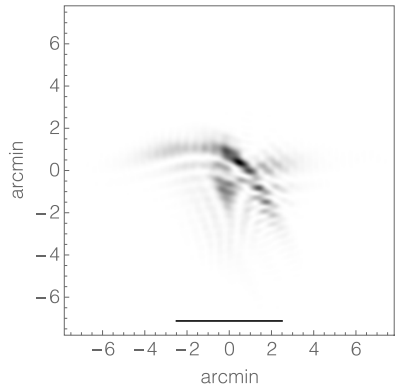
ImageSamples

This is the size of the PSF image in samples. Because the image is square, this is both the number of rows and columns. It contributes to the accuracy of the computed PSF. This will be discussed further below.

Degrees

This is the size of the PSF image in degrees. Because the image is square, this is both the width and height. It contributes to the accuracy of the computed PSF. This will be discussed further below. Here is an example in which the size is 0.25°. Because we are no longer using the default of 0.5°, we also must specify the image size to **PSFPlot**.

PSFPlot[ZernikePointSpread[
TestCoefficients, Degrees→.25],
Degrees→.25]



DerivedParameters

The user will typically specify the PSF size in degrees, in which case **PupilSamples** will be determined automatically, and the computed value will be expressed here. The user may however select **PupilSamples** directly, in which case the **Degrees** option will be computed automatically and expressed here.

Apodization

Apodization refers to nonuniform transmission through the pupil. In **ZernikePointSpread**, by default **Apodization→False**. Alternatively a Gaussian standard deviation in millimeters can be supplied. Apodization is discussed more extensively below.

Verbose

When **Verbose→True, ZernikePointSpread** provides a set of diagnostic images and statistics.

Computing a retinal image

Our purpose in computing the PSF is usually to obtain the retinal image of some object. Here we illustrate the basic steps. First we create an image of a single letter, explicitly specifying the image dimensions in pixels and in degrees. We specify the letter size to be 10 arcmin, the size of the 20/40 line on an eye chart.

```
fontsize = 10/60;
imagesize = 256;
degrees = .5;
fontsizepixels = fontsize imagesize/
degrees;
letter = ImageReflect[LetterImage["Z",
ImageSize→imagesize,
  FontFamily→"Sloan,"
  FontSize→fontsizepixels]];
```

We create a psf.

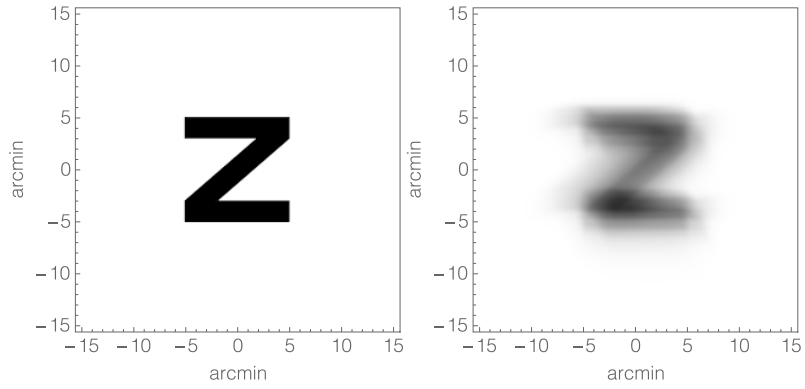
```
psf = ZernikePointSpread
[TestCoefficients];
```

We convolve the letter image and the PSF. The **Wrap** function shifts the origin of the PSF to the center of the array, as expected by the Mathematica function **ImageConvolve** (see Wrapped images).

```
blurredletter = ImageConvolve[letter,
Wrap[psf]];
```

We display the original and blurred letters.

```
Row[ImagePlot /@ {letter,
blurredletter}]
```



Note that because the PSF merely redistributes light, the integral of input and output images must be (approximately) the same.

```
Total[ImageData[#],2]& /@ {letter,
blurredletter}
```

```
{ 61167.9, 61174.1}
```

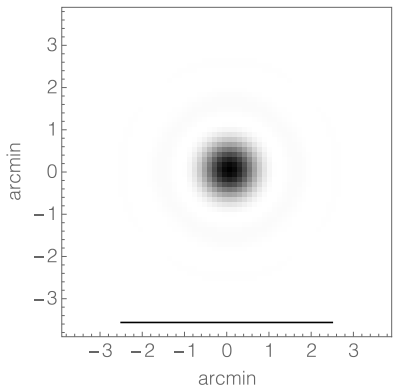
Computing the diffraction-limited PSF and MTF

As a useful test and an instructional example we can compute the PSF and OTF for the case in which there are no aberrations, and the optics is limited only by diffraction. We supply an empty list of Zernike coefficients, and set the pupil diameter to 2 to increase the prominence of diffraction. This will compute the monochromatic result at the default wavelength of 555 nm.

```
{psf, otf} = ZernikePointSpread[{},
PupilDiameter→2, OTF→"Both"];
```

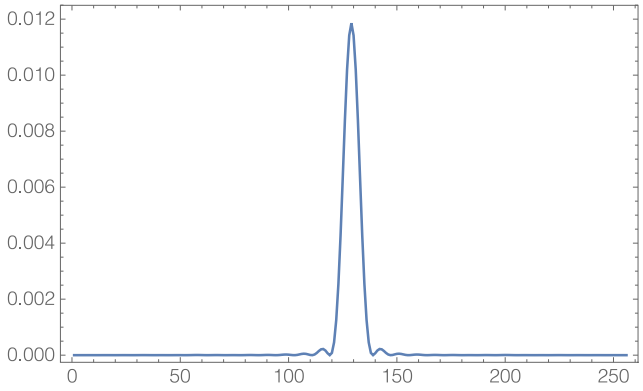
We plot the PSF. We magnify by 4 to see more detail.

```
PSFPlot[psf, Magnification→4]
```



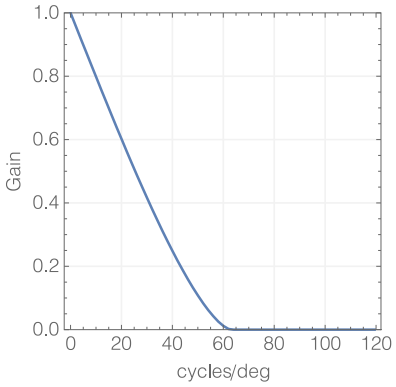
We can also look at a cross section through the peak.

```
ListLinePlot[Wrap[psf[[1]]]]
```



We can also view the radial MTF with linear axes.

```
RadialMTF[otf, LinearPlot→True]
```



It is well known (Goodman, 2005) that the cut-off frequency (the so-called diffraction limit) for a given pupil diameter and wavelength is given in cycles/degree by the formula

$$\frac{10^6 \pi\, pupil}{180\lambda} \tag{5}$$

which in this case equals 62.9 cycles/degree, which agrees with the figure.

Chromatic aberration

To compute a polychromatic PSF it is useful to have a function that describes the defocus induced by longitudinal chromatic aberration. We have implemented a published formula (Thibos, Ye, Zhang, & Bradley, 1992) that describes defocus in diopters at a wavelength λ for an eye in focus at 589 nm,

$$D_{589}(\lambda) = 1.68524 - \frac{0.63346}{\lambda 10^{-3} - 0.2141}. \tag{6}$$

This is implemented by ChromaticDefocus, as in the following example.

```
ChromaticDefocus[500]
```

```
-0.53043
```

For an eye in focus at λ_0, the defocus at other wavelengths λ will be

$$D(\lambda, \lambda_0) = D_{589}(\lambda) - D_{589}(\lambda_0) \tag{7}$$

This is implemented by **ChromaticDefocus**, with a second argument that is the wavelength in focus, as in the following example,

```
ChromaticDefocus[500,555]
```

```
−0.357471
```

This can be converted to a Zernike defocus coefficient in micrometers by the formula

$$c_2^0 = \frac{D(\lambda, \lambda_0)p^2}{16\sqrt{3}}. \quad (8)$$

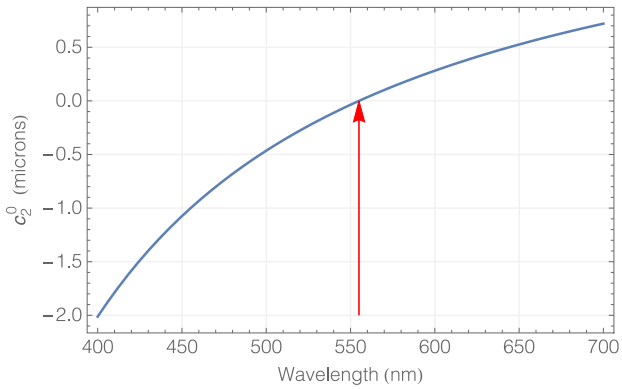
where p is the pupil diameter in mm. This is implemented by **ChromaticDefocusZernike**, as in the following example

```
ChromaticDefocusZernike[500,555]
```

```
−0.464368
```

A plot of this function is shown below, with an arrow marking the wavelength in focus (555 nm).

```
Plot[ChromaticDefocusZernike
[wavelength, 555],
{wavelength, 400, 700}
  , Epilog→{Red, Arrow[{{555, −2},
  {555, 0}}]}
  , FrameLabel→{"Wavelength (nm)",
  "c20(microns)"}
  , GridLines→Automatic,
  Evaluate@ZernikeStyles]
```



Computing a polychromatic PSF

Wavefront aberrations are typically recorded for a single wavelength, which allow computation of a monochromatic PSF, and thereby the retinal image of a monochromatic object. To compute the retinal image of a polychromatic object, we need to compute a polychromatic PSF (Artal et al., 1989; Marcos et al., 1999; Nestares, Navarro, & Antona, 2003; Ravikumar et al., 2008; Van Meeteren, 1974). The general approach is to consider a polychromatic source image as the sum of a set of monochromatic images, and to filter each by a PSF for the appropriate wavelength. In one special case, the source image is spectrally homogeneous, meaning that the image varies only in intensity, not in its spectral composition. Examples are an achromatic object illuminated by a single source, or a grayscale image. In that case a polychromatic PSF can be constructed as a sum of monochromatic PSFs, and the single intensity image filtered accordingly.

To compute polychromatic PSFs we must be able to compute a PSF for an arbitrary wavelength, given set of wavefront aberrations recorded at one particular wavelength. To do this, we first assume that only the defocus Zernike coefficient varies with wavelength, due to longitudinal chromatic aberration (Marcos et al., 1999). Then we compute the relevant defocus, assuming a particular in-focus wavelength, and using the function **ZernikeChromaticDefocus**. In the following sections we illustrate the use of **ZernikePointSpread** to compute polychromatic PSFs under various scenarios.

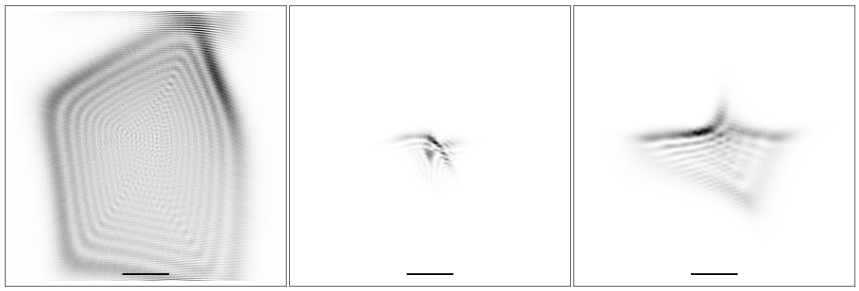
Multispectral PSFs

In the previous examples, we created the PSF for monochromatic light at 555 nm. If the observer were viewing a multispectral image, consisting of the sum of several monochromatic subimages (bands), then we can compute the appropriate set of PSFs to filter each band separately. In this example, we assume three bands at 400, 555, and 700 nm, and assume that the eye is in focus at 555. The second argument is a list of the wavelengths. Note that we have not altered the default option **Wavelength→555**, so that remains the wavelength in focus.

```
psfs = ZernikePointSpread
[TestCoefficients, {400, 555, 700}] ;
```

We plot the PSFs. Note the large defocus at 400 nm, caused by chromatic aberration. These three PSFs could now be used to convolve the three monochromatic bands, and the sum would represent the resulting retinal image.

```
Row[PSFPlot[#,ImageSize→160,
FrameLabel→None,FrameTicks→None]& /
@ psfs]
```



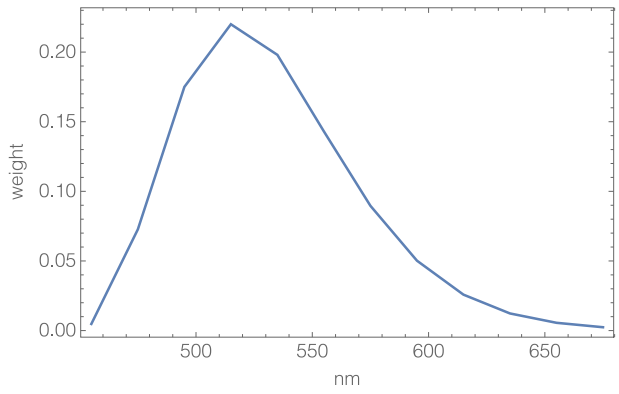
Polychromatic PSF

If the eye is viewing an image that is spectrally homogeneous (it varies only in intensity, not spectrally), then we can compute an appropriate single PSF for the relevant spectrum. We do this by representing the spectrum by a set of sampled wavelengths and corresponding weights. We then compute a set of PSFs, one for each wavelength, with chromatic aberration, as in the example above. Then we multiply by the corresponding weights and sum them up.
Here is an example spectrum. The weights add up to one.

```
spectrum = {{455,0.00477},{475,
0.0727},{495,0.175},{515,0.22},{535,
0.198},{555,0.143},{575,0.0896},
{595,0.0502},{615,0.0258},{635,
0.0123},{655,0.00558},{675,0.0024}}
```

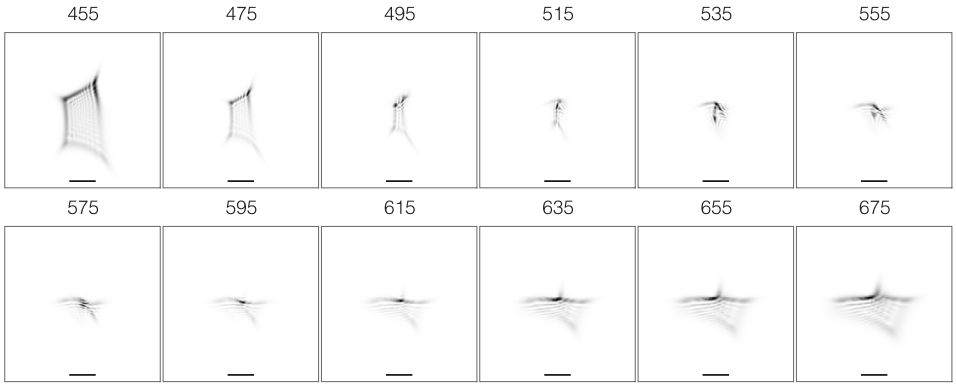
We plot it.

```
ListLinePlot[spectrum,ZernikeStyles]
```



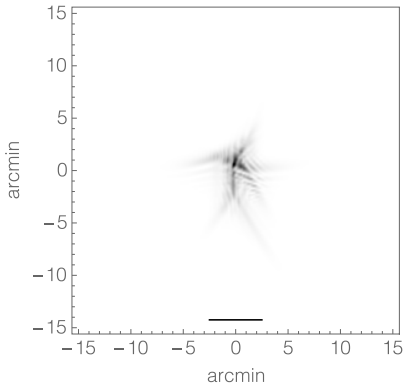
First we will create the individual PSFs separately, for illustration, and plot them labeled by their wavelengths.

```
wavelengths = First /@ spectrum;
psfs = ZernikePointSpread
[TestCoefficients,wavelengths] ;
figs = PSFPlot[#,ImageSize→100,
FrameLabel→None,FrameTicks→None]& /
@ psfs;
Grid[Partition[MapThread[
  Labeled[##1,Top, LabelStyle→{12,
  FontFamily→"Helvetica"}]&
  ,{figs,wavelengths}],6],
  Spacings→0]
```



Combining these with appropriate weights, we get the single PSF.

```
weights = Last /@ spectrum;
psf = Total[psfs weights];
PSFPlot[psf]
```



Here is how to create the polychromatic PSF in a single call.

```
psf = ZernikePointSpread
[TestCoefficients,spectrum];
```

Polychromatic white light PSF

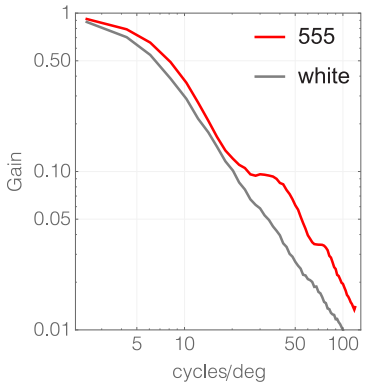
In some cases the source image may be that of an achromatic object illuminated by white light, or a grayscale image. For this special case we employ as a spectrum the CIE photopic luminosity function, sampled with some spacing, and centered on the wavelength in focus (555 nm by default). The user specifies the spacing in nm. For comparison, we will compute both the monochromatic and the white light version. In the second case, the second argument is the spacing (nm) of wavelength samples around the in-focus wavelength.

```
otf1 = ZernikePointSpread
[TestCoefficients, OTF→True] ;

otf2 = ZernikePointSpread
[TestCoefficients, 20, OTF→True] ;
```

We compute the radial mean MTF for each, and plot them. We see that inclusion of the other wavelengths, and their chromatic aberration, significantly reduces gain at the higher spatial frequencies.

```
RadialMTF[{otf1,otf2},
PlotStyle→{Red,Gray},
  PlotLegends→Placed[{"555",
  "white"},{Right,Top}]]
```



Trichromatic PSFs

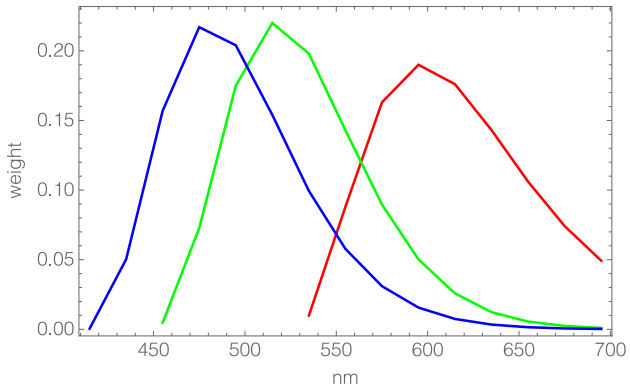
In modern electronic displays, color images are produced by combining three (or more) images, each created with a specific primary. The individual images are spectrally homogeneous, but their individual spectra may be broad band. In this case, the spectrum argument to ZernikePointSpread is a list of spectra, and the result is a list of PSFs (or OTFs, or {PSF, OTF} pairs, depending on the OTF option).

Here is an example of spectra for three bands, nominally red, green, and blue. These are approximations to spectra of organic light-emitting diode (OLED) display primaries. Each has weights that sum to 1.

```
spectra = {{535,0.00986},{555,0.088},
{575,0.163},{595,0.19},{615,0.176},
{635,0.143},{655,0.106},{675,
0.0741},{695,0.0493}},{{455,
0.00477},{475,0.0727},{495,0.175},
{515,0.22},{535,0.198},{555,0.143},
{575,0.0896},{595,0.0502},{615,
0.0258},{635,0.0123},{655,0.00558},
{675,0.0024},{695,0.000991}},{{415,
0.000458},{435,0.0503},{455,0.157},
{475,0.217},{495,0.204},{515,0.154},
{535,0.0994},{555,0.0578},{575,
0.031},{595,0.0156},{615,0.00744},
{635,0.0034},{655,0.0015},{675,
0.000641},{695,0.000266}}};
```

We plot them.

```
ListLinePlot[spectra,
PlotStyle→{Red,Green,Blue},
  FrameLabel→{"nm","weight"},
  ZernikeStyles]
```



We compute both PSF and OTF for the three primaries.

```
tri = ZernikePointSpread
[TestCoefficients,spectra,
OTF→"Both"];
```

The result has the following dimensions.

```
Dimensions[tri]
```

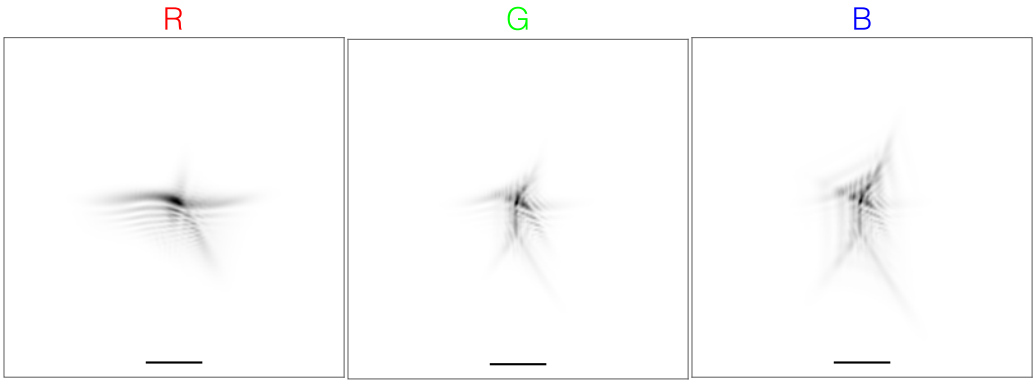
```
{3,2,256,256}
```

For clarity, we rearrange and name the parts.

```
{psfs,otfs} = Transpose[tri];
```

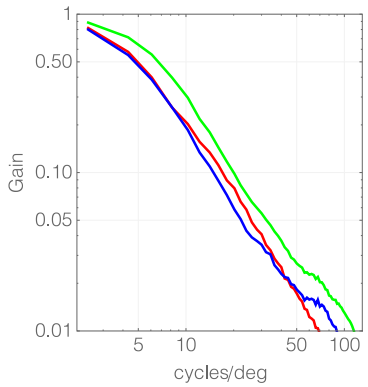
We look at the PSFs.

```
Row[MapThread[
  PSFPlot[#1, ImageSize→160,
  FrameLabel→None, FrameTicks→None,
    PlotLabel→Style[#2, #3]]& ,
      {tri[[All,1]], {"R," "G," "B"},
      {Red, Green, Blue}}]]
```



We compare the radial MTFs. The green primary has the best gain, consistent with the most compact PSF.

```
RadialMTF[otfs, PlotStyle→{Red,
Green, Blue}]
```



To illustrate the use of the trichromatic PSF to compute the trichromatic retinal image, we again create a letter image. The letter is again 10 arcmin in size, but this time the image is only 0.25°.

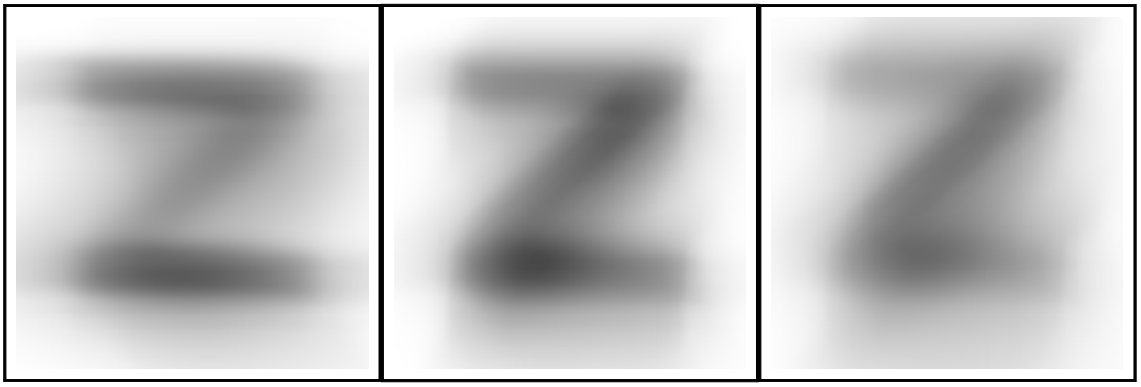
```
fontsize = 10/60;
imagesize = 128;
degrees = .25;
fontsizepixels = fontsize imagesize/
degrees;
letter = LetterImage["Z",
ImageSize→imagesize,
FontFamily→"Sloan",
  FontSize→fontsizepixels]
```

We blur this image by each of the three trichromatic PSFs, creating three blurred images that represent the red, green, and blue color channels

```
blurredcolorletter =
ImageConvolve[letter,Wrap[#]]& /@
psfs;
```

We look at the individual blurred color channels.

```
Row[Framed /@ blurredcolorletter]
```



We simulate the blurred color image by combining the three blurred color channels as an RGB image.

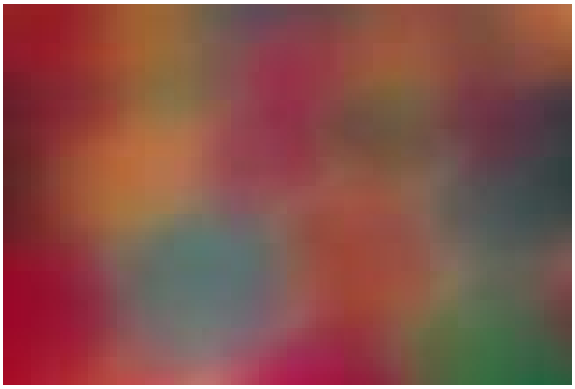
```
Framed @
ColorCombine[blurredcolorletter ,
"RGB"]
```

Here we apply the PSF to a more complicated and colorful image. This image is 160 × 107 pixels, or 0.312° × 0.209° (512 pixels/°).

```
image
```

We use the Mathematica functions **ColorSeparate** and **ColorCombine** to separate the three color channels, and to recombine them after filtering.

```
ColorCombine[MapThread
[ImageConvolve,
  {ColorSeparate[colorimage],
  Reverse[Wrap[#]]& /@ psfs}]]
```



Database of 200 eyes

Wavefront aberration data for 200 eyes from 100 young healthy observers measured at 633 nm and several pupil diameters (Thibos, Hong et al., 2002) are available in the object **ThibosHongBradley ChengData**. The data structure consists of a list indexed by pupil diameter, observer, eye, and mode. The first index corresponds to pupil diameters of 3, 4.5, 6, and 7.5 mm. The third index is in the order left eye, right eye. The number of observers varies with pupil diameter: 70 for 7.5 mm and 100 for the rest. The number of modes varies with pupil diameter from 15 to 66. Each coefficient is represented in our standard format as a triple $\{n,m,c\}$ where n is the order, m is the frequency, and c is the coefficient. Additional documentation is available in the supplementary notebook Zernike.nb.

We can select the data for the right eyes of the first 24 observers, at a 6-mm pupil, as follows. In the result each record consists of 36 Zernike coefficients, up to order 7.

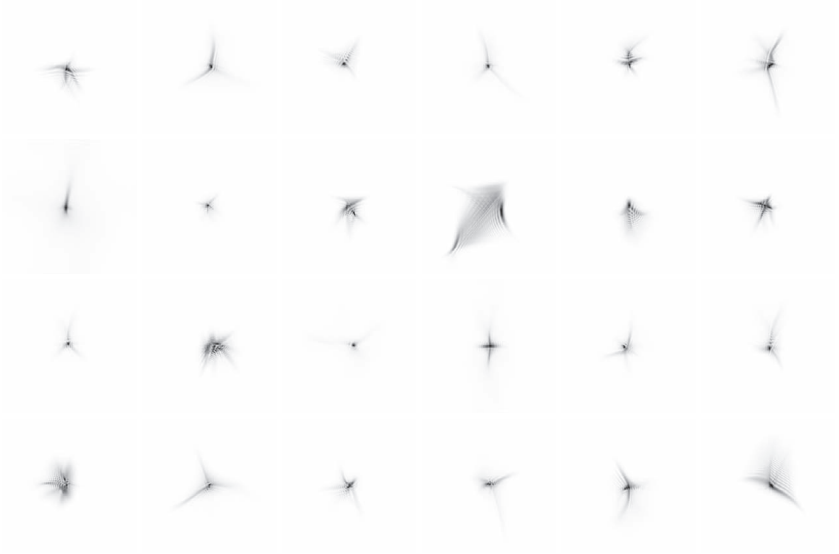
```
eyes = ThibosHongBradleyChengData[[3,
;; 24, 2]];
```

We compute the white light PSF and OTF for each eye.

```
{psfs,otfs} =
Transpose[ZernikePointSpread[#, 20,
OTF → "Both"]& /@ eyes];
```

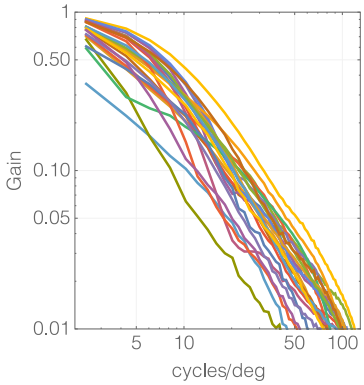
We display the PSFs.

```
Grid[Partition[PSFPlot[#,
ScaleMark→False, Frame→False]& /@
psfs,6]]
```



We can also show the radial MTFs for these 24 eyes, which illustrates the variability among a group of young, healthy, well-corrected eyes.

```
RadialMTF[otfs]
```



This database is very useful for simulating the statistical distribution of performance in a population of observers (Watson, 2013; Watson & Ahumada, 2012).

Scaling Zernike coefficients to a different pupil size

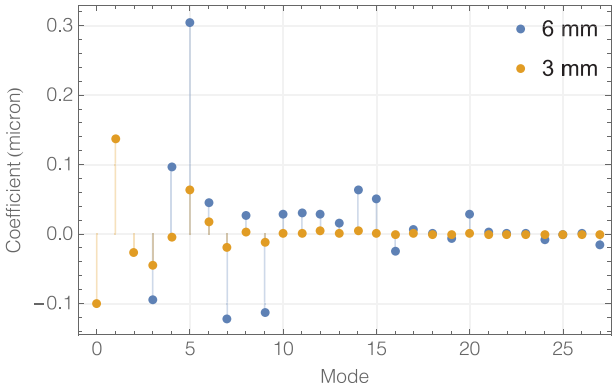
Zernike coefficients are measured over a pupil of a specific diameter. Measurements at several pupil diameters for the same eye may be available. It is also possible to numerically transform Zernike coefficients from one pupil diameter to another (usually smaller) diameter (Bara et al., 2006; Dai, 2006; Díaz et al., 2009; Janssen & Dirksen, 2006; Mahajan, 2010; Schwiegerling, 2002). We make use of the formula of Dai (2006) for the new coefficients b based on the old coefficients a,

$$b_n^m = r^n \left[a_n^m + \sum_{i=1}^{(o-n)/2} \left(a_{n+2i}^m \sqrt{(n+2i+1)(n+1)} \sum_{j=0}^{i} r^{2j} \frac{(-1)^{i+j}(n+i+j)!}{(n+j+1)!(i-j)!j!} \right) \right] \quad (9)$$

where r is the ratio of new to old pupil diameters, o is the highest order considered. The formulas of Diaz et al. (2009) and Janssen et al. (2006) are equally valid; we use Dai (2006) because it is a relatively simple expression and it appears to be the fastest to compute in our environment. We note that this formula can be used to scale for smaller or larger pupils (ratios less than or greater than 1), though the error is larger for ratios greater than 1 (Bará, Pailos, Arines, López-Gil, & Thibos, 2014; Dai, 2011; Ommani, Hutchings, Thapa, & Lakshminarayanan, 2014).

We provide the function **`ZernikeScalePupil`** to scale coefficients for smaller or larger pupils. In this example, we scale the **`TestCoefficients`** from their original 6-mm pupil diameter to a new 3-mm diameter. The second argument is the ratio of new to old pupil diameters. We plot the old and new coefficients.

```
zc = ZernikeScalePupil
[TestCoefficients,3/6];
ZernikePlot[{TestCoefficients,zc},
  PlotLegends→Placed[{"6 mm","3 mm"},
  {Right,Top}]]
```

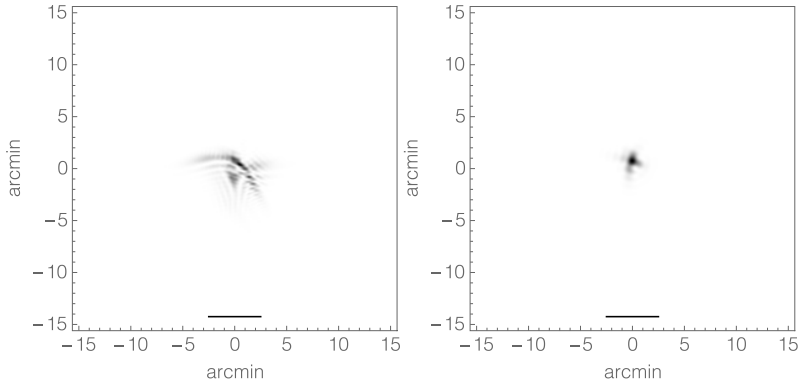


We create the corresponding PSFs and OTFs.

```
{psfs, otfs} = Transpose[MapThread[
  ZernikePointSpread[#1,
  PupilDiameter→#2, OTF→"Both"] &,
  {{TestCoefficients, zc}, {6, 3}}]];
```

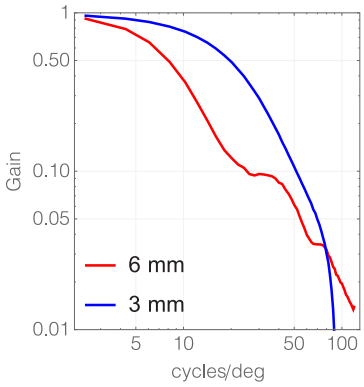
The 3-mm PSF is more compact and shows fewer obvious aberrations.

```
Row[PSFPlot /@ psfs]
```



Likewise the radial MTF shows higher gain over most frequencies for the 3-mm pupil diameter.

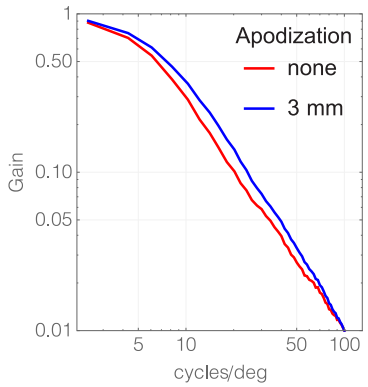
```
RadialMTF
[otfs, PlotStyle→{Red,Blue},
  PlotLegends→Placed[{"6 mm","3 mm"},
  {Left,Bottom}]]
```



Apodization

Apodization refers to nonuniform transmission through the pupil. A notable example is the Stiles-Crawford effect (Stiles & Crawford, 1933). Although this is an effect occurring at the level of the retina, it may be regarded as effectively due to apodization at the pupil (Metcalf, 1965). It has been modeled as a Gaussian, with a standard deviation estimated at around 3 mm under certain conditions (Atchison & Scott, 2002). In **ZernikePointSpread**, by default `Apodization→False`. Alternatively a Gaussian standard deviation in millimeters can be supplied. In this example we show the radial MTF for a 6-mm pupil in white light with and without Gaussian apodization with a standard deviation of 3 mm.

```
otfs = ZernikePointSpread
[TestCoefficients, 20,
  Apodization→#, OTF→True] & /@
  {False, 3.};
RadialMTF
[otfs, PlotStyle → {Red,Blue},
  PlotLegends→Placed
  [{"none","3 mm"},{Right,Top}]]
```



Varying defocus and astigmatism

Defocus and astigmatism are of interest in part because they are aberrations that can be corrected with spectacle lenses. We can easily compute PSFs and OTFs with varying amounts of defocus or astigmatism. For defocus we use the function **InverseEquivalentDefocus** to compute the magnitude of the Zernike coefficient corresponding to defocus. For example, for a 6-mm pupil and 1 diopter of defocus,

```
c = InverseEquivalentDefocus[1., 6]
```

```
1.29904
```

To update a list of coefficients we simply append the defocus term, with the appropriate values of order, frequency, and magnitude: {2,0,1.29904}. Note that it does not matter whether a defocus term already exists in the list (as it does here); the two terms will be correctly combined by **ZernikePointSpread**.

```
Append[TestCoefficients, {2, 2, c}]
```

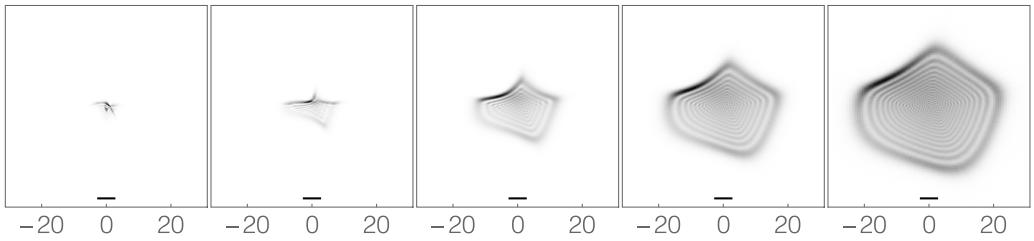
```
{{2,−2,−0.0946},{2,0,0.0969},{2,2,
0.305},{3,−3,0.0459},{3,−1,−0.121},
{3,1,0.0264},{3,3,−0.113},{4,−4,
0.0292},{4,−2,0.03},{4,0,0.0294},
{4,2,0.0163},{4,4,0.064},{5,−5,
0.0499},{5,−3,−0.0252},{5,−1,
0.00744},{5,1,0.00155},{5,3,
−0.00686},{5,5,0.0288},{6,−6,
0.00245},{6,−4,0.00185},{6,−2,
0.00122},{6,0,−0.00755},{6,2,
−0.000693},{6,4,0.000551},{6,6,
−0.0148},{2,2,1.3}}
```

To create the corresponding PSF we would simply supply this new list to **ZernikePointSpread**. Here we illustrate creation of a family of PSF and OTF, with defocus proceeding from 0 to 2 diopters in steps of 1/2 diopter. We set **Degrees → 1** to allow room for the large PSF at the largest defocus.

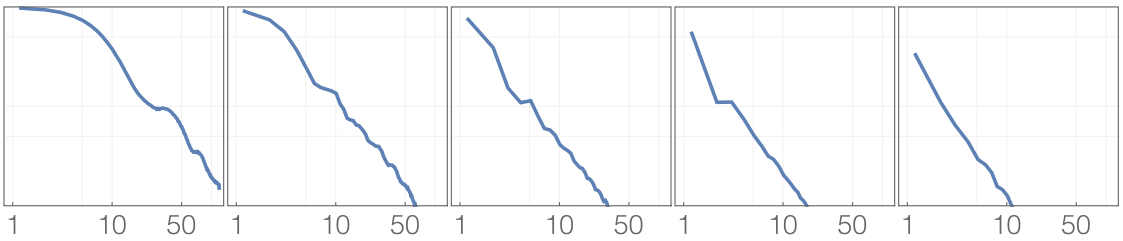
```
result = ZernikePointSpread[
  Append[TestCoefficients, {2, 0,
  InverseEquivalentDefocus[#, 6]}],
  Degrees→1., OTF→"Both"] & /@
  Range[0, 2, .5];
```

Now we plot the PSFs and radial MTFs.

```
Row[PSFPlot[#, Degrees→1,
ImageSize→100,
  FrameTicks → {{−20, 0, 20}, None,
  None, None},
  FrameLabel→None] & /@ result[[All,
  1]]]
```




```
Row[RadialMTF[#, Degrees→1,
ImageSize→100,
  FrameTicks→{{1, 10, 50}, None, None,
  None},
  FrameLabel→None] & /@ result[[All,
  2]]]
```

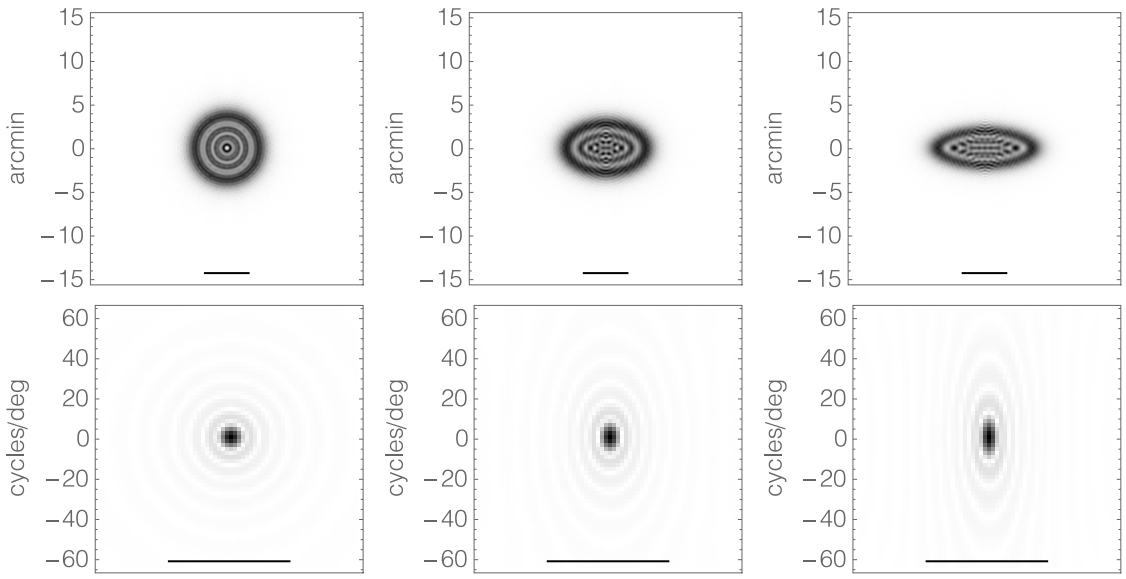


Turning now to astigmatism, we use a more general function that allows the user to specify the defocus (in diopters), the astigmatism (in diopters), and the angle of the astigmatism (in radians). For example, 1 diopter of defocus and 0.2 diopters of astigmatism at an angle of 0.3 at a pupil diameter of 6 mm, would yield

```
SpheroCylindricalCoefficients[1.,
0.2, 0.3, 6]
  {{2,−2,0.207},{2,0,1.3},{2,2,
  0.303}}
```

The function returns a list of second order Zernike coefficients that can then be supplied to **ZernikePointSpread**. In the following example, we show a sequence of PSFs and OTFs with constant defocus (0.5 diopters) and increasing astigmatism at angle 0. Astigmatism results in blur in one direction defined by the angle, so the PSF gets broader in that dimension, while the MTF gets narrower.

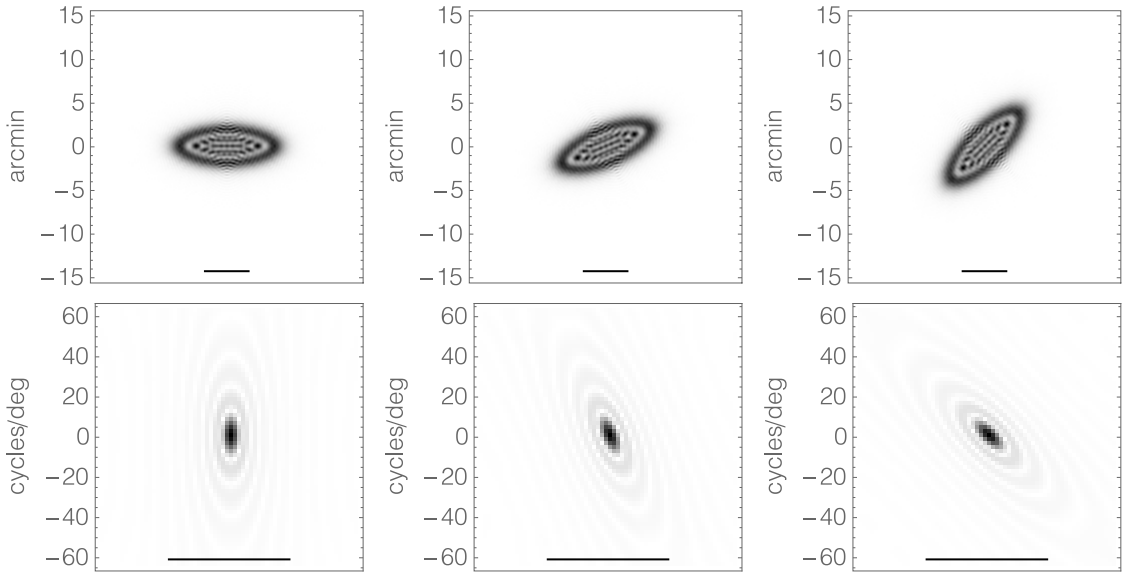
```
defocus = 0.5; pupil = 6; angle = 0.;
options = Sequence @@ {ImageSize→180,
FrameTicks→{None, Automatic}};
Grid[Transpose[({psf, otf} =
ZernikePointSpread[
  SpheroCylindricalCoefficients
  [defocus, #, angle, pupil],
  PupilDiameter→pupil, OTF→"Both"];
  {PSFPlot[psf, FrameLabel→{None,
  "arcmin"}, options],
  MTFPlot[otf, FrameLabel→{None,
  "cycles/deg"},
  Magnification→4, options]}
  ) & /@ {0, .1, .2}]]
```



In the following example we fix the defocus, and vary the angle of astigmatism. Note that the MTF is always elongated orthogonal to the PSF.

```
defocus = 0.5; pupil = 6; astigmatism =
0.2;
options =
  Sequence @@ {ImageSize→180,
  FrameTicks→{None, Automatic}};
```

```
Grid[Transpose[({psf, otf} =
ZernikePointSpread[
    SpheroCylindricalCoefficients
    [defocus, astigmatism, #, pupil],
    PupilDiameter→pupil,
    OTF→"Both"];
    {PSFPlot[psf, FrameLabel→{None,
    "arcmin"}, options],
    MTFPlot[otf, FrameLabel→{None,
    "cycles/deg"},
    Magnification→4, options]}
    ) & /@ {0, Pi/8, Pi/4}]]
```



Some useful functions

WavefrontRMS

This computes the RMS error over the wavefront represented by a set of coefficients, which is equivalent to the RMS of the Zernike coefficients.

```
WavefrontRMS[TestCoefficients]
```

```
0.391534
```

EquivalentDefocus

This function computes the so-called equivalent defocus (in diopters) of a single Zernike coefficient or a set of coefficients. It effectively computes the defocus term that would produce the same RMS error as the set of coefficients. The second argument is the pupil diameter.

```
EquivalentDefocus
[TestCoefficients,6]
```

```
0.301403
```

InverseEquivalentDefocus

Given a defocus in diopters, this returns the value of the Zernike coefficient corresponding to defocus (order 2, frequency 0). The second argument is the pupil diameter.

```
InverseEquivalentDefocus[1., 6]
```

```
1.29904
```

```
EquivalentDefocus
[InverseEquivalentDefocus[1., 6], 6]
```

```
1.
```

SpheroCylindricalCoefficients

Given defocus, astigmatism, angle, and pupil diameter, this returns the list of corresponding Zernike coefficients.

```
SpheroCylindricalCoefficients[1.,
0.2, 0.3, 6]
```

```
{{2, −2, 0.207}, {2, 0, 1.3}, {2, 2,
0.303}}
```

ZernikeMode

Zernike coefficients can also be indexed by a single integer: the mode (Thibos et al., 2002a). Given an order and a frequency, this returns the mode of a Zernike polynomial.

```
ZernikeMode[3, −3]
```

```
6
```

ZernikeIndices

Given a single index Zernike mode, this returns the order and frequency of a Zernike polynomial.

```
ZernikeIndices[6]
```

```
{3, −3}
```

Technical issues in calculation of the PSF from Zernike coefficients

Computation of the PSF image from Zernike coefficients relies upon discrete sampled representation of continuous functions, and is therefore subject to sampling and aliasing errors. In general, to expedite computation, we would like the PSF image to have as few samples (pixels) as possible. But the considerations below provide constraints on how small that number can be.

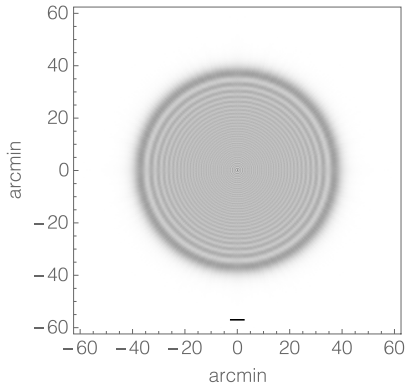
While software implementations may differ, they generally will include four important parameters: p – the pupil diameter in mm, λ – the wavelength in nm, d – the PSF image size in degrees, and n – the PSF image size in pixels.

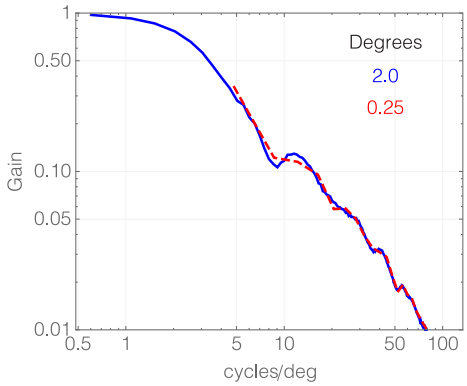
Size of PSF

The first consideration is that d must be large enough to accommodate the PSF. For modest wavefront aberrations, a value of 1/4 degree may be adequate. For large aberrations, a larger PSF image will be required. Here is an example for −4 diopters of defocus with a 6-mm pupil:

```
c20=InverseEquivalentDefocus[−4., 6];

PSFPlot[ZernikePointSpread[{{2,0,
c20}}, Degrees→2], Degrees→2]
```





Resolution of the PSF

The resolution of the PSF in pixels/degree is given by n/d. That value should typically be large enough to accommodate the spectral composition of the PSF. Since the well-corrected OTF may extend up to 120 cycles/degree, it is advisable to set n/d to at least twice this value. In **ZernikePointSpread**, the default values of d (Degrees) and n (ImageSamples) are 0.5 and 256, yielding 512 pixels/degree.

Resolution of OTF

A second consideration is that the spacing of samples in the OTF is $1/d$. Likewise the lowest frequency represented (apart from zero) is $1/d$. Thus if an accurate and fine-grained representation of the OTF or MTF is desired, or if one is interested in behavior at low spatial frequencies, a relatively large value of d must be used. In the following example, we compute the OTF from the coefficients of the right eye of observer 10, using a PSF size of 0.25 degree or 2.0 degrees. As can be seen, several features of the MTF are evident only for the larger value.

```
zc = ThibosHongBradleyChengData[[3,
10, 2]];
otfs = ZernikePointSpread[zc,
  Degrees→#,OTF→True,
  ImageSamples→1024]& /@ {.25,2};
Show[Reverse@MapThread
[RadialMTF[#1,PlotStyle→#2,
Degrees→#3]&,
  {otfs,{{Red,Dashed},Blue},{.25,
  2}}]
  ,Graphics[{Text["Degrees",
  Scaled[{.8,.9}]],
    Red,Text["0.25",Scaled[{.8,
    .7}]],
    Blue,Text["2.0",Scaled[{.8,
    .8}]]}]]
```

Resolution of aberration image

From the preceding parameters, another parameter may be derived: s – the pupil diameter in samples:

$$s = \frac{10^6 \pi d p}{180 \lambda}. \tag{10}$$

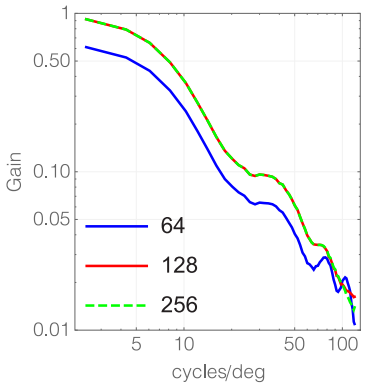
The parameter s determines the resolution with which the wavefront aberration image is sampled. This is of course closely related to the resolution of the OTF discussed above. Since λ and p are not free parameters, we see from Equation 10 that s is controlled by d. Thus depending on the application, we want d large enough to provide an accurate description of the OTF. We provide a function **PupilSamples** to compute values, and show examples for two wavelengths below. To compute a polychromatic PSF, we compute a linear combination of monochromatic PSFs, and thus we are concerned about the worst case. The largest value of s occurs for the smallest wavelength, so the worst case (requiring the most samples) is at 400 nm.

Avoiding wrap-around errors

From Equation 3 we see that the PSF is obtained through Fourier transformation of the generalized pupil function. In software, that function is represented by a finite discrete sampled image, and is transformed using the DFT. In two dimensions the DFT assumes toroidal boundaries, meaning the image can be considered the surface of a torus, with upper and lower edges, and right and left edges connected. This means that the DFT will produce "wrap-around" errors, unless the generalized pupil image is padded with zeros. This padding is reflected in the difference between s (the number of pupil image samples) and n (the number of PSF image samples). In general, we recommend that n be at least twice s. We provide a function to compute that number: **SafeImageSamples**. In the following

example, we show the radial MTF for three values of n: 64, 128, and 256. Note the very large discrepancy for 64, and the smaller but significant discrepancy for 128 (at the highest frequencies).

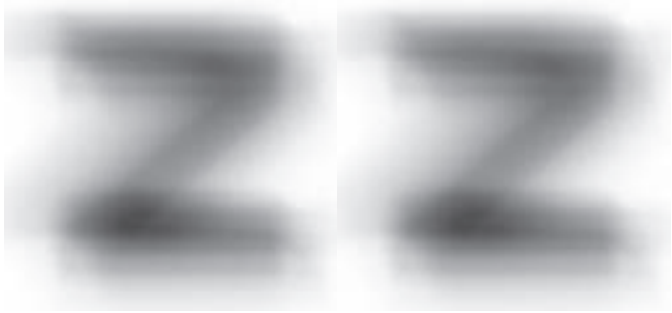
```
samples = {64,128,256};
otfs = ZernikePointSpread
[TestCoefficients,
  OTF→True,ImageSamples→#]& /@
  samples;
RadialMTF[otfs,PlotStyle→{Blue,Red,
{Dashed,Green}},
  PlotLegends → Placed[samples,
  {Left,Bottom}]]
```



Maintain consistent spatial resolution

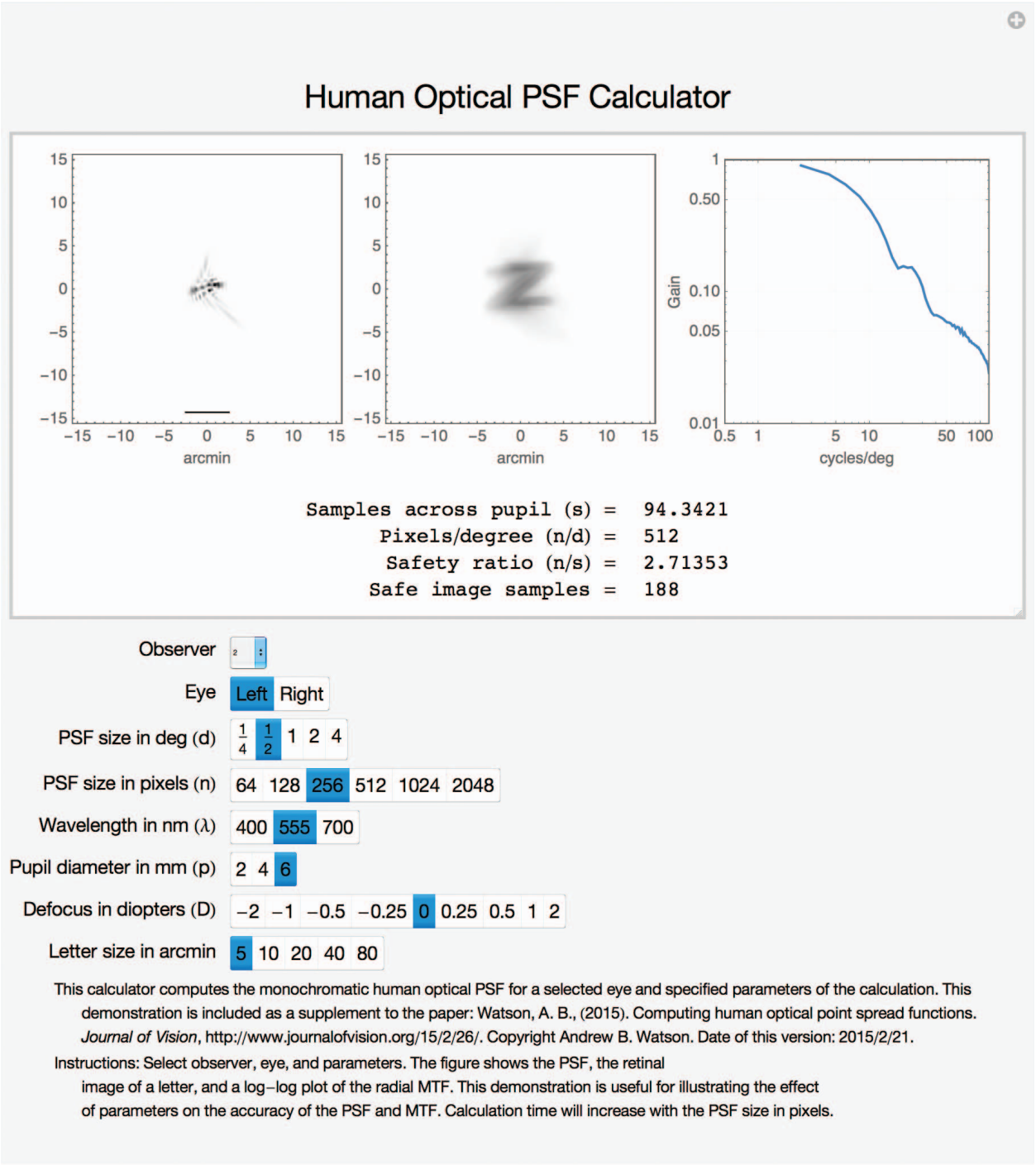
Beyond the constraints noted above, if the PSF is created in order to filter a source image, it is essential that the PSF and the image have the same visual resolution, in pixels/degree. To illustrate, we create two PSFs, with different pixel and degree dimensions, but the same visual resolution. We use each to convolve an image and show that the result is nearly the same.

```
psfs = MapThread[ZernikePointSpread
[TestCoefficients,
  Degrees→#1,ImageSamples→#2]&,
  {{0.25,0.5},{128,256}}];
Row[ImageConvolve[letter,
Reverse[Wrap[#]]] & /@ psfs]
```



Demonstration of effects of parameters

We have provided as a supplement to this article a tool that allows one to explore the effect on the PSF and OTF of various parameters, as shown in the figure below. After the user selects a set of parameters, the PSF, the retinal image of a letter, and the radial MTF are shown. Some of the statistics of the result are also shown. When the safety ratio n/s is less than 2, it is shown in red as a warning.



Speed of computation

At the heart of the computation of a PSF is the generation of the Zernike basis function images, one for each mode in the set of Zernike coefficients. This generation of many images can be slow. The time taken is proportional to s^2. If a polychromatic PSF is generated, the time will be multiplied by the number of wavelengths. However, the Zernike images need only to be computed once, for any particular value of s, or combination of p (pupil diameter), λ (wavelength), and

d (PSF image degrees). The function **ZernikeImage** remembers previously computed results, so subsequent calls are much faster.

Here is an example. The function **AbsoluteTiming** returns seconds. We evaluate the same expression twice.

```
AbsoluteTiming[ZernikePointSpread
[TestCoefficients];][[1]]

  0.277885

AbsoluteTiming[ZernikePointSpread
[TestCoefficients];][[1]]

  0.118230
```

Keywords: Zernike, PSF, blur, optics, retinal image, software

Acknowledgments

I thank Larry Thibos for early advice on optical computations and for providing the database of wavefront aberrations. I thank Pablo Artal for useful comments and calculations. I thank José Antonio Díaz Navas for generous assistance with pupil scaling formulas. This work supported by the NASA Space Human Factors Research Project WBS 466199.

Commercial relationships: none.
Corresponding author: Andrew B. Watson.
Email: andrew.b.watson@nasa.gov
Address: NASA Ames Research Center, Moffett Field, CA, USA.

References

Artal, P. (1990). Calculations of two-dimensional foveal retinal images in real eyes. *Journal of the Optical Society of America A*, *7*(8), 1374–1381, http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=2398446

Artal, P., Santamaria, J., & Bescos, J. (1989). Optical-digital procedure for the determination of white-light retinal images of a point test. *Optical Engineering*, *28*(6), 286687.

Atchison, D. A., & Scott, D. H. (2002). Contrast sensitivity and the Stiles–Crawford effect. *Vision Research*, *42*(12), 1559–1569, http://www.sciencedirect.com/science/article/pii/S0042698902000846

Bara, S., Arines, J., Ares, J., & Prado, P. (2006). Direct transformation of Zernike eye aberration coefficients between scaled, rotated, and/or displaced pupils. *Journal of the Optical Society of America A: Optics, Image Science, & Vision*, *23*(9), 2061–2066, http://www.ncbi.nlm.nih.gov/pubmed/16912732

Bará, S., Pailos, E., Arines, J., López-Gil, N., & Thibos, L. (2014). Estimating the eye aberration coefficients in resized pupils: is it better to refit or to rescale? *Journal of the Optical Society of America A: Optics, Image Science, & Vision*, *31*(1), 114–123.

Bracewell, R. (2003). *Fourier analysis and imaging*. New York: Springer.

Coe, C., Bradley, A., & Thibos, L. (2014). Polychromatic refractive error from monochromatic wavefront aberrometry. *Optometry & Vision Science*, *91*(10), 1167–1174.

Dai, G. (2011). Validity of scaling zernike coefficients to a larger diameter for refractive surgery. *Journal of Refractive Surgery*, *27*(11), 837–841.

Dai, G.-m. (2006). Scaling Zernike expansion coefficients to smaller pupil sizes: A simpler formula. *Journal of the Optical Society of America A*, *23*(3), 539–543, http://josaa.osa.org/abstract.cfm?URI=josaa-23-3-539

Dai, G.-m. (2008). *Wavefront optics for vision correction*. Bellingham, WA: SPIE.

Díaz, J. A., Fernández-Dorado, J., Pizarro, C., & Arasa, J. (2009). Zernike coefficients for concentric, circular scaled pupils: An equivalent expression. *Journal of Modern Optics*, *56*(1), 131–137, http://dx.doi.org/10.1080/09500340802531224

Goodman, J. W. (2005). *Introduction to Fourier optics* (3rd ed.). Englewood, CO: Roberts & Co.

Janssen, A. J. E. M., & Dirksen, P. (2006). Concise formula for the Zernike coefficients of scaled pupils. *Journal of Micro/Nanolithography, MEMS, and MOEMS*, *5*(3), 030501-030501-3, http://dx.doi.org/10.1117/1.2345672

Mahajan, V. N. (2010). Zernike coefficients of a scaled pupil. *Applied Optics, 49*(28), 5374–5377, http://ao.osa.org/abstract.cfm?URI=ao-49-28-5374

Mahajan, V. N. (2013). *Optical imaging and aberrations: Wavefront Analysis* (Vol. 3). Bellingham, WA: SPIE.

Marcos, S., Burns, S. A., Moreno-Barriusop, E., & Navarro, R. (1999). A new approach to the study of ocular chromatic aberrations. *Vision Research*, *39*(26), 4309–4323, http://www.sciencedirect.com/

science/article/B6T0W-3XT69YK-2/2/45dd4c75848bd1f6c762482708921da3

Metcalf, H. (1965). Stiles-Crawford apodization. *Journal of the Optical Society of America A*, *55*(1), 72–73, http://www.opticsinfobase.org/abstract.cfm?URI=josa-55-1-72

Nestares, O., Navarro, R., & Antona, B. (2003). Bayesian model of Snellen visual acuity. *Journal of the Optical Society of America A: Optics, Image Science, & Vision*, *20*(7), 1371–1381, http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=12868641

Ommani, A., Hutchings, N., Thapa, D., & Lakshminarayanan, V. (2014). Pupil scaling for the estimation of aberrations in natural pupils. *Optometry & Vision Science*, *91*(10), 1175–1182.

Ravikumar, S., Thibos, L. N., & Bradley, A. (2008). Calculation of retinal image quality for polychromatic light. *Journal of the Optical Society of America A: Optics, Image Science, & Vision*, *25*(10), 2395–2407, http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=18830317

Schwiegerling, J. (2002). Scaling Zernike expansion coefficients to different pupil sizes. *Journal of the Optical Society of America A: Optics, Image Science, & Vision*, *19*(10), 1937–1945, http://www.ncbi.nlm.nih.gov/pubmed/12365613

Stiles, W. S., & Crawford, B. H. (1933). The luminous efficiency of rays entering the eye pupil at different points. *Proceedings of the Royal Society B*, *112*, 428–450.

Thibos, L. N., Applegate, R. A., Schwiegerling, J. T., & Webb, R. (2002). Standards for reporting the optical aberrations of eyes. *Journal of Refractive Surgery*, *18*(5), S652–S660.

Thibos, L. N., Hong, X., Bradley, A., & Cheng, X. (2002). Statistical variation of aberration structure and image quality in a normal population of healthy eyes. *Journal of the Optical Society of America A: Optics, Image Science, & Vision*, *19*(12), 2329–2348, http://josaa.osa.org/abstract.cfm?URI=josaa-19-12-2329

Thibos, L. N., Ye, M., Zhang, X., & Bradley, A. (1992). The chromatic eye: A new reduced-eye model of ocular chromatic aberration in humans. *Applied Optics, 31*(19), 3594–3600, http://ao.osa.org/abstract.cfm?URI=ao-31-19-3594

Van Meeteren, A. (1974). Calculations on the optical modulation transfer function of the human eye for white light. *Journal of Modern Optics*, *21*(5), 395–412.

Voelz, D. G. (2011). *Computational fourier optics: A MATLAB tutorial*. Bellingham, WA: SPIE.

Watson, A. B. (2013). A formula for the mean human optical modulation transfer function as a function of pupil size. *Journal of Vision*, *13*(6):18, 1–11, http://journalofvision.org/content/13/6/18, doi:10.1167/13.6.18. [PubMed] [Article]

Watson, A. B., & Ahumada, A. J. (2012). Modeling acuity for optotypes varying in complexity. *Journal of Vision*, *12*(10):19, 1–19, http://journalofvision.org/content/12/10/19, doi:10.1167/12.10.19. [PubMed] [Article]

Watson, A. B., & Ahumada, A. J., Jr. (2008). Predicting visual acuity from wavefront aberrations. *Journal of Vision*, *8*(4):17, 1–19, http://journalofvision.org/content/8/4/17, doi:10.1167/8.4.17. [PubMed] [Article]

Wolfram Research, Inc. (2013). *Mathematica* (Version 9.0 ed.) [software]. Champaign, IL: Author.

Appendix

Mathematica notebook and demonstration

As a supplement to this report we provide a Mathematica notebook Zernike.nb. This contains definitions and example usage of the functions described here. It also contains the database of wavefront aberrations for 200 eyes collected by Thibos, Hong et al. (2002). We also provide a separate demonstration titled ZernikePointSpreadDemo.cdf. These files can be viewed with the free CDF player available at http://wolfram.com/cdf-player/.