Background

When attempting to detect exoplanets, one can observe the change in brightness of star spots. Sometimes however due to the inconsistency on the outer edges of a pixel, stars can seem to be changing brightness due to the light falling on a different part of the pixel.

If we were able to deduce how sensitivity changes at some point on the pixel, you could mathematically correct for this phenomena.

Experiment

We have access to a set of images containing grids of stars represented as spots. The intensity of these spots follow the airy disk pattern. Using these images, we can select a grid of pixels and store the intensity of each. Subsequently, for each sub-pixel, we can calculate the intensity and store it in a row vector. The response of a subpixel is defined as the sum of the intensity of all subpixels times the coefficient of the subpixel sensitivity map corresponding to that subpixel.

Intensities of certain subpixels can be determined by using the airy disk equation.

Airy Disk

The intensity of the airy patter can be given as a function of the angle of observation relative to the aperture

$$I(\theta) = I_0 \left(\frac{2J_1(x)}{x} \right)^2$$

where $x = \frac{\pi q}{\lambda N}$

$$\lambda = \text{wavelength}$$
$$q = \text{distance from aperture}$$
$$N = \text{f number}$$

Fourier Transform of Airy Disk

The Fourier transform of the airy disk function can be represented as the optical transfer function, defined as

$$OTF(\rho') = \frac{2}{\pi} \left[\cos^{-1}(\rho') - \rho' \sqrt{1 - \rho'^2} \right]$$

Where

$$\rho' = \rho / \rho_c$$
$$\rho_c = \frac{1}{\lambda N}$$

Image Stabilization

Our point spread function must also take into account the jitter of the centroids of the spots (the fact that they shift over time).

If we represent our gaussian as

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{1}{2} (x - \mu)^2 / \sigma^2 \right]$$

Then the Fourier transform is another gaussian

$$G(x) = \frac{1}{2\sigma^3\sqrt{2\pi}} \exp\left[-\frac{\pi^2 x^2}{1/2\sigma^2}\right]$$

For now, we shall remain in the Fourier space and take the inverse transform at a more convenient time.

Final PSF

$$PSF(\lambda, N, k_x, k_y, \sigma_x, \sigma_y) = G(x) \cdot \mathrm{OTF}(\rho')$$

Note:

$$\rho = \sqrt{k_x^2 + k_y^2}$$

Calculating response of subpixels

Definitions

$f(x, y)$ = PSF with (x, y) being offset from spot center

$s(x, y)$ = sub pixel sensitivity map with (x, y) relative to bottom left of pixel

If the center of a spot is at (x_0, y_0), the response of pixel (p, q) is given by:

$$\begin{aligned} r_{p,q} &= \int_{x=0}^{1} dx \int_{y=0}^{1} dy\, f(p + x - x_0, q + y - y_0)\, s(x, y) \\ &= \sum_{i=0}^{N_x - 1} \sum_{j=0}^{N_y - 1} S_{j+iN_y} \int_{x=0}^{\Delta x} dx \int_{y=0}^{\Delta y} dy\, f(p + i\Delta x + x - x_0, q + j\Delta y + y - y_0) \end{aligned}$$

if $F(k_x, k_y)$ is the Fourier transform of $f(x, y)$

$$f(x, y) = \int_{-\infty}^{\infty} dk_x \int_{-\infty}^{\infty} dk_y\, F(k_x, k_y)\, e^{2\pi i(xk_x + yk_y)}$$

We can plug this into $r_{p,q}$ as shown:

$$r_{p,q} = \sum_{nx=0}^{N_x - 1} \sum_{ny=0}^{N_y - 1} S_{j+iN_y}\, I_{p,q,n_x,n_y}$$

Where

$$\begin{aligned} x &= p + n_x \Delta x + \frac{\Delta x}{2} - x_0 \\ y &= p + n_y \Delta y + \frac{\Delta y}{2} - y_0 \\ I_{p,q,n_x,n_y} &= \int_0^{\Delta x} dx \int_0^{\Delta y} dy\, f(p + i\Delta x + x - x_0, q + j\Delta y + y - y_0) \\ &= \int_0^{\Delta x} dx \int_0^{\Delta y} dy \int_{-\infty}^{\infty} dk_x \int_{-\infty}^{\infty} dk_y\, F(k_x, k_y)\, e^{2\pi i(xk_x + yk_y)} \end{aligned}$$

We can separate

$$\begin{aligned} \int_{-\Delta x/2}^{\Delta x/2} dx\, e^{2\pi i k_x x} &= \frac{\sin(\pi \Delta x k_x)}{\pi k_x} \\ \int_{-\Delta y/2}^{\Delta y/2} dy\, e^{2\pi i k_y y} &= \frac{\sin(\pi \Delta y k_y)}{\pi k_y} \end{aligned}$$

Because we are numerically calculating these integrals, we can pick some large enough K_x and K_y because the integral will converge

$$I(x,y) = \int_{-K_x}^{K_x} dk_x \int_{-K_y}^{K_y} dk_y \, F(k_x,k_y) \, \frac{\sin(\pi \Delta x k_x)\sin(\pi \Delta y k_y)}{\pi^2 k_x k_y} \, e^{2\pi i (x k_x + y k_y)}$$

The imaginary part of this function integrates to zero, and the real part is the same in all four quadrants, thus we can do the following

$$I(x,y) = 4\int_{0}^{K_x} dk_x \int_{0}^{K_y} dk_y \, F(k_x,k_y) \, \frac{\sin(\pi \Delta x k_x)\sin(\pi \Delta y k_y)}{\pi^2 k_x k_y} \cos(2\pi k_x x)\cos(2\pi k_y y)$$

We can approximate this integral using the midpoint rule and breaking the regions into ν_x and ν_y parts.

$$
\Delta k_x = \frac{K_x}{\nu_x} \,\,\,\,\,\,\,\, \Delta k_y = \frac{K_y}{\nu_y}
$$

$$\Delta k_x = \frac{K_x}{\nu_x} \qquad \Delta k_y = \frac{K_y}{\nu_y}$$

When approximating the integral, we will have a double summation with m and n is running indices such that

$$k_x(m) = \left(m + \frac{1}{2}\right)\frac{K_x}{\nu_x}$$
$$k_y(n) = \left(n + \frac{1}{2}\right)\frac{K_y}{\nu_y}$$

$$I(x,y) = 4\frac{K_x K_y}{\nu_x \nu_y}\sum_{m=0}^{\nu_x - 1}\sum_{n=0}^{\nu_y - 1} F[k_x(m),k_y(n)]\frac{\sin(\pi \Delta x k_x)\sin(\pi \Delta y k_y)}{\pi^2 (m+\frac{1}{2})(n+\frac{1}{2})}$$
$$\cos\left[2\pi(m+1/2)\frac{K_x}{\nu_x}x\right]\cos\left[2\pi(n+1/2)\frac{K_y}{\nu_y}y\right]$$

We can separate this rather long expression into

$$\psi_{m,n} = F[k_x(m),k_y(n)]\frac{\sin(\pi \Delta x k_x)\sin(\pi \Delta y k_y)}{\pi^2 (m+\frac{1}{2})(n+\frac{1}{2})}$$

For ease of use we also should discretize x and y.

$$x_k = k\Delta x \qquad y_l = l\Delta y$$
$$\Delta x = \frac{1}{2K_x} \qquad \Delta y = \frac{1}{2K_y}$$
$$x_k = \frac{k}{2K_x} \qquad y_l = \frac{l}{K_y}$$

Now we have

$$I_{k,l} = I(x_k,y_l) = 4\sum_{m=0}^{\nu_x - 1}\sum_{n=0}^{\nu_y - 1}\psi_{m,n}\cos\left[\pi(m+1/2)\frac{k}{\nu_x}\right]\cos\left[\pi(n+1/2)\frac{l}{\nu_y}\right]$$

Which is exactly in the form of a DCT-II and 2-D REDF10 transform described by FFTW.

Solving for subpixel coefficients

Let P_x and P_y be the number of pixels in the x and y direction respectively, and let p_x and p_y be iterators over the grid of pixels.

Let $I_{p,q}$ be the intensity of the q^{th} subpixel of the p^{th} pixel

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ \vdots \\ r_{p_y+p_xP_y} \end{bmatrix} = \begin{bmatrix} I_{0,0} & I_{0,1} & I_{0,2} & \cdots & I_{0,n_y+n_xN_y} \\ I_{1,0} & I_{1,1} & I_{1,2} & \cdots & I_{1,n_y+n_xN_y} \\ I_{2,0} & I_{2,1} & I_{2,2} & \cdots & I_{2,n_y+n_xN_y} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_{p_y+p_xP_y,0} & I_{p_y+p_xP_y,1} & I_{p_y+p_xP_y,0,2} & \cdots & I_{p_y+p_xP_y,n_y+n_xN_y} \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ \vdots \\ S_{n_y+n_xN_y} \end{bmatrix}$$

It is quite evident that standard matrix solving will not work here due to a dimension mismatch. In order to solve for the subpixel coefficients, we can employ least squares.