Final Report: Exoplanet Data Reduction Tool

Hannah Diamond-Lowe & Zakir Gowani
CMSC 12300: Big Data
Professor Anne Rogers, TA Gustav Larsson

The driving motivation of this project is to simplify a key step in the process of analyzing exoplanet data from the *Spitzer Space Telescope*. The goal of the step is to determine exactly how much of a data frame (a FITS image) is useful for analysis. Each FITS image consists of 64 32x32 pixel frames. To use too much of a data frame would yield abundant background noise that would dilute the relevant information. To use too little would yield incorrect conclusions about the nature of the exoplanet we are trying to study. Currently, the process of determining exactly how much of a data frame to use is arduous and time consuming. It involves manual testing aperture sizes (Figure 1) down to tenths of a pixel, running the data reduction pipeline through to completion and then comparing the standard deviation of the resulting residuals. Though the process is simple, it is time consuming, since running the data through to the final modeling stage requires a significant amount of computing power. Appropriate aperture sizes usually range from around 2 to 3 pixels, and while it is possible to make intelligent guesses to narrow down the possibilities, it typically requires at least attempts to find and confirm the best aperture size to use for a given dataset.

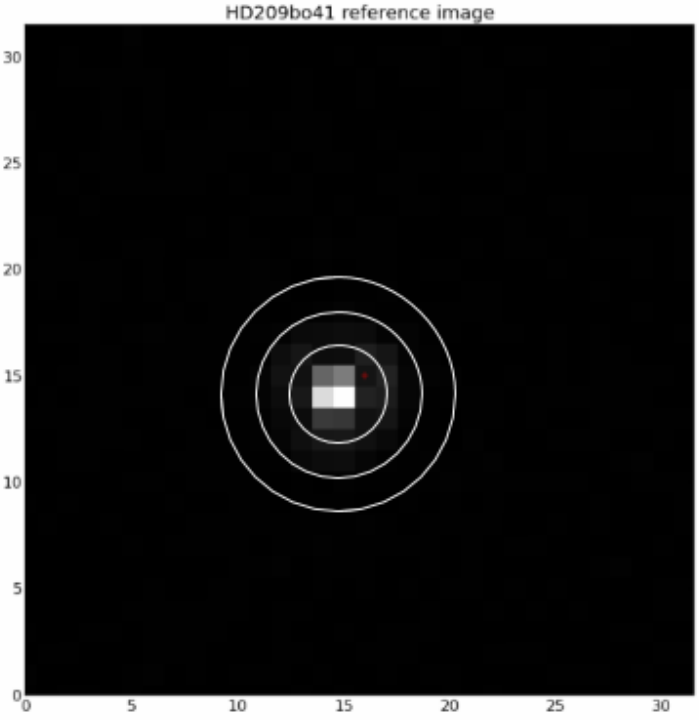

Figure 1 - Illustration of three different aperture sizes for determining how much data to use from the frame.

We aim to shrink the amount of time spent on this step by developing an algorithm that will narrow down the possibilities to a one-pixel range or smaller. We still test a series of aperture sizes, but instead of running the data reduction through to completion, we use an algorithm to guess at the appropriate pixel size. The algorithm utilizes a Gaussian curve approximation for a point spread function is:

$$I(q) = I_0 \exp\left(\frac{-q^2}{2\sigma^2}\right)$$

where I is a measure of the amount of light, q is the radius of the aperture in pixels, and σ is the standard deviation, which for the purposes of this assignment, can be approximated as:

$$\sigma = 0.45\lambda F$$

where λ is the wavelength in which the data was taken (in microns) and F is the f-stop number, which can be calculated by dividing the focal length of the *Spitzer Space Telescope* by its diameter. Using the Gaussian curve by itself will always favor large aperture sizes, since the larger the aperture, the more flux is admitted as compared to the "noise" flux left outside of the aperture. We therefore utilize a rising exponential penalty function:

$$P(q) = \exp(q - 5)$$

where P is the penalty that is a function of q, the aperture radius in python.

Figure 2 depicts the Gaussian approximation curve we use to weight the amount of flux in a given annulus. As can be seen from the plot, information closest to the brightest region of the frame (small aperture radius) is heavily weighted, while regions on the outer edges of the frame (large aperture radius) contribute least to the overall flux.

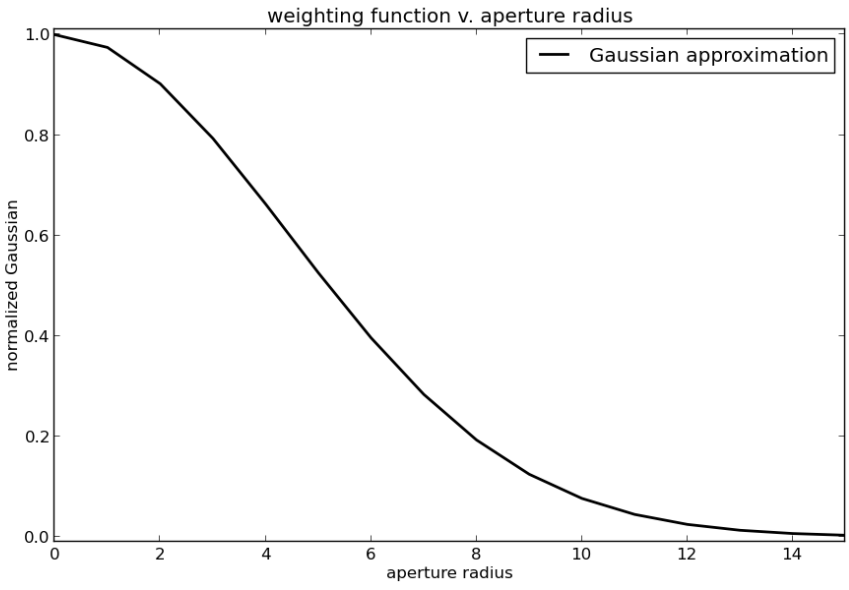

Figure 2 - Gaussian approximation curve developed from parameters relevant to our data.

We determine the best aperture size by using the Gaussian curve to weight the amount of flux within a given aperture size, or the signal. We then weight the flux outside a given aperture, or the noise. We take the signal-to-noise ratio and subtract a penalty function based on how large the aperture radius is. Figure 3 demonstrates the weighted flux as a function of aperture size, along side the penalty associated with using larger apertures.

Depending on the aperture we are testing, we divide the integrated region to the right of the aperture radius in question by the integrated region to the left of the aperture radius in questions. This gives the signal-to-noise ratio we are looking for. We then subtract the value of the penalty function for that aperture radius.

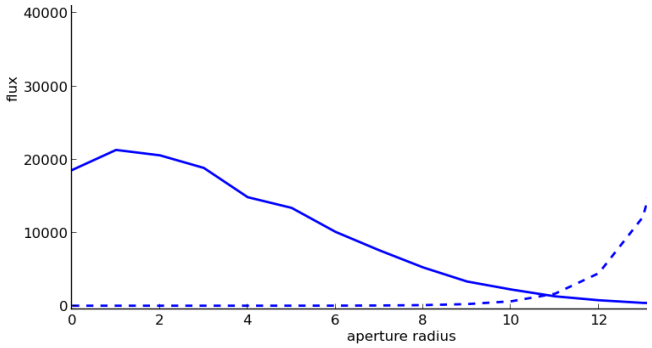

Figure 3 - Flux, multiplied by Gaussian weighting curve (Fig. 1), along side penalty function

After producing the signal-to-noise ratio, minus the result of the penalty function, for each aperture radius, we compare the results. Because we are interested in the most signal as possible, we take the two aperture radii that result in the highest value of signal-to-noise minus penalty. These become our upper and lower bounds for our ideal aperture radius. Narrowing down the possible aperture radii to this one-pixel region is a big step forward in what is usually a long and tedious process of finding the best aperture size for a given dataset.

In order to develop the most efficient process, we test several methods for achieving the best aperture radius range. The first is to simply find the best aperture range for every data frame in the dataset and then take an average of the upper and lower bounds. While this might be the most precise solution, it takes a long time to run (even in parallel) and preserves too much information, especially for datasets where the frames are taken at short exposure times (~0.1 seconds). The expensive part of the process is finding the signal-to-noise ratio for each aperture radius we are testing. One way to cut down on processing time is to create one average frame from the 64 original data frames in each FITS file. This requires averaging the amount of flux from each pixel position in the 32x32 array, across all 64 frames. Though this still requires looking at each frame, the computation coast of averaging a series of numbers is much smaller than the cost of designating fifteen different annuli and comparing them for each frame. Once we have an average frame we process it as we would one individual data frame, and find the best aperture range for the FITS file as a whole. We then parallelize this process across all FITS files, as opposed to individual frames.

Another approach to this problem is to sample a few data frames from each FITS file and use these samples to get the best aperture range. Our code has the option to sample 4 of the 64 frames in a FITS file and determine the best aperture range for these files. This takes about four times as long as the previous method described, but it does provide a higher resolution solution which could be advantageous, especially for datasets in which data frames are taken for long exposures. Long exposures inherently create an “average” frame, and changes between these long exposure frames can have an impact on optimal aperture sizes.

One of our project goals was to implement our algorithm in parallel on the Research Computing Cluster using the MPI library for Python. Because the transition to parallel processing is relatively simple for this particular analysis, we focused on making the most efficient use of our resources. We experimented with a number of parallel processing methods and conducted a series of timing experiments to determine which methods were fastest. In each of our methods, the input is a directory containing FITS files to be analyzed; the output is a tab-delimited text file which maps FITS filenames to aperture numbers.

In the first method, labeled in our script parallelizer.py as "Method A", each thread contributes to the analysis of a single FITS file. The 64 images in a FITS file are distributed as evenly as possible to the set of available threads (systems with thread numbers that do not divide cleanly into 64 are accounted for). The main thread collects aperture sums from its peers, and then divides by 64 to obtain the average calculated aperture for the star exposure.

In the second method, labeled "Method B" in our script, the main thread obtains the full list of FITS files and distributes a set of files to each thread to process in their entirety. This approach proved to be ~30% slower than Method A, which we believe is due to the general fact that parallel processing time is controlled by the slowest thread. When threads process the same FITS file in Method A, the difference in thread processing time is much less than when each thread processes entire files.

Methods A and B are inefficient in that they calculate apertures for each of the 64 images in the FITS file; apertures tend to remain constant between adjacent images, as they are taken in rapid succession, excluding cases of interest where exoplanet movement contributes to starlight variations.

To adapt to these conclusions about our dataset, we moved in a different direction and tested two approximations for which some accuracy was sacrificed for efficiency. The third approach, labeled "Method C", averages the 64 FITS frames into a single image, reducing noise; Method D performs a sampling of the 64 frames, randomly selecting 4 representatives from the 64 images with the requirement that each representative come from a different 16-image sector of the fileset (recall that the images are ordered sequentially in time).

As expected, the latter two methods performed very differently in our timing experiments. Figure 4 contains the timing output for the each of the four methods on a set of 640 images (10 FITS files) on a system with 4 cores. Note that while the latter two methods discard information, the majority of their outputs are equivalent to the outputs of the first two methods.

Equipped with an optimized processing method, we performed our analysis on a 10 GB FITS file dataset. Two examples of interesting and visually appealing results from our analysis are depicted in Figure 5. The loss of information in the immediate vicinity of the alleged star object is minimal due to the stringency of our approximated PSF.

Method A	288.1 s	Threads cooperate on files

Method B	430.0 s	Threads work on disjoint filesets
Method C	11.5 s	Disjoint filesets; frames averaged into one
Method D	21.8 s	Disjoint filesets; 4 random representative frames

Figure 4 - Timing experiment outputs for a variety of parallelization methods.

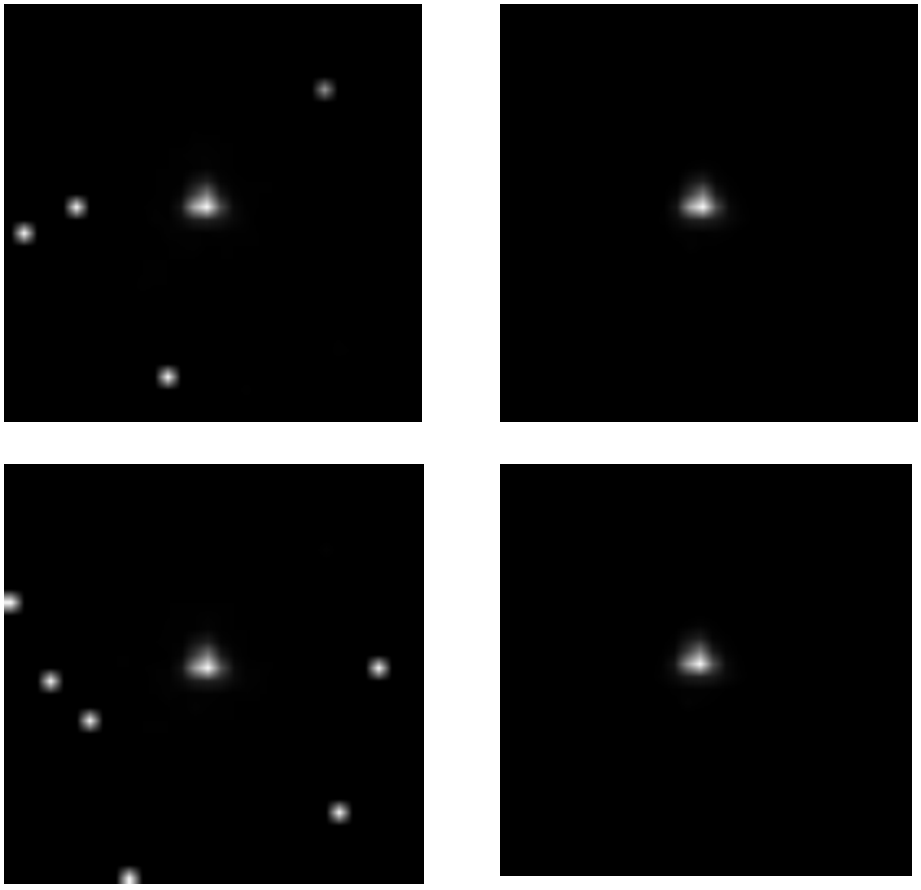

Figure 5 - Algorithm results on first frame of FITS file 4054. Left: pre-processing. Right: result of aperture calculation script. Output of pixel radius tuple (3,2). The radius 3 result is depicted.