Afrah A. Ali

Physics 139

4 December 2022

Lab 6 Report

In this lab, I utilize the dark-subtracted, flat-fielded Lick Shane AO images from Lab 5 to measure the flux of a targeted star. The goals of this lab are to characterize the point-spread-function (PSF) in each reduced image, to perform aperture photometry, and perform photometry via PSF fitting.

Please note that the code for this lab has been divided into two different PDF files. The first file is titled "AFRAHAA_PYTHON_LAB6.pdf" and the second file is titled "AFRAHAA_PYTHON_LAB6_CONTINUED.pdf". Please refer to the files in the specified order.

PART ONE

Using one of the individual dark-subtracted, flat-field corrected images of the star, we are asked to measure the full width at half maximum (FWHM) of the star in both the x and y directions. To do this, I first created holders for the flat images and dark images. To create a holder for the flat images, I created a loop that appends each flat file to an initially empty array titled *flat_holder*. To create a holder for the dark images, I first saved every individual image as its own variable using the *fits.open()* function. I then saved all of those variables into one master list titled *dark_holder*. To obtain the normalized flat image needed to measure the FWHM, I divided *dark_holder* by its median and saved it as a new variable titled *normalized_flat*. This process resulted in obtaining a dark-subtracted, flat-field corrected image of the star.

After obtaining this image, I initialized the x and y values from the image using the following lines of code:

```
x = np.median(normalized_flat[:,:,0], axis = 0)
y = np.median(normalized_flat[:,:,0], axis=1)
```

As you can see, the x values are initialized as being on the zeroth axis, and the y values are initialized as being on the first axis. They are both, however, located within the same index of the normalized flat image.

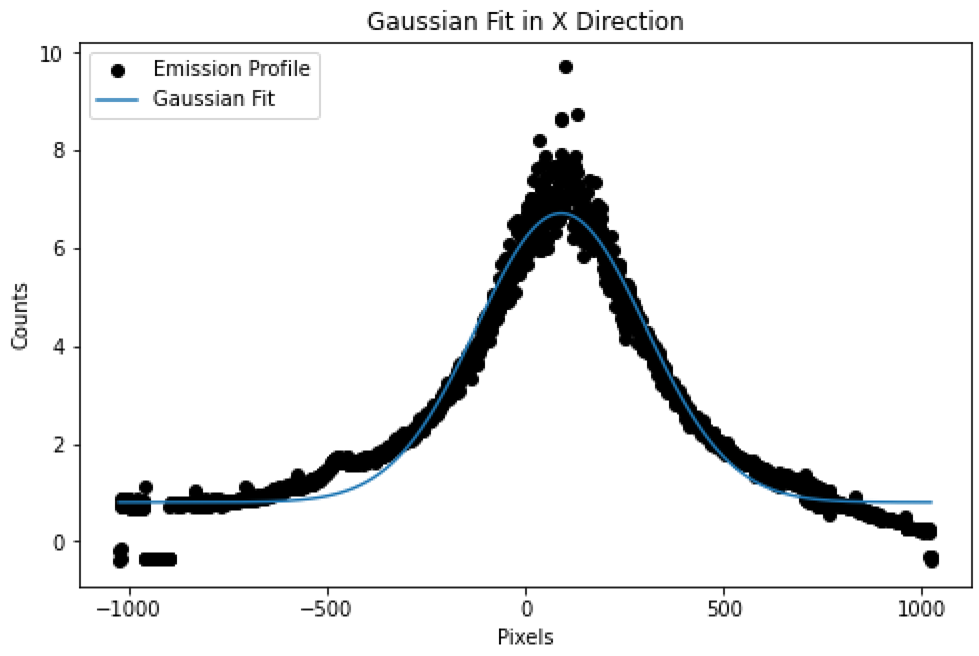
Next, I declared two variables called *xdata* and *ydata* and filled them with ranges for each variable. I did this through the following lines of code:

```
xdata = np.arange(-1024,1024)
ydata = x # Focusing on x direction so set y to x to standardize
```

Notice that the *ydata* has been set to *x*. These lines of code are being used to generate a Guassian fit in the x-direction, so to standardize the data for the x-direction, I have set the *ydata* to be *x*.

To initialize the Gaussian fit, I used the code provided to us. I then created a figure to plot on top of. To plot the measured emission profile, I plotted *xdata* against *ydata.* To plot the Gaussian fit of the emission profile, I plotted *xdata* against the Gaussian fit for *xdata* which is obtained through the simple algorithm *g(xdata)*. I overplotted the Gaussian fit on top of the emission profile.

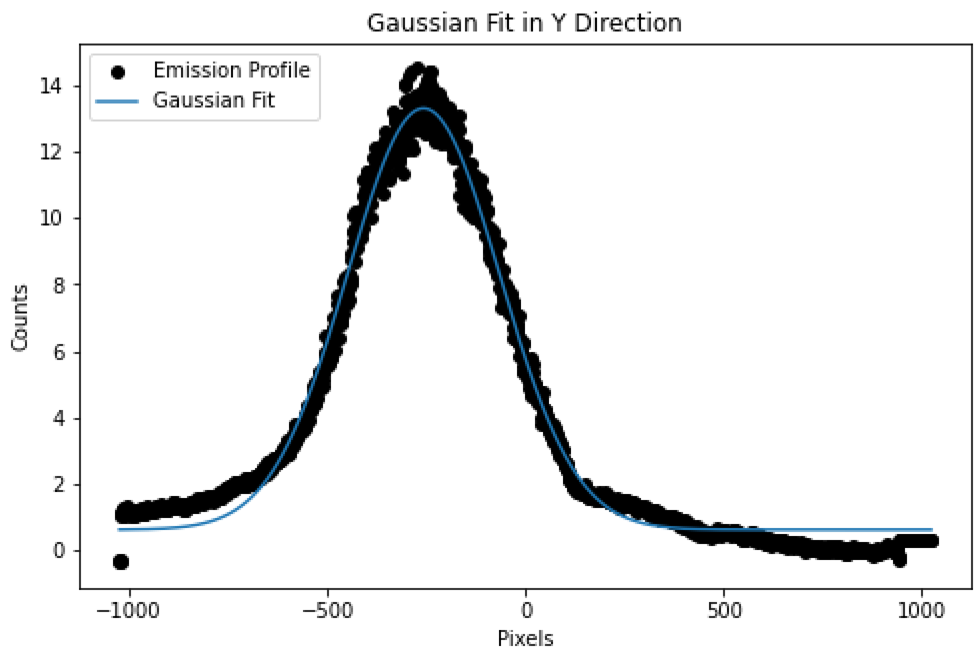
The following plot includes the emission profile as well as its Gaussian fit in the x-direction:



To plot the emission profile and its Gaussian fit in the y-direction, I followed the exact same process as I described above after making the following change to my code:

```
xdata = np.arange(-1024,1024)
ydata = y # Focusing on y direction
```

Notice that the only difference between the creation of the plots in the x and y directions is the variable that *ydata* is equal to. When plotting for the y-direction, *ydata* must be set to *y*. The following plot includes the emission profile as well as its Gaussian fit in the y-direction:



It is clear in both of the plots above that the Gaussian fit matches very closely with the measured emission profile. Generally, the FWHM for a Gaussian is

Since these images were produced using wholistic arrays and lists that contain information about the dark images, flat images, normalized flat images, and science frames instead of individual images, I am unsure about how to repeat the process above for two other science images. In the code used to produce the two plots above, I use an image produced through the following line of code:

```
normalized_flat = dark_holder/np.median(dark_holder)
```

As you can see, the variable dark_holder being used in this line of code is a wholistic list; not just a single dark image. Because of this, I cannot produce additional plots that pertain to individual images.

To fix this problem, I revised my method to do part one of this lab. The revised version is described in the following section.

PART ONE - REVISED

First, I saved all the given files into separate lists based on the type of file and the exposure time of the file. I then found the median of each list containing dark images and saved them into the variables *d15_med*, *d44_med*, and *d150_med*. I then subtracted these median dark values from the flat images and target images using two respective *for* loops. It is important to note that I only subtracted the dark values that had the same exposure time as the flat images and target images that they were being subtracted from. After this process, I was left with a list filled with dark subtracted flat images titled *flat44s_d* as well as a list of dark subtracted target images titled *tar150s_d*. I found the median of *flat44s_d* and then created a subsection of the flat image titled *flat_zoom*. I then took another median of this subsection and titled it *zoom_med*. To normalize the subsection of the frame for the flat image, I divided the median flat image by the median of the subsection by the median of the subsection itself. I took a subsection of the resulting image and saved it as a variable titled *zoomed_norm_flat*. I normalized the given science images by taking equally sized subsections from them and dividing those subsections by *zoomed_norm_flat*. At this point in the code, I had now made all the necessary manipulations to the given images, making them applicable to part one of this lab.

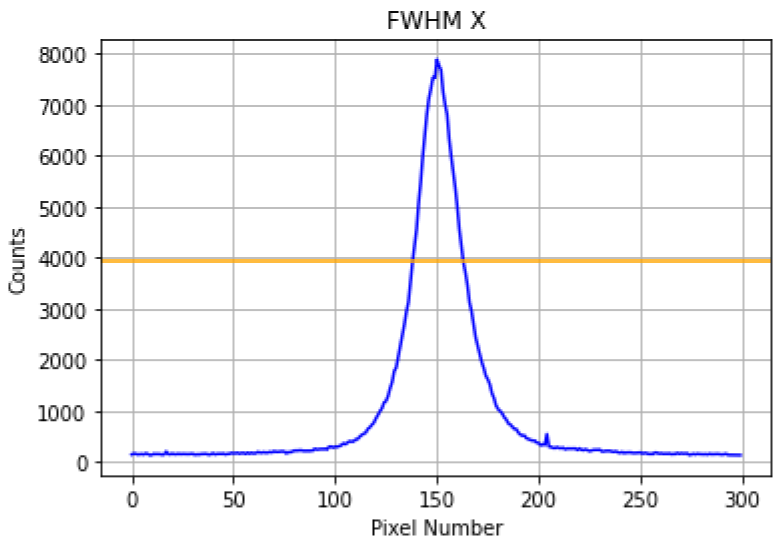
I also created the co-added version of the science image in this part of the lab. To do this, I created a function called *centerfinder()* that splits a given image in half repeatedly until there is an 8 x 8 pixel grid that contains the center of the star displayed in the image. This function operates under the assumption that the star being displayed in an image is the brightest object in the entire image. This function assumes that the half of the image with the higher average value must then contain the star. I ran this function for each of the three normalized science image frames and saved the centered versions of the images into a list called *snippets*. After creating three equally sized, centered images, I co-added all the images onto one main image. I did this by creating an empty array that is of the same size as each centered image and filling it with all the snippets of centered images.

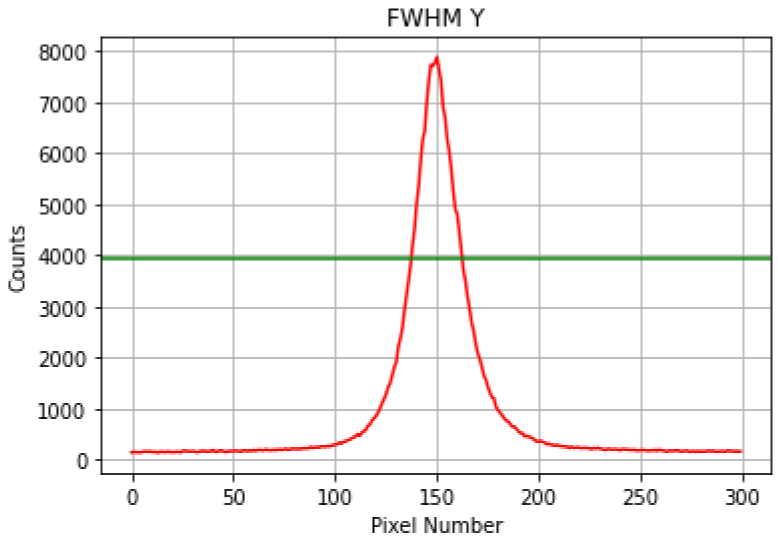
To find the FWHM of the emission profiles, I first found the half maximum value of the emission profile distribution. I created a function called *FWHM()* The function collects all the pixels within 1σ of the half maximum value, then all the pixels that are within 0.5σ of the half maximum value, then all the pixels that are within 0.33σ of the half maximum value, and so on. This iterative process continues until there are only two values that match the given requirement for σ, and those two values are saved as the two half maximum values. To find the FWHM value, I simply subtract the smaller value from the two found by the larger one. The FWHM values for each of the science frames are displayed in the table below.

Table 1 - Measured FWHM Values & FWHM Values Determined by Gaussian Fits

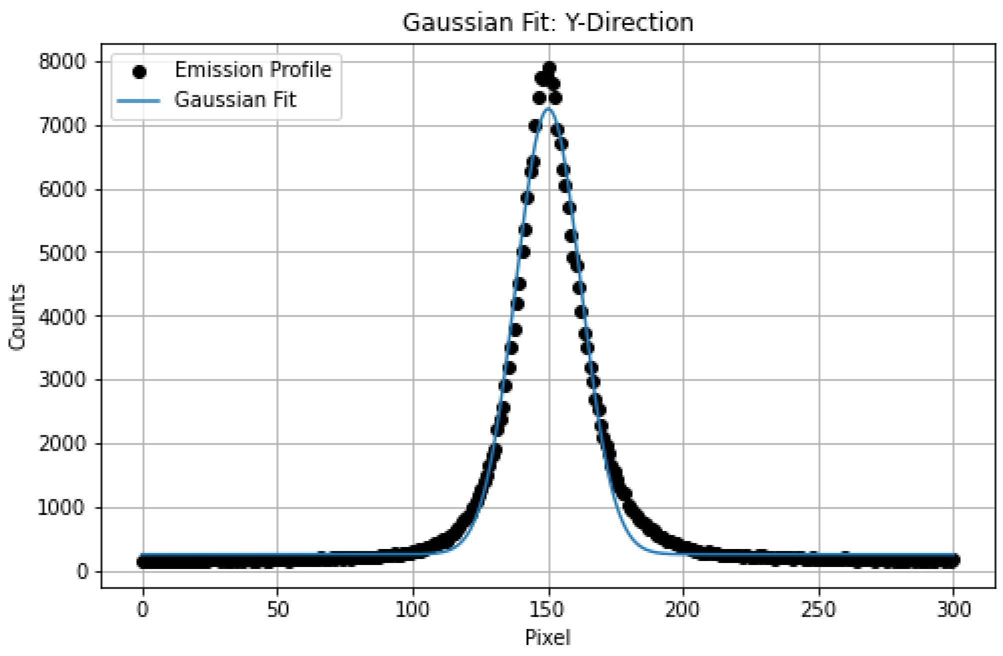
Frame	FWHMx (Pixels)	FWHMy (Pixels)	FWHMx (Arcsec)	FWHMy (Arcsec)	FWHMg,x (Pixels)	FWHMg,y (Pixels)
s0576	19	21	0.627	0.693	22.3	22.96
s0579	25	25	0.825	0.825	28.44	28.25
s0581	18	20	0.594	0.66	22.51	23.28
s0585	22	20	0.726	0.66	24.2	22.99
Co-Added Img	20	21	0.66	0.693	23.84	22.96

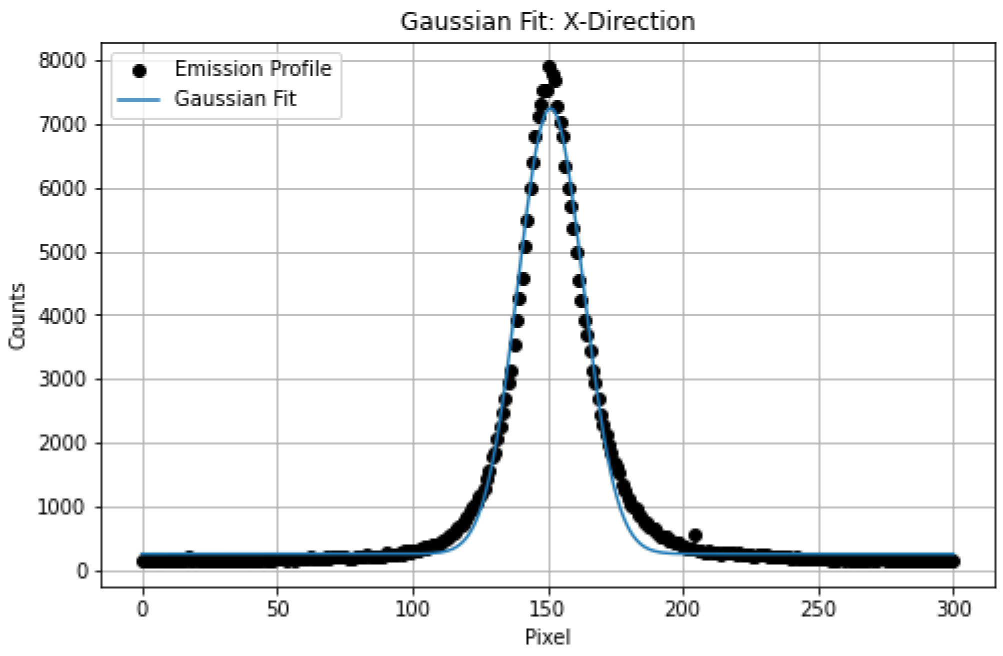
Below are the plots generated from the process of finding the FWHM for one of the normalized science frames:





The emission profiles that the graphs above pertain to as well as their Gaussian fits were produced using the code provided to us in the lab document. Those plots are shown below:





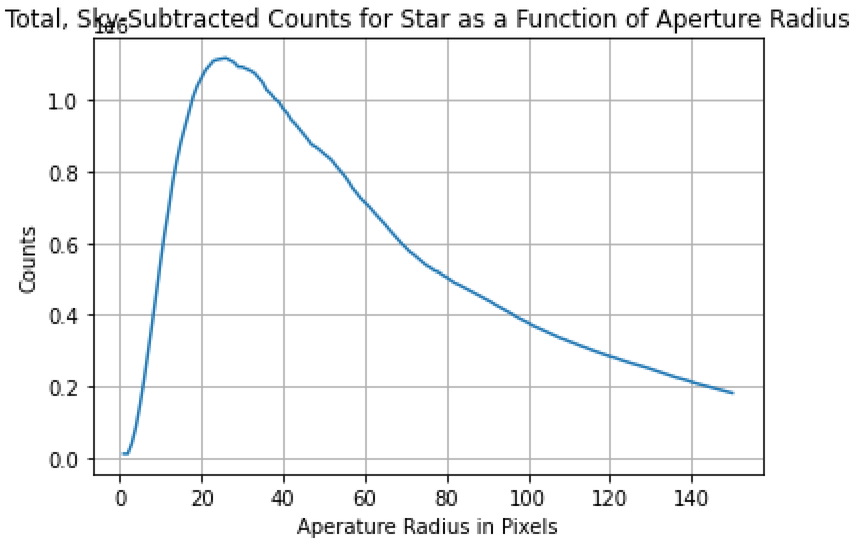
Please refer to the attached code for the FWHM plots, emission profiles, and Gaussian fits for the rest of the science images.

As you can see from the graphs above, the Gaussian distributions fits very closely with the measured emission profiles in the X and Y directions. The measured peak emissions are only slightly higher than the peaks of the Gaussian distributions.

The computed FWHM for a Gaussian distribution is generally equal to 2.355σ (approximately equal to 20) which is very close to 22.3, the FWHM of the fitted Gaussian distributions above.

The width of the point spread function (PSF) varies minimally from frame to frame. All of its values are roughly equal to 20, as is seen in Table 1. This aligns with the PSF of the final, co-added image which has width dimensions of 24 x 23. The final, co-added image is very circular.

PART TWO
In this part of the lab, we are asked to use the final, combined, reduced image for the targeted star to measure the curve of growth for the star. This is done using an aperture to measure the average sky brightness and appropriate subtracting. To do this, I created a function called *func()* that finds the average sky brightness and subtracts it from the star. This function then plots the pixel counts as a function of aperture radius (in pixels). The following graph is the curve of growth for the star with an aperture radius of 1.

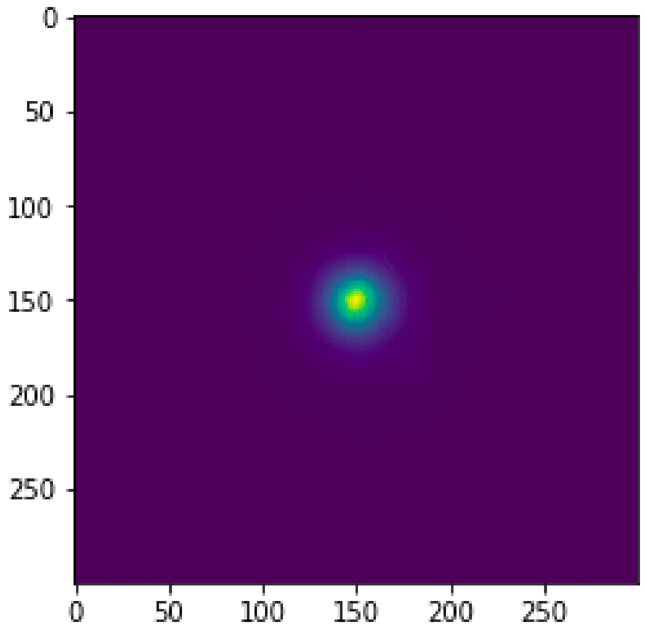


I varied the aperture radius and replotted the graph multiple times, but it changed minimally and still followed the general trend of the graph above. It is important to note that the peak of all the graphs produced always occurred around the value *x = 25 pixels*. Because of this consistency, this aperture size is ideal to measure the total flux of the star; an aperture size of 25 captures as much of the star's light as possible.

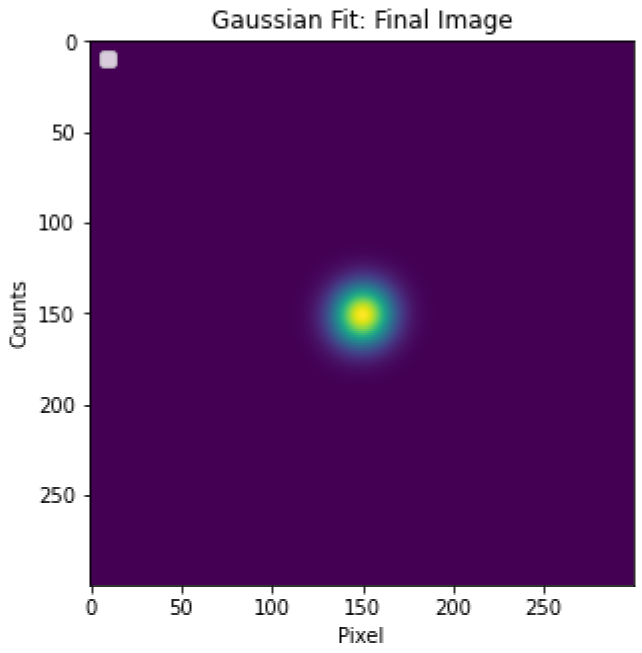
I used this value to find the total flux of the star by first taking a subsection of the co-added image that focuses on the star and taking the sum of that subsection. I then divided that sum by the exposure time, 150 seconds, to get the total counts per second for the targeted star. The total flux of the star is *45421.764 counts/second*.

PART THREE
In this part of the lab, we are asked to fit a 2D Gaussian to the stellar emission of the final, combined, and reduced image of the targeted star. I did this using a process very similar to the one used to generate the 1D Gaussian fits in part one. The only difference is that I added a variable called *z_data* that holds information about the co-added image.

The final image is displayed below:



The Gaussian fit for this image is displayed below:



The parameters for the Gaussian fit above are as follows:

Amplitude 1: 7867.28 pixels
X Mean: 150.09 pixels
Y Mean: 150.75 pixels
X STD: 11.83

Y STD: 11.51
θ: 84 ArcSec
Amplitude 2: 194.31

There is very little variation amongst the parameters obtained from the 2D Gaussian fit and the parameters obtained from part one. The peak of the 2D Gaussian fit is slightly lower than the peak of the 1D Gaussian fit. The values for the means and standard deviations of the 2D Gaussian fit are greater than those of the 1D Gaussian fit. The measured flux from the 2D Gaussian fit is much less than the flux measured using aperture photometry. The flux measured by the 2D Gaussian fit is 41989.58 counts/pixel, and the flux measured using the aperture photometry is 45421.76 counts/pixel.