Astronomy Lab: Report 6

Kiavash Teymoori 99100585
Alireza Talebi 400109821

Introduction

The goal of this experiment is to analyze the Point Spread Function (PSF) of stars in an astronomical image by modeling their intensity profiles using Gaussian functions. By doing this, we aim to estimate the Gaussian width (σ) for each star, which represents the effective seeing in arcseconds. From the distribution of these widths, we will calculate statistical properties such as the mean and variance to assess the overall image quality and the uncertainty in our measurements.

Experiment

Star Selection

We began by loading a .fit file containing the astronomical image and converted it into a NumPy array. The image dimensions were extracted. Two empty arrays, sigma-hist and delta-sigma-hist, were initialized to store the Gaussian width values and their uncertainties for each selected star.

```
#initializing_data

starsss=pd.read_csv("stars.csv")
loc_x = np.array(starsss["x0"].tolist())
loc_y = np.array(starsss["y0"].tolist())

img = fits.open('light-r_20C-2023_10_10-exp00.03.00.000-1x1_High_3.fit')
img_data = img[0].data
img_len_x, img_len_y = img_data.shape
asec_per_pix=0.047
sigma_hist=np.array([])
delta_sigma_hist=np.array([])
```

We used the software Siril to select stars. Siril allows users to define a region and identify individual stars using its "Pick a Star" tool. By applying this tool multiple times, Siril generates a list of star coordinates and parameters such as Full Width at Half Maximum (FWHM). We exported this list as a CSV file and processed it using the Pandas library in Python for access to the star positions.

Code Analysis

A custom centroid-finding function was used to compute the center of each star based on weighted averages of pixel intensities.

```
#centeroid finder
def centeroid(pic):
    x_len, y_len = pic.shape
    c_cord = np.array([0,0])
    weight = np.sum(pic)
    for i in range(x_len):
        for j in range(y_len):
            c_cord[0] += pic[i][j]*i
            c_cord[1] += pic[i][j]*j
    return c_cord[0]//weight , c_cord[1]//weight
```

We then extracted a 100×100 pixel frame around each stars centroid. Since Siril uses an inverted y-axis compared to standard array indexing in Python, we corrected this by adjusting the coordinate transforma-

tion during cropping.

```
#crop_around
def crop_around(stars_img, star_x, star_y):
    return stars_img[star_y-50: star_y+50].T[star_x-50: star_x+50].T
```

We selected a horizontal slice through the star's centroid and normalized the pixel intensity values. This 1D profile was fitted to a Gaussian function using scipy.optimize.curve-fit.
Instead of directly fitting (which could cause division-by-zero errors), we used the parameter "a" and later converted it back to σ. The uncertainty in σ ($\Delta\sigma$) was calculated using error propagation from the covariance matrix returned by the fitting function.

We had a lot of bad pixels we used a filtering function to scan the image for pixels that significantly deviated from the median of their 8 neighboring pixels. If such pixels were found, they were replaced with the local median to ensure that we have an accurate Gaussian profile.

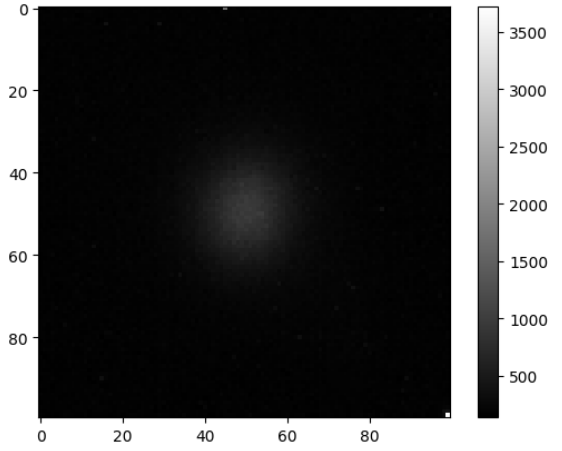
```
#bad pixel removal
def bad_pix_rem(star_img):
    for i in range(1,99):
        for j in range(1,99):
            nhood=np.array([star_img[i-1][j],star_img[i-1][j-1],star_img[i-1][j+1],star_img[
                                                    i][j-1],star_img[i][j+1],
                            star_img[i+1][j],star_img[i+1][j-1],star_img[i+1][j+1]])
            nhood_med=np.median(nhood)
            if star_img[i][j]>200 + nhood_med:
                star_img[i][j]=nhood_med
    return star_img
```

The experiment was repeated for all selected stars. Each stars σ and $\Delta\sigma$ values were stored, and a histogram of was generated.
We repeated steps 3 and 4 for twenty-two individual stars, selected independently. For each star, we recorded the Gaussian width (σ) and its uncertainty ($\Delta\sigma$). A histogram of values was then plotted. The mean and variance of this histogram were calculated.

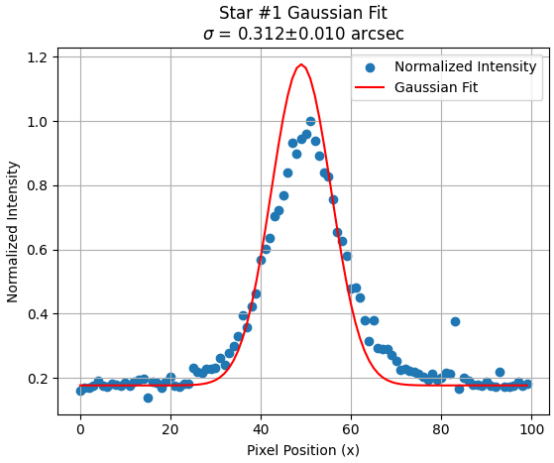
```
for i in range(len(loc_x)):
    star_0_img=bad_pix_rem(crop_around(img_data, int(loc_x[i]), img_len_y - int(loc_y[i])))

    #corrected_pic_print
    plt.imshow(star_0_img, cmap='gray')
    plt.colorbar()
    plt.show()

    #curve_fit_setup

    x_data=np.arange(0,100)
    intensity=star_0_img[0:100][int(centeroid(star_0_img)[0])]
    max_intensity=intensity.max()
    intensity=intensity/max_intensity

    def gauss(x,a,b):
        return np.exp(-a * (x-centeroid(star_0_img)[0])**2)+b

    popt, pcov=curve_fit(gauss,x_data,intensity)
    delta_a=(pcov[0][0])**(1/2)
    sigma_pix=(popt[0]**(-1)/2)**(0.5)
```

```
    delta_sigma_pix=(sigma_pix**3)*delta_a
    sigma=sigma_pix*asec_per_pix
    delta_sigma=delta_sigma_pix*asec_per_pix
    sigma_hist=np.append(sigma_hist,sigma)
    delta_sigma_hist=np.append(delta_sigma_hist,delta_sigma)

    # intensity profile for one star
    plt.figure()
    plt.scatter(x_data, intensity, label="Normalized Intensity")
    plt.plot(x_data, gauss(x_data, *popt), color='red', label="Gaussian Fit")
    plt.xlabel("Pixel Position (x)")
    plt.ylabel("Normalized Intensity")
    plt.title(f"Star #{i+1} Gaussian Fit\n$\sigma$ = {sigma:.3f}  {delta_sigma:.3f} arcsec")
    plt.legend()
    plt.grid(True)
    plt.show()
```

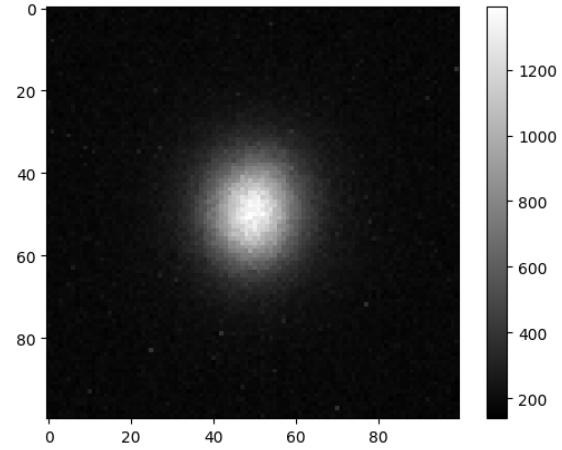
Results



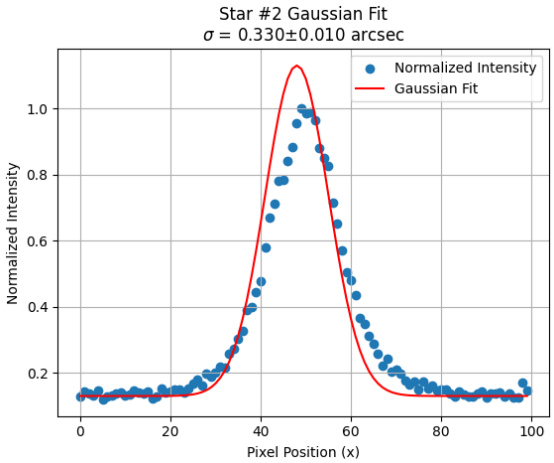
Star 1- Image



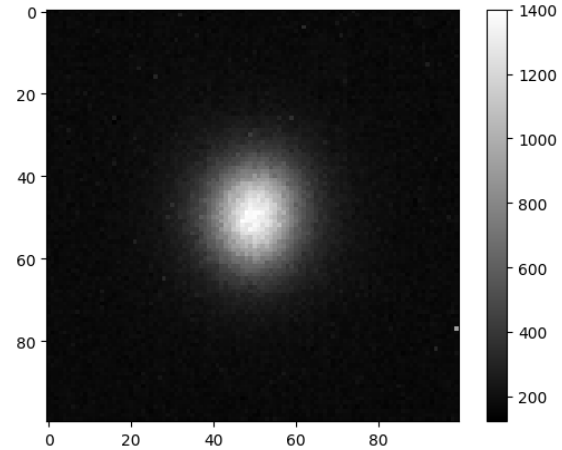
Star 1- Fit



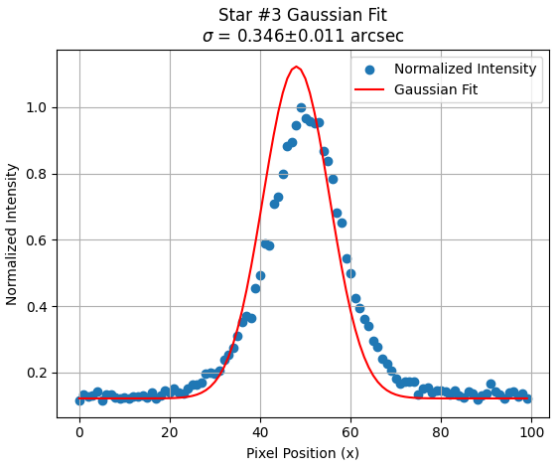
Star 2- Image



Star 2- Fit

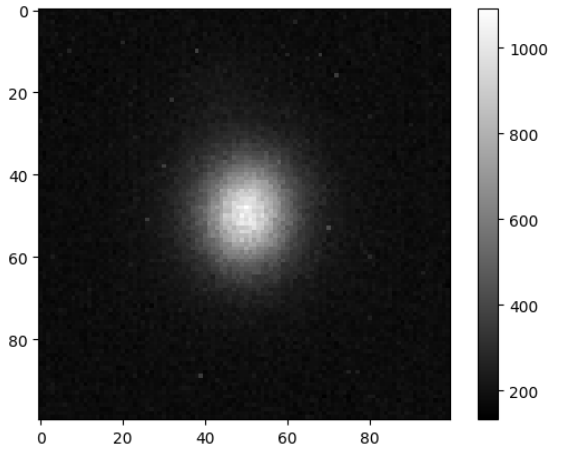


Star 3- Image

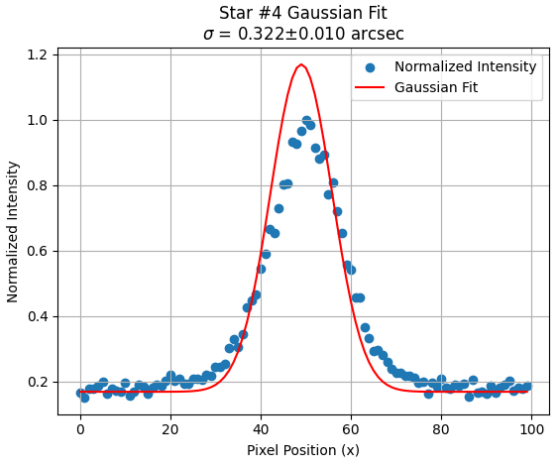


Star 3- Fit

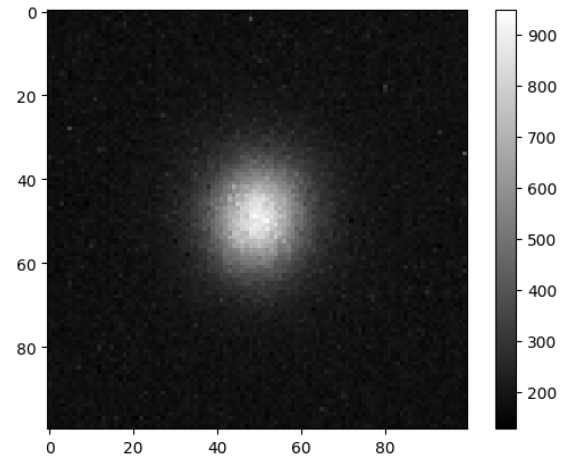
Figure 1: Star images and Gaussian fits (Stars 13).



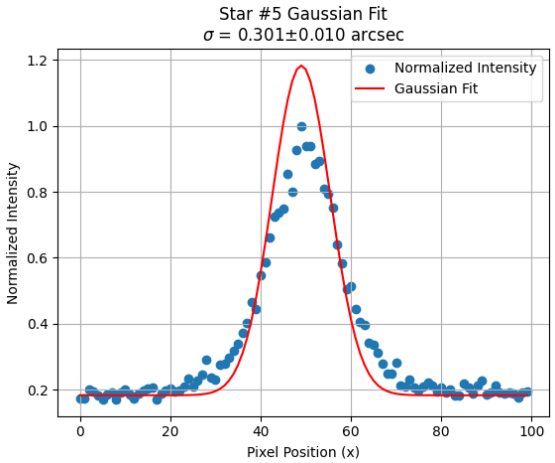
Star 4- Image



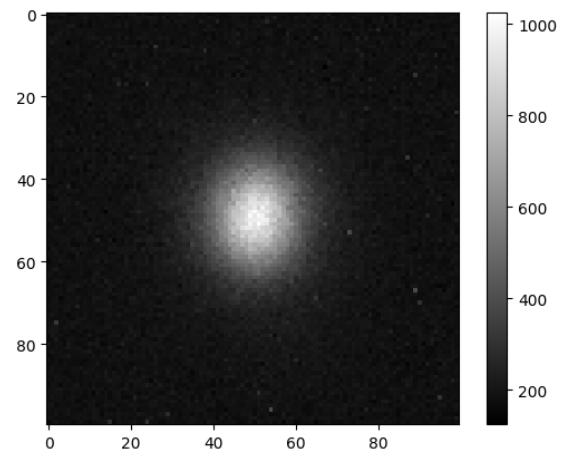
Star 4- Fit



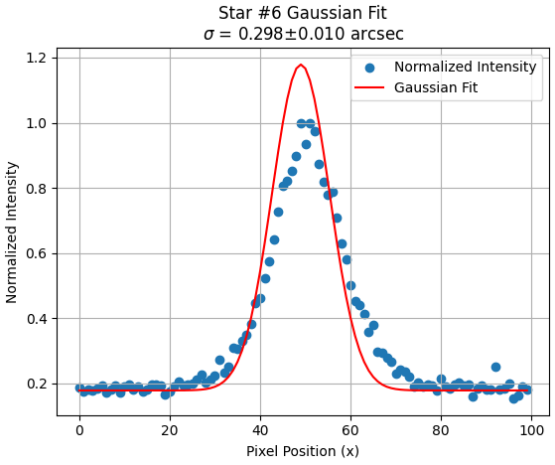
Star 5- Image



Star 5- Fit

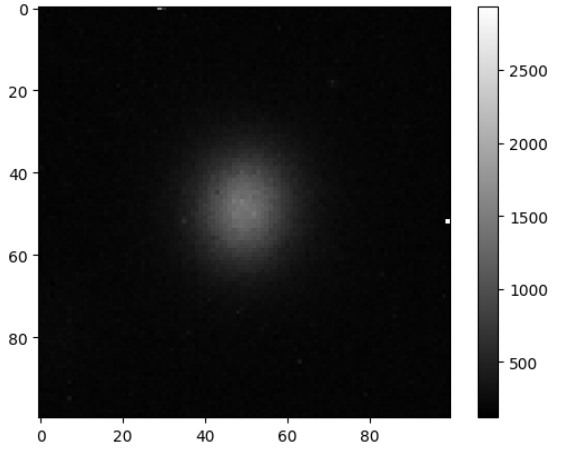


Star 6- Image

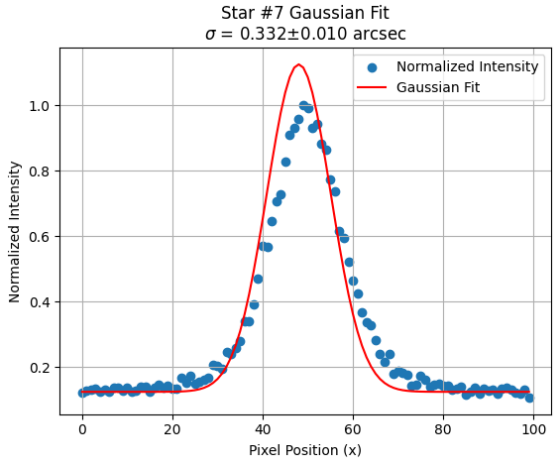


Star 6- Fit

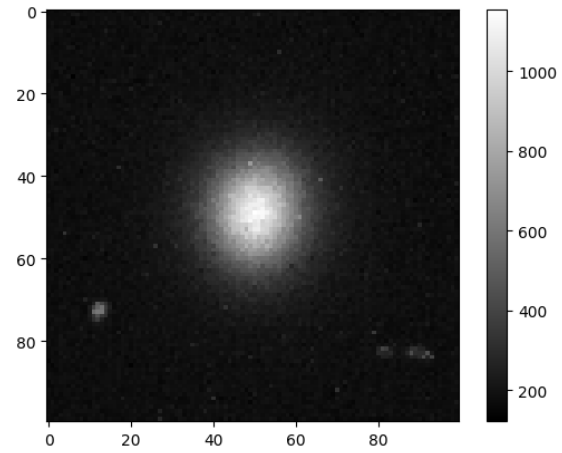
Figure 2: Star images and Gaussian fits (Stars 46).



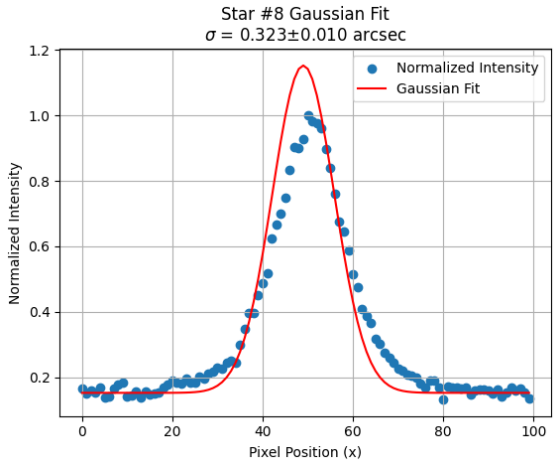
Star 7- Image



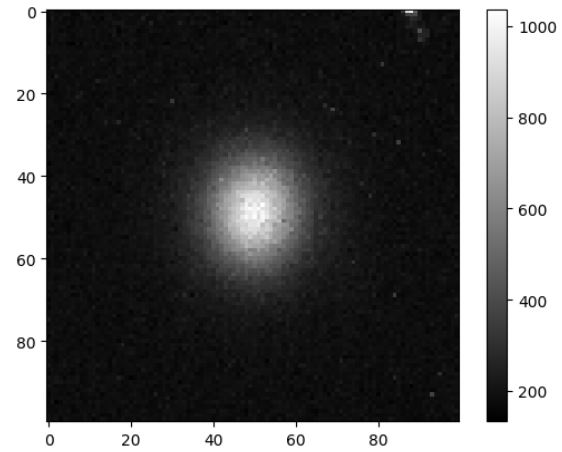
Star 7- Fit



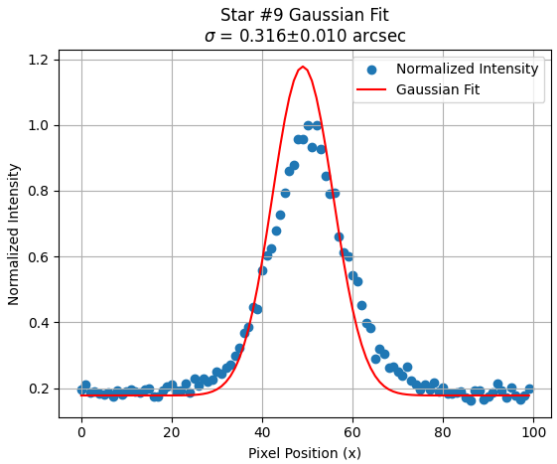
Star 8- Image



Star 8- Fit

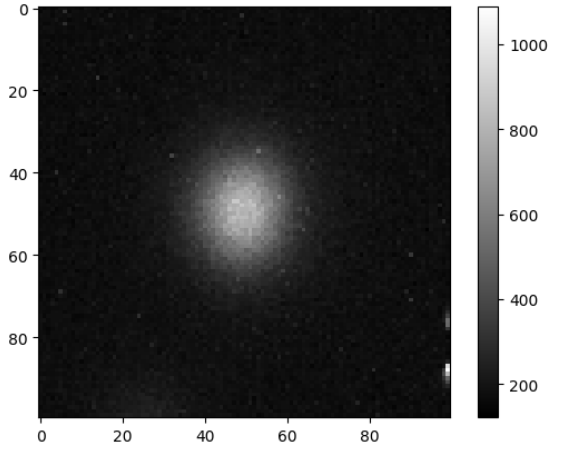


Star 9- Image

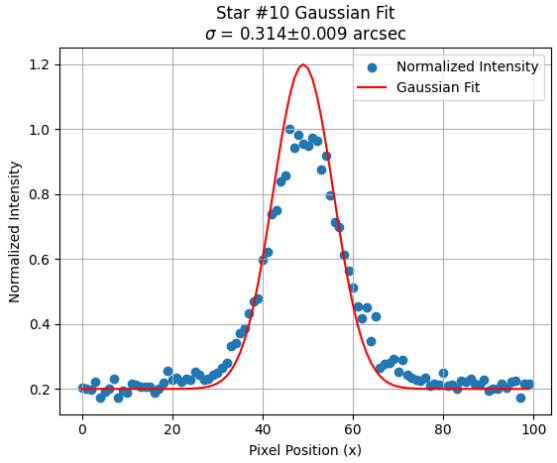


Star 9- Fit

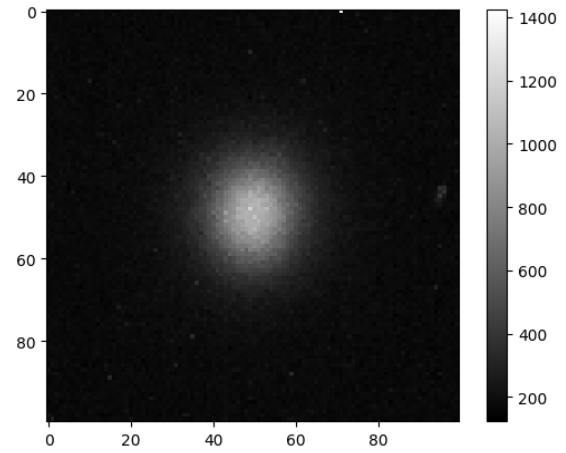
Figure 3: Star images and Gaussian fits (Stars 79).



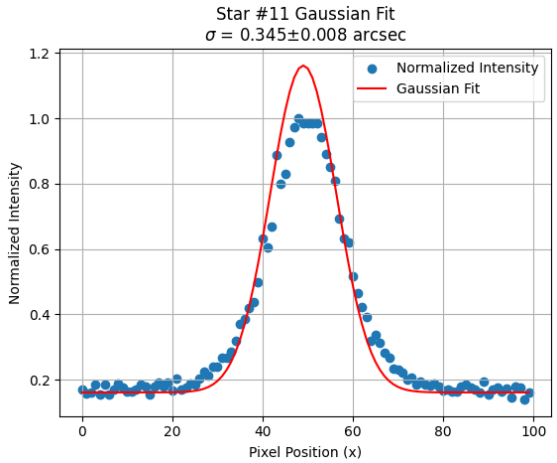
Star 10- Image



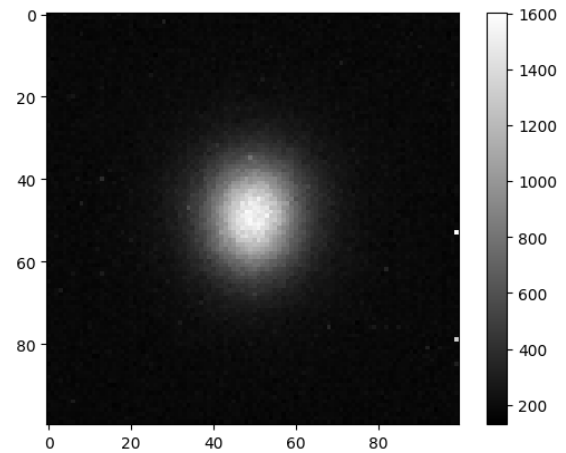
Star 10- Fit



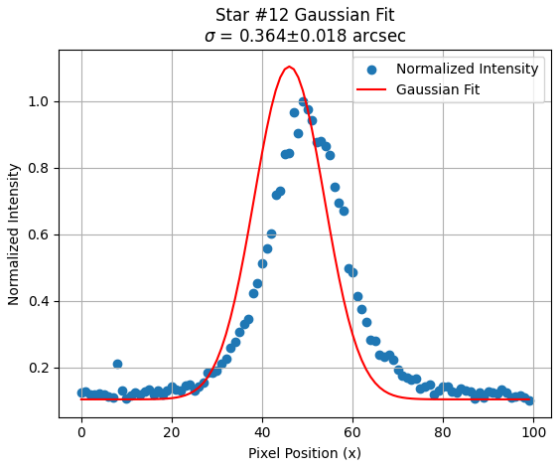
Star 11- Image



Star 11- Fit

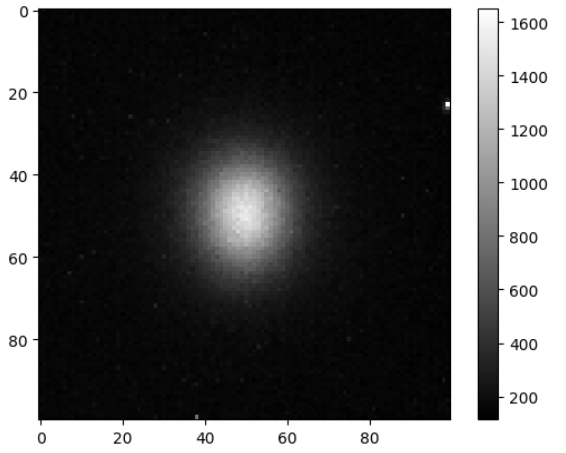


Star 12- Image

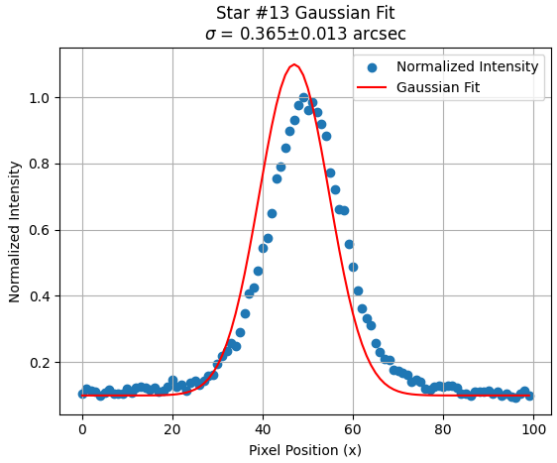


Star 12- Fit

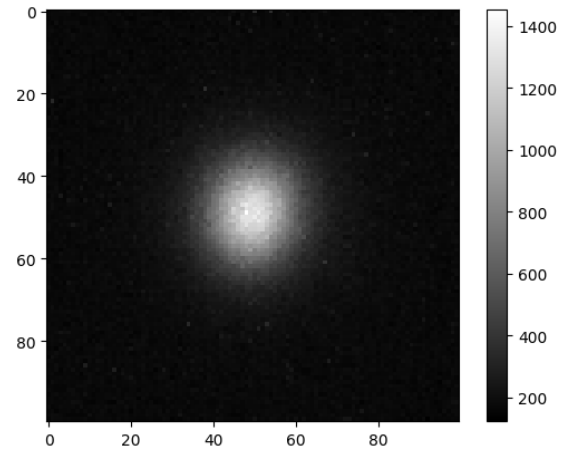
Figure 4: Star images and Gaussian fits (Stars 1012).



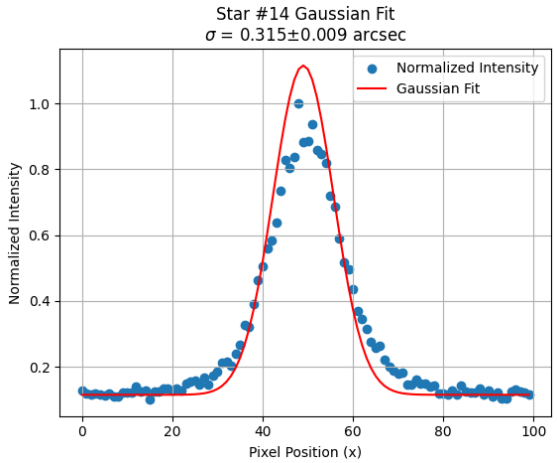
Star 13- Image



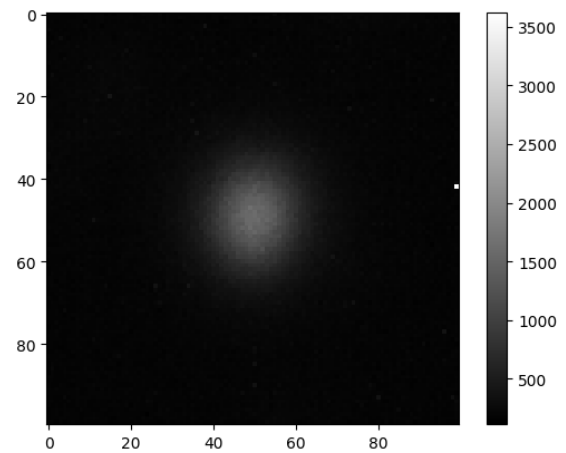
Star 13- Fit



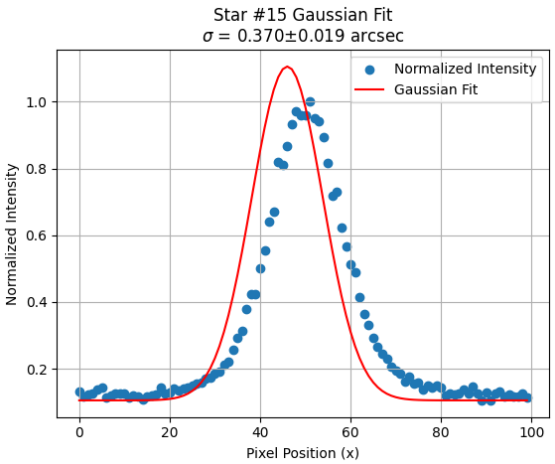
Star 14- Image



Star 14- Fit

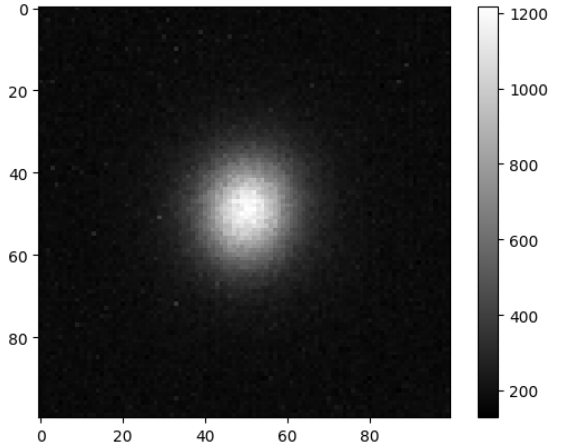


Star 15- Image

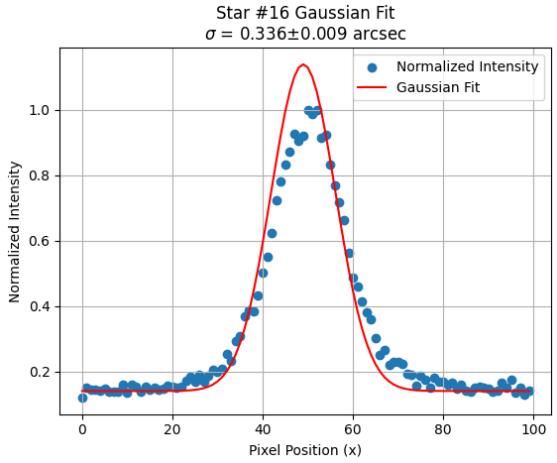


Star 15- Fit

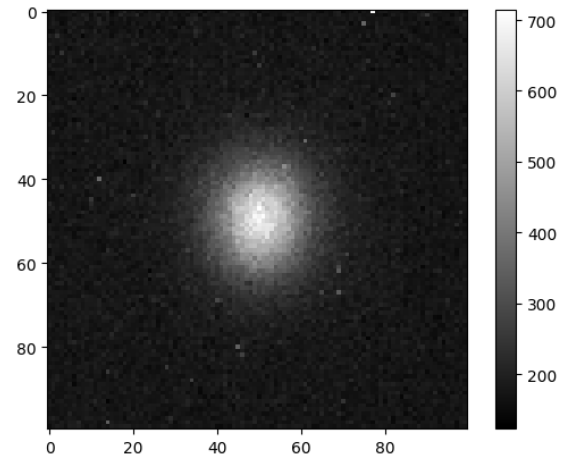
Figure 5: Star images and Gaussian fits (Stars 1315).



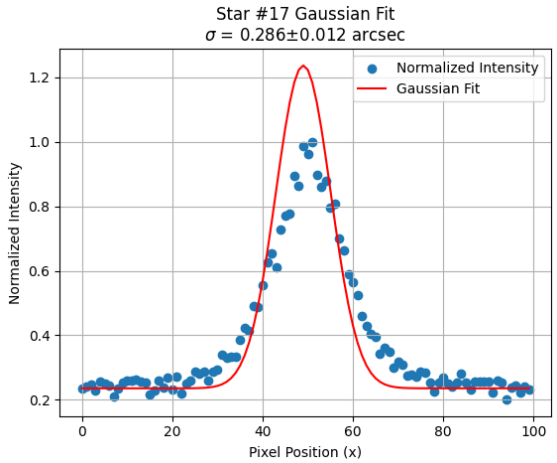
Star 16- Image



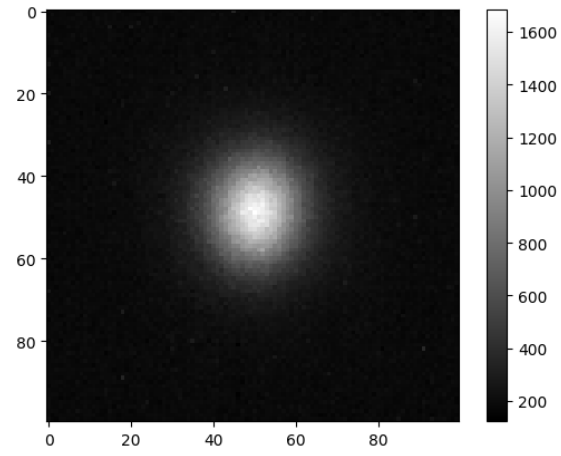
Star 16- Fit



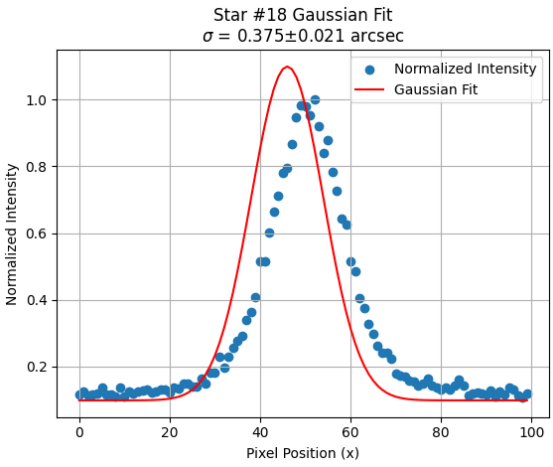
Star 17- Image



Star 17- Fit

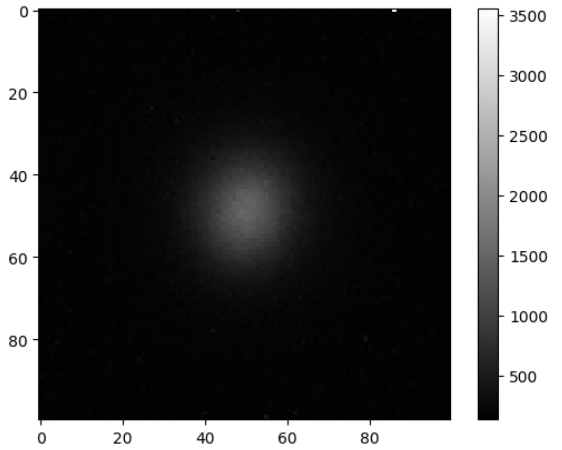


Star 18- Image

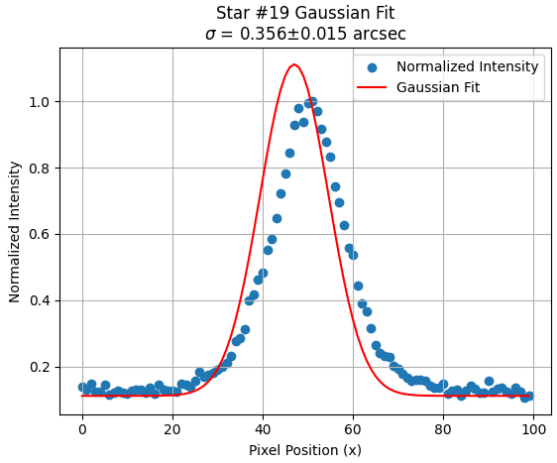


Star 18- Fit

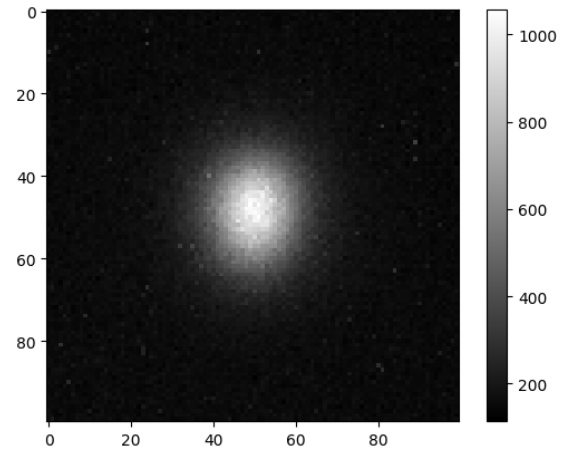
Figure 6: Star images and Gaussian fits (Stars 1618).



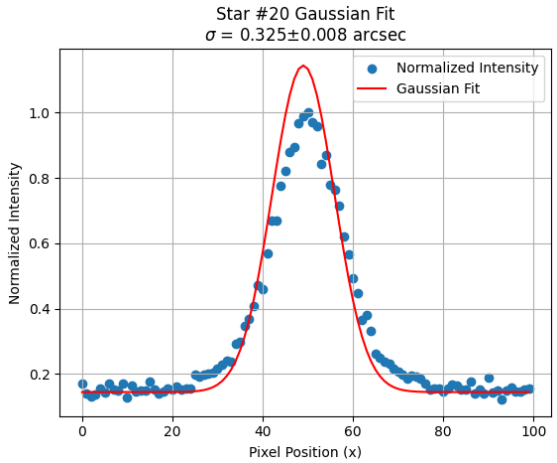
Star 19- Image



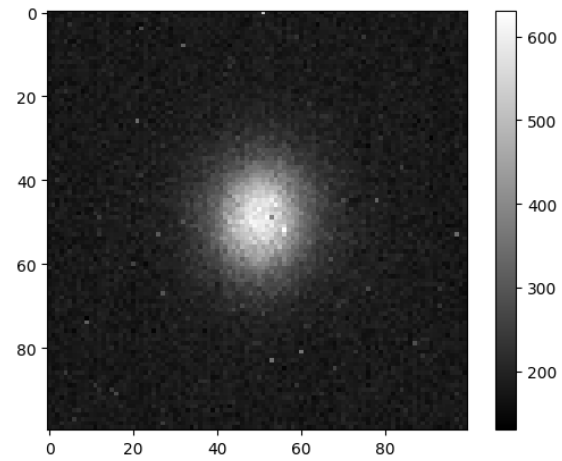
Star 19- Fit



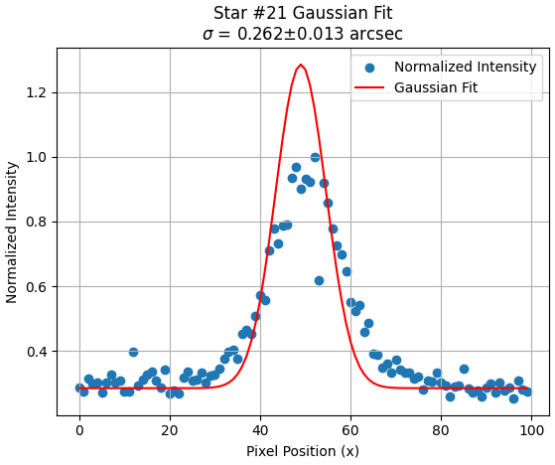
Star 20- Image



Star 20- Fit

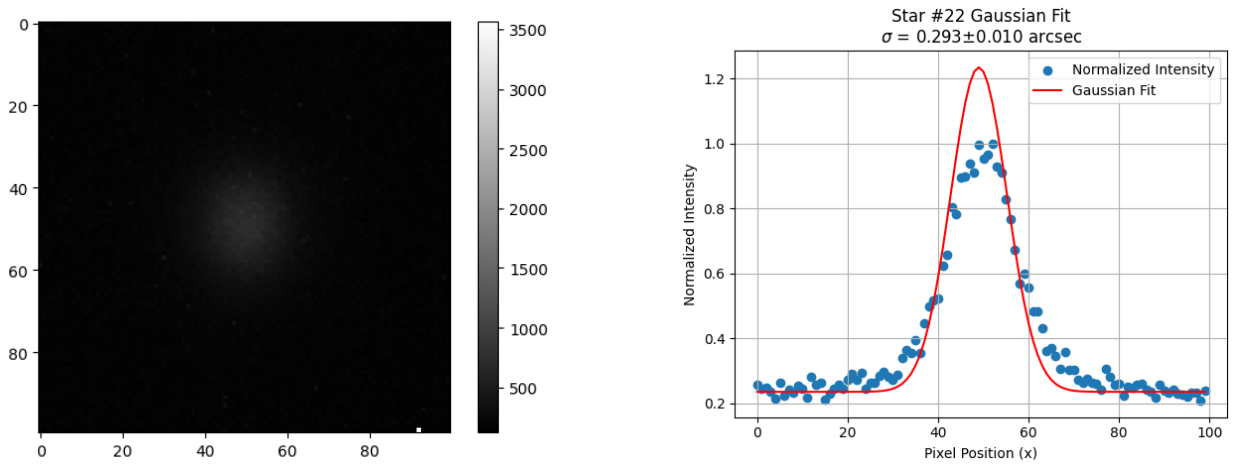


Star 21- Image



Star 21- Fit

Figure 7: Star images and Gaussian fits (Stars 1921).



Star 22- Image

Star 22- Fit

Figure 8: Star image and Gaussian fit (Star 22).

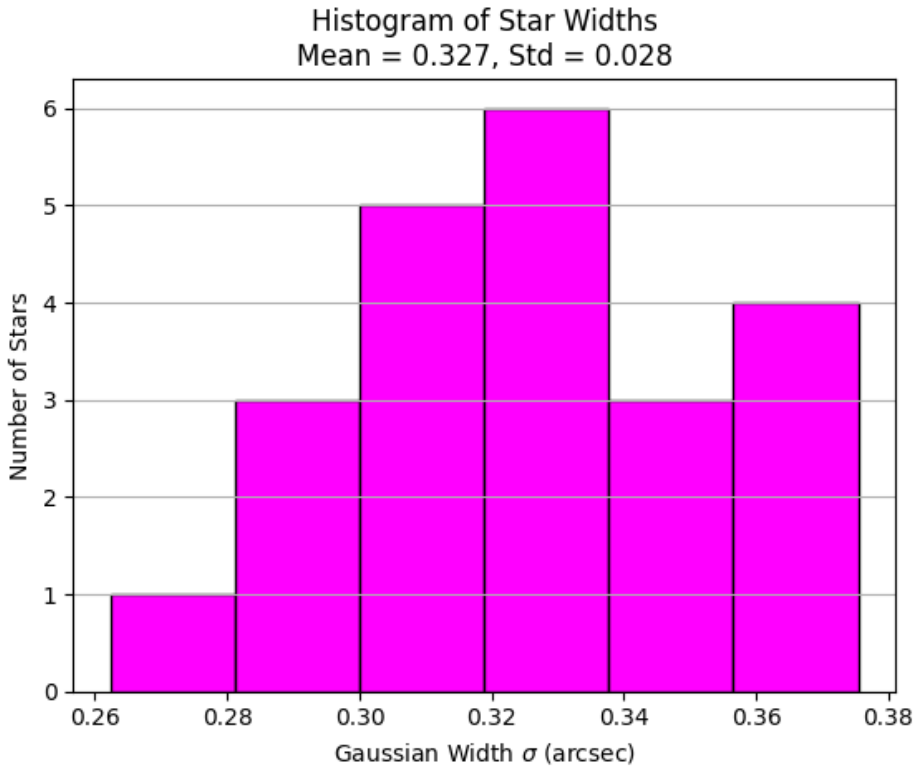


Figure 9: Star Width Histogram

The histogram doesn't look really good because we only had 22 stars, and selecting a huge number of stars by hand is not efficient.

Question

Does the average value have any relationship with exposure time?

Since all stars were measured from the same single image, we cannot directly assess a time-based trend. Therefore, no conclusive relationship between width and exposure time can be established from this dataset alone. But theoritically speakig, the only thing that changes is the amplitude and it shouldn't change our profile.