section4_1

February 1, 2024

1 Astronomy 443 - Lab 1

1.1 Purpose

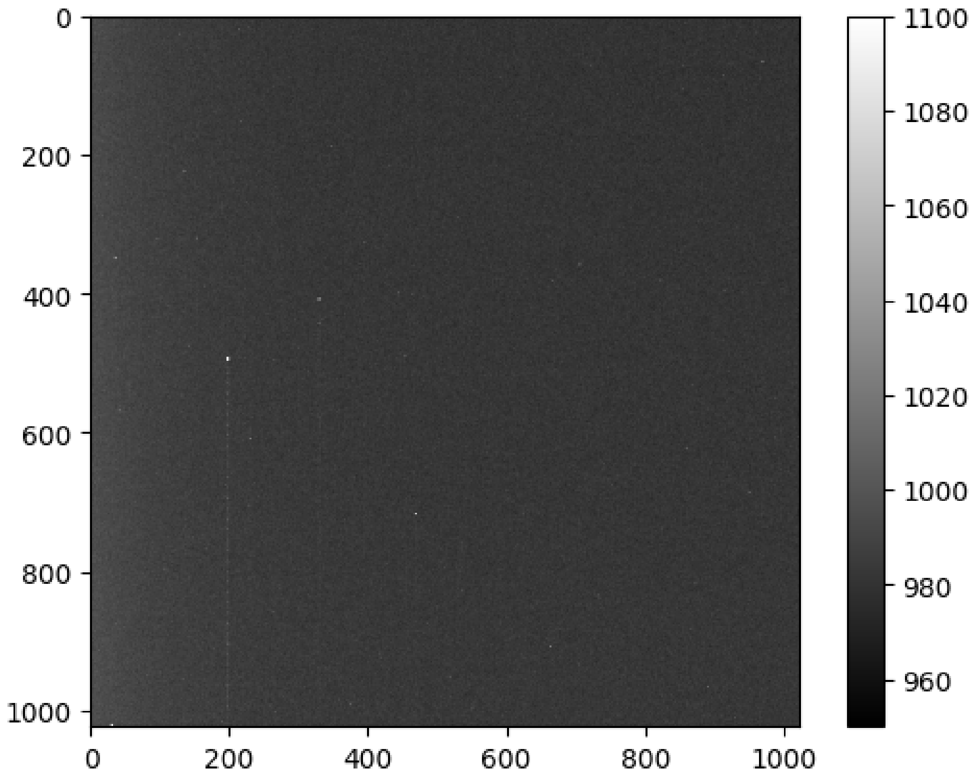
The objectives of this lab are to learn how to use the equipment needed to conduct observations during the semester. This equipment includes the STL-1001E, DADOS spectrographs, and the Mt. Stony Brook 14-inch telescope. Sets of calibration data for imaging and spectroscopic observations were taken, and strategies on analyzing said data were developed.

1.2 4.1 Bias Frames

A bias frame is an exopsure of 0 seconds, with the camera shutter closed, at the temperature of observation. They are used to determine the bias level of every pixel and weed out faulty ones (e.g. hot pixels). The below image is a bias frame at the temperature of observation, -5 Celsius.

[3]:
```
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

from astropy.io import fits

#open FITs file
list = fits.open("../images/lab1_bias.00000001.BIAS.FIT")

#get date from FITs file
image_data = list[0].data

plt.imshow(image_data,cmap='gray', vmin=950, vmax=1100)
plt.colorbar()
```

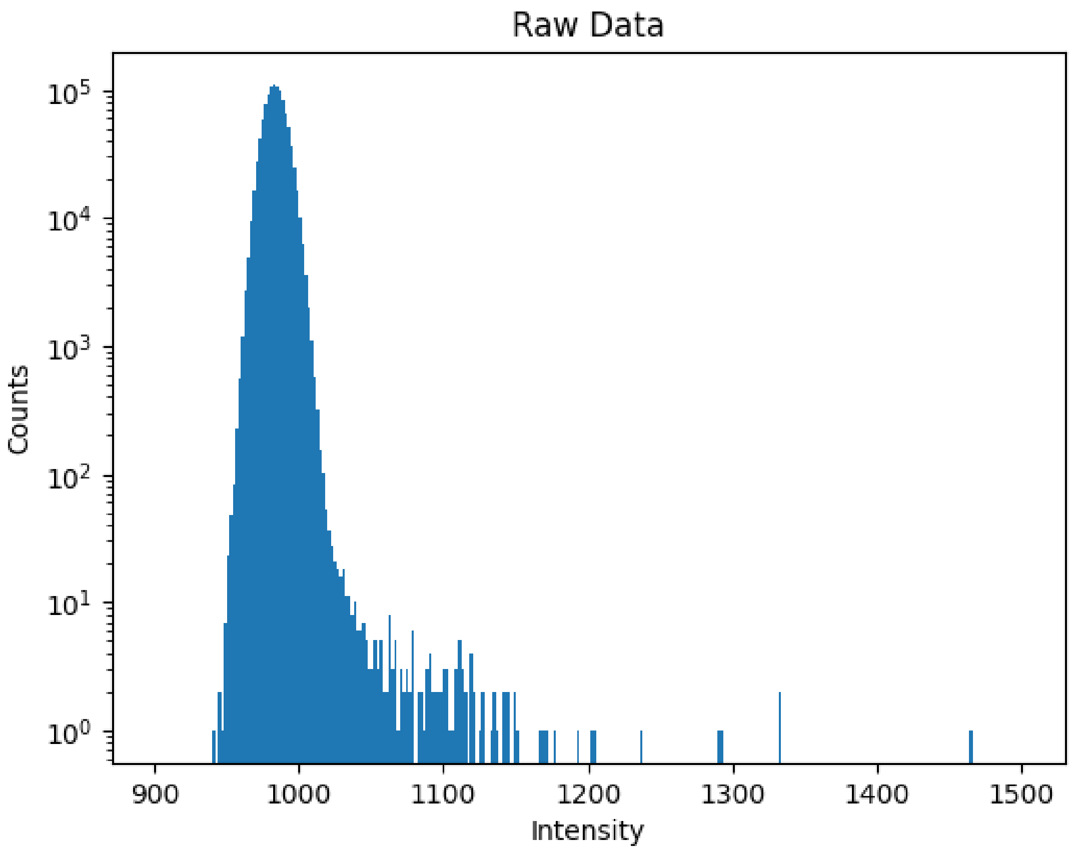
[3]:
```
<matplotlib.colorbar.Colorbar at 0x10df11a90>
```



Astropy's unique FITs file handling functions make it quick and easy to plot the count levels of the intensities of pixels.

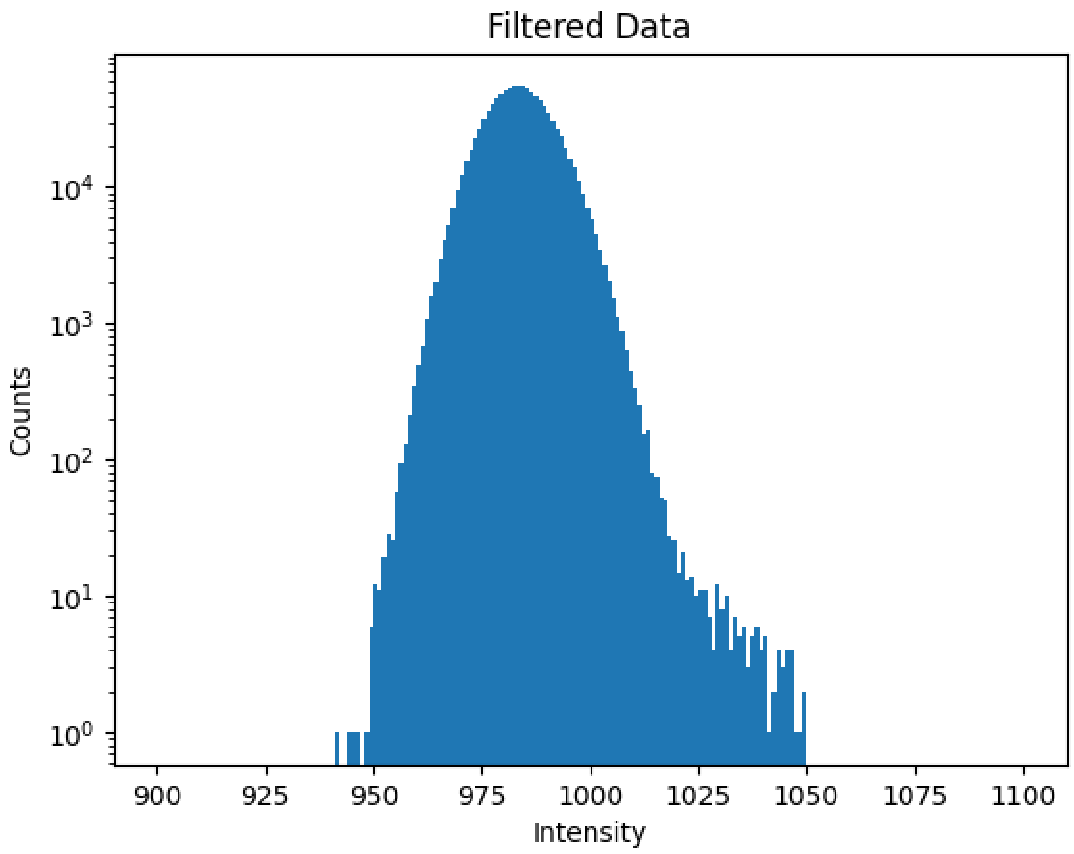
```
[4]: flat_data = image_data.flatten()
     histogram = plt.hist(flat_data, range=(900,1500),log=True, bins=300)
     plt.ylabel('Counts')
     plt.xlabel('Intensity')
     plt.title('Raw Data')
```

```
[4]: Text(0.5, 1.0, 'Raw Data')
```



It can be seen in the graph above that there are some pixels with a large dark current, causing them to become saturated. To filter the data, an intensity threshold is applied to the original data. Essentially, any pixels with a count greater than the threshold (1250 in this case) will be discarded from all of the data.

[35]:
```python
threshold = 1050

# Create a boolean mask indicating elements below the threshold
mask = flat_data < threshold

# Use the boolean mask to filter the data
filtered_data = flat_data[mask]

plt.hist(filtered_data, range=(900,1100),log=True, bins=200)

plt.title('Filtered Data')
plt.xlabel('Intensity')
plt.ylabel('Counts')
plt.show()
```



```
[36]: num_cut = len(flat_data) - len(filtered_data)
      frac_cut = (len(flat_data)-len(filtered_data))/len(flat_data) * 100

      print(f'number cut from original data: {num_cut}')
      print(f'fraction cut from original data: {frac_cut}')
```

```
number cut from original data: 133
fraction cut from original data: 0.012683868408203125
```

The graph of Filtered Data provides a more even distribution of counts, as there are no more erronious outliers. By subtracting the number of counts of the filtered data from the number of counts of the raw data, it is determined that there were 133 hot pixels (about 0.0127% of pixels).

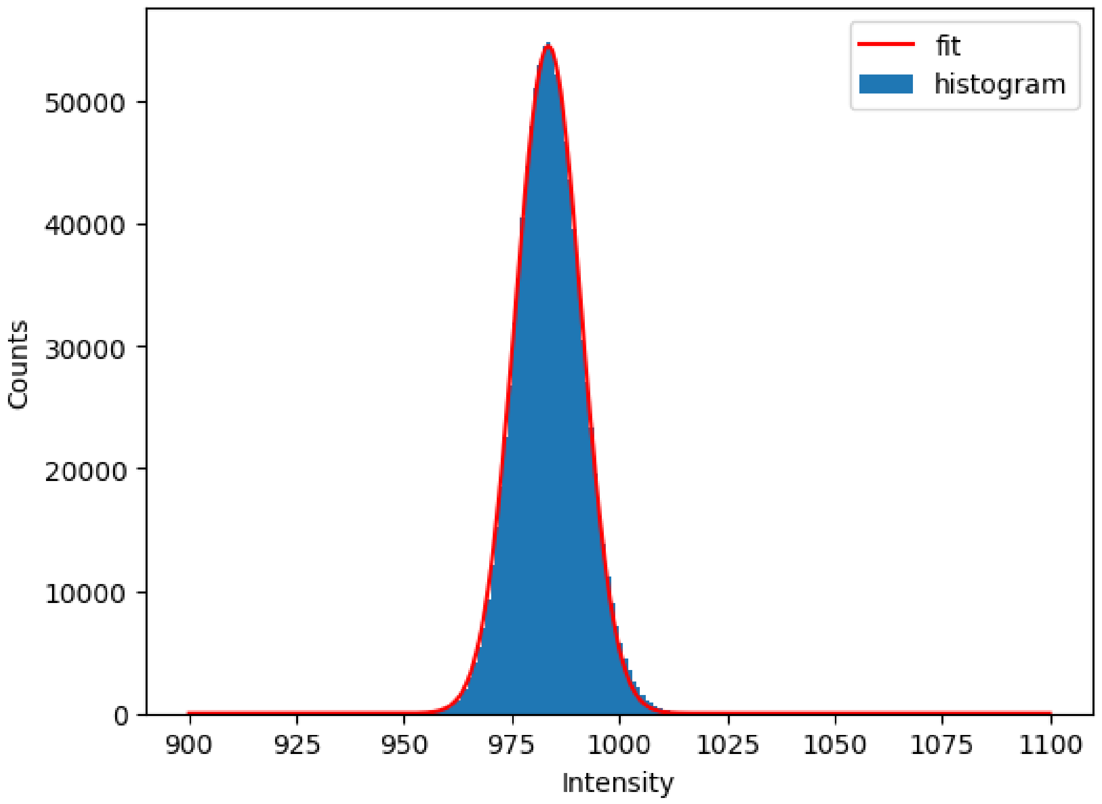
Using Numpy's mean and standard deviation functions, the following were calculated:

```
[37]: print('Mean:', np.mean(filtered_data))
      print('Stdev:', np.std(filtered_data))
```

```
Mean: 983.3167630476812
Stdev: 7.8315264971103025
```

Using the calculated mean and standard deviation values, Astropy was once again utalized to plot an overlaying Gaussian. Astropy contains a variety of models and fitting algorithms, of which Gaussian1D and LevMarLSQFitter are used here to produce the desired fit.

[42]:
```
from astropy.modeling import models, fitting

def plot_gaussian(data, r, mu, sigma, num_bins):

    #Get counts and bins
    bin_heights, bin_borders = np.histogram(data, range = r,bins = num_bins)
    bin_widths = np.diff(bin_borders)
    bin_centers = bin_borders[:-1] + bin_widths / 2

    #initialize Gaussian model
    t_init = models.Gaussian1D(mean=mu,stddev=sigma)
    #Select fitting algorithm to be used
    fit_t = fitting.LevMarLSQFitter()
    #Make best fit function
    t = fit_t(t_init, bin_centers, bin_heights)

    #Determine useful x-values for Gaussian fit
    x_interval_for_fit = np.linspace(bin_borders[0], bin_borders[-1], 10000)

    #Make plots
    plt.figure()
    plt.bar(bin_centers, bin_heights, width=bin_widths,␣
 ↪label='histogram',log=False)
    plt.plot(x_interval_for_fit, t(x_interval_for_fit), label='fit', c='red')
    plt.xlabel('Intensity')
    plt.ylabel('Counts')
    plt.legend()
    plt.show()

plot_gaussian(filtered_data,mu = 983.32,sigma = 7.83, r =␣
 ↪(900,1100),num_bins=200)
```



The gain, number of electrons, and number of counts are related by:

$$gain = \frac{N_{electrons}}{N_{counts}}$$

Using the spec sheet for the STL1001E, the actual gain is 2 and the number of electrons is 14.8e RMS. With a read noise equal to the standard deviation of counts, 7.83, the number of electrons was found to be:

$$2 \times 7.83 = 15.66e^{-}$$

While this is approximately one electron higher than the spec sheet, it is also relatively accurate.

```
[ ]:
```