

LAB 1: PHOTON COUNTING & DETECTORS

JESSICA BIRKY, JULIAN BEAZ-GONZALEZ, RUSSELL VAN-LINGE

ABSTRACT

In this lab, we examine the statistics of photon counting, and the noise properties of a charged couple device (CCD) using 2048x2048 pixel optical images taken on the Nickel 1 meter direct imaging camera. In particular we characterize the effect of several types of noise: first the inherent noise of photon counting by comparing to the Poisson distribution, second the read noise, and lastly we comment the effect of different systematics such as instrument defects or dark current due to temperature variations. In total we collect XX frames integrated at varying exposure times ranging from 0 to 768 seconds. Comparing the slope between mean and variance of each bias-subtracted, combined frame of each exposure, we determine the gain of the detector to be XX ADU/s. Normalizing our frames (dividing the counts by exposure time) we find that as we add more exposures, the mean of the mean (MOM) detector counts per second converges to XX ADU/s and the standard deviation of the mean (SDOM) decreases to XX ADU/s with more frames added.

1. INTRODUCTION

In order to understand the. Advancements in technology in the 20th century has now made it possible to observe electromagnetic radiation from the scales of $10^{-12} - 10^3\text{m}$ (as short as gamma rays to as long as radio waves). Understanding how detectors work, the statistics of light collection, as well as the limitations of the instrument and sources of contaminations are a fundamental first steps before performing any scientific analysis of light with an instrument.

CCDs, the photoelectric effect

Types of noise we want to quantify

2. OBSERVATIONS

Nickel telescope, what are bias and flat frames (Table 1)

Start File #	End File #	# Frames	Type	Exposure Time (sec)
401	404	3	Bias	0
419	422	3	Flat	3
423	425	3	Flat	6
426	428	3	Flat	12
429	431	3	Flat	24
432	434	3	Flat	48
435	437	3	Flat	96
438	439	3	Flat	192
440	–	1	Flat	384
441	–	1	Flat	768

Table 1. Observation log, recording the files associated with each exposure time. Files 405 – 418 were removed due to inconsistent count numbers.

3. DATA REDUCTION & METHODS

3.1. *Theoretical Distribution of Light*

Before we can characterize our observations, it is important to first determine what we expect our data will look like. That is, we are measuring the number of photons hitting a detector over a fixed amount of time, what do we predict will be the number of photons that will hit each pixel in that time interval? Let's assume that on average μ photons will hit a pixel over fixed time interval t . Now consider dividing that time interval into n increments: the probability that a n photons arrive to the detector is approximately Binomial. Now in the limit that $n \rightarrow \infty$, we arrive at the Poisson distribution:

$$P(x; \mu) = \frac{\mu^x}{x!} e^{-\mu} \quad (1)$$

which we interpret as the probability that x photons will arrive to a detector pixel in time interval t , given that μ photons arrive in that interval on average. The expected mean and variation for this distribution being $\langle x \rangle = \mu$ and $\langle \sigma^2 \rangle = \mu$. Under the conditions that the mean of the data is much smaller than the number of observations ($\mu \ll n$), and

In practice, the Poisson distribution is unstable to compute for large values of x because the term $x!$ becomes too large for the memory of the computer to handle. In order to try to reduce the memory load for large values of x , we approximate the $x!$ term using the Sterling approximation:

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \quad (2)$$

the code for the Poisson distribution is implemented in lines 7 – 9 of Appendix 8.1.

However when the conditions for the Poisson distribution are not met very well, the Gaussian distribution may instead be a good approximation:

$$P(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3)$$

where the input parameters are the mean and standard deviation of the data (μ and σ). The code for the Gaussian distribution is implemented in lines 11 – 13 of Appendix 8.1.

3.2. Data Reduction

For each exposure observed, the data output from the Nickel instrument is stored in a fits file, which contains the two-dimensional array of detector counts at each pixel, as well as other relevant variables about the detection (such as the detector temperature, time of observation, exposure time, etc.). The first step of reduction consists of combining the bias frames, and our procedure for combining frames (lines 19 – 26, 8.2) is to take the median count value at each pixel. Second, for each exposure, we read the individual files from a directory, subtract the combined bias, then combine those into a single frame (lines 9 – 11, 8.4).

As discussed in further the results (4.1)

3.3. Types of Detector Noise

How to calculate the readnoise, other noise, error propagation

$$\sigma_{ADU}^2 = \frac{ge}{C} (ADU - ADU_0) + \sigma_0^2 \quad (4)$$

Thus the gain (ge/C) is the linear slope between the mean and variance of the ADU counts for each bias-subtracted, combined exposure frame. To numerically determine the gain from our data we fit a second-order polynomial to our mean vs. variance plot (lines 1 – 7, 8.5), and the gain is then the first order term (coefficient of x).

4. DATA ANALYSIS & MODELING

4.1. Diagnostic Plots: Combined Images & Histograms

Figure 1 shows the combined bias frames, and the figures in 4 and 5 show the bias-subtracted, combined histograms and images for each of the eight different exposure times (3, 6, 12, 24, 48, 96, 192, 384, and 768 seconds), using the procedures described in 3.2. Lines 25 – 33 of 8.3 plots the histogram, and lines 72 – 75 plot the image.

The left side of each panel shows the histogram of the number of detector counts in analog to digital units (ADU), normalized such that the sum of all of the bins is one (plotted in black). The solid blue vertical line marks the mean of the data, and the dashed blue line marks the median, indicating which distributions are symmetric or skewed. For all histograms however, the mean and median are very close, meaning the distributions are quite symmetric. The two shades of green mark ± 1 and ± 2 standard deviations above and below the mean. Finally, the distribution in red shows the expected Poisson distribution, as a function of the mean of the data (\bar{x}), with a one standard deviation range also shaded in red (where the expected standard deviation for the Poisson is $\sigma = \sqrt{\bar{x}}$). The distribution in cyan shows the Gaussian fit using the mean and standard deviation of the data (\bar{x}, s).

The right side of each panel shows the 2D images. In order to visualize the detector counts per pixel, we must map the number of counts to an RGB color value, which we use the `matplotlib` ‘gray’ colormap for. To emphasize the features in the data, we choose the min and max values of the colormap to be the 10th and 90th percentile counts of the data.

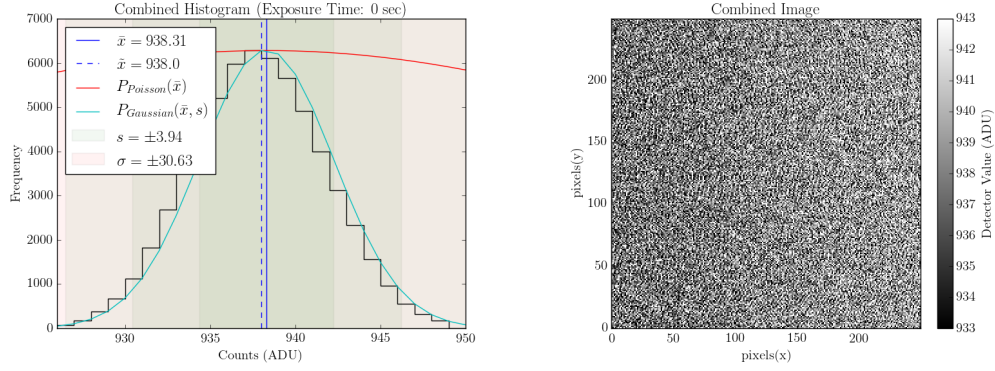


Figure 1. Combined bias frames.

4.2. Mean of the Mean & Standard Deviation of the Mean

Describe computation, show plots

4.3. Quantifying Detector Noise

Figure 2 is the mean vs. variance plot for each bias-subtracted, combined exposure. As described in 3.3 we fit a second-order polynomial using numpy to the point on the plot. The fit is mostly described by the first-order gain term which is XX , with a very small second-order correction of XX .

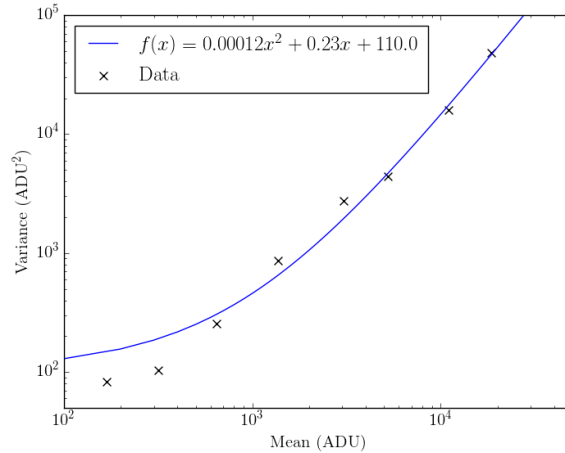


Figure 2. Mean vs. variance for all bias-subtracted exposures, fitted to a second-order polynomial.

5. DISCUSSION

5.1. Poisson & Gaussian Comparisons

Looking at Figures 4 and 5, we find that the detector count distribution is close to a Poisson distribution for low exposure times, but for larger exposure times it increasingly diverges with a much larger standard deviations than expected (as summarized by Figure 2).

We also see that low exposure times (3, 6, and 12 sec) have a single peak in the distribution around the mean, however at 48 sec the distribution becomes bimodal, and for high exposure times (96, 192,

384, and 768 sec) the distribution develops three separate peaks. Comparing these histograms to the corresponding images at the right, we can see the regions on the detector which correspond to the different peaks: the dark edges make up roughly the bottom third of the distribution, and the bright center making up the higher third of the distribution.

5.2. Sources of Noise & Systematic Effects

Variations in the quantum efficiency of each pixel. Dust particles on the lens scatter and absorb light unevenly at different points on the detector.

Checked for systematics due to temperature variation that could maybe be responsible for dark current: found temperature to vary only within 4C (Figure 3). Checked for systematic differences in different datasets.

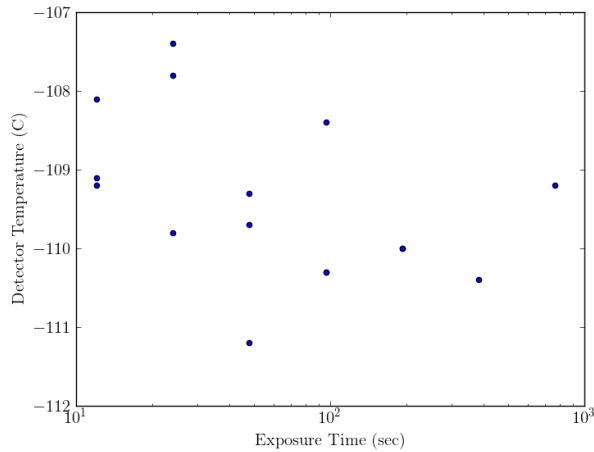


Figure 3. Temperature of the CCD detector for different exposure frames does not vary more than about 4C.

6. CONCLUSION

7. AUTHOR CONTRIBUTIONS

This project was done in collaboration with Julian Beas-Gonzalez and Russell Van-Linge (Group E), using data collected by Group B. Russell wrote code for fitting Poisson and Gaussian distributions and computing MOM/SDOM, Julian wrote code for histograms and computing MOM/SDOM, and I wrote code for combining frames/bias subtracting images, plotting histograms/images and slicing images (except for a few snippets for making histograms and reading fits files from the class notes). We met several times outside of class to compare results, fix issues, and combine our code together, which can be found on github: <https://github.com/jbirky/photonCounting>.

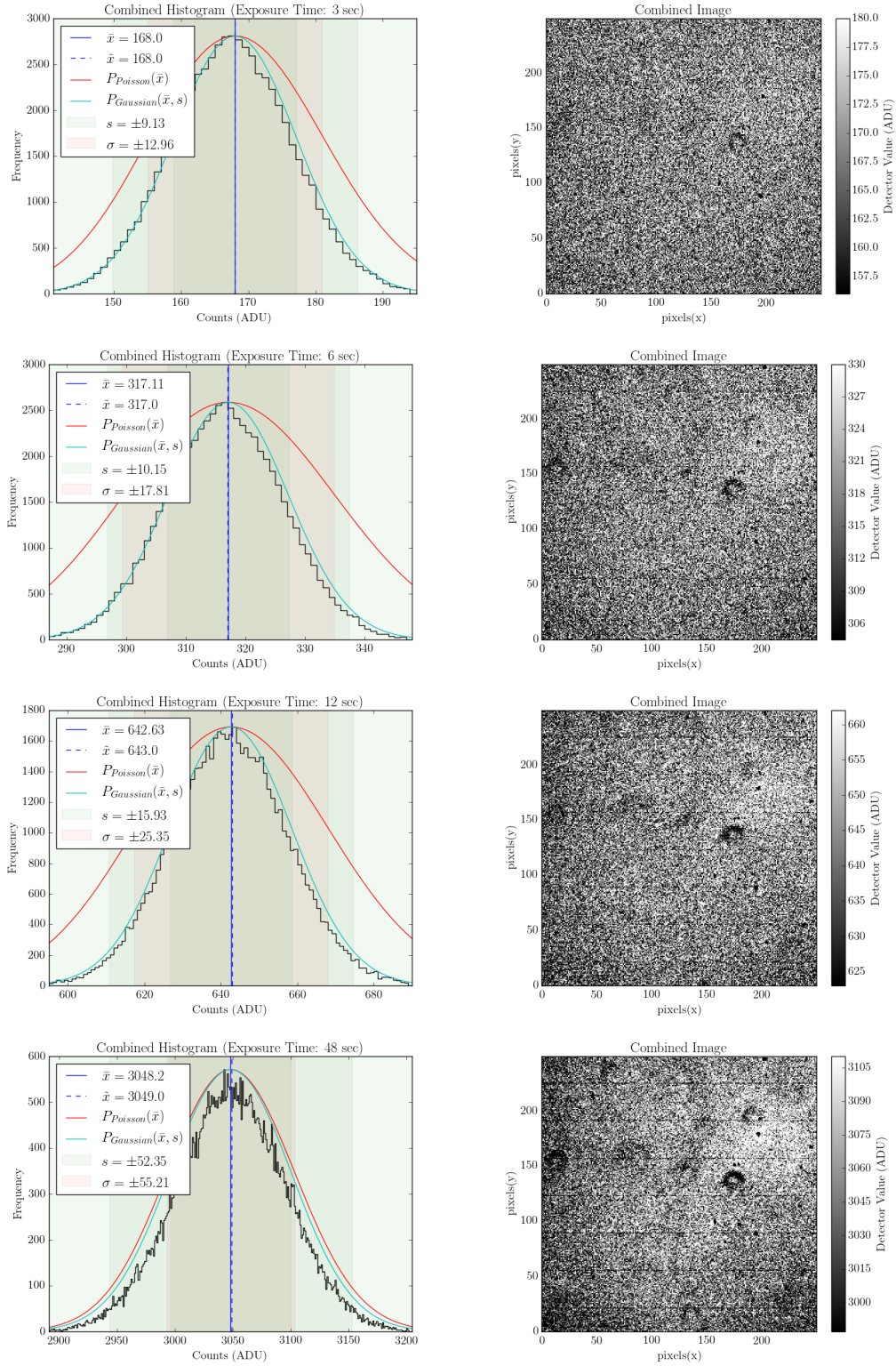


Figure 4. Combined, bias-subtracted, flat frame histograms and images for varying exposure times (0, 3, 6, and 12 sec).

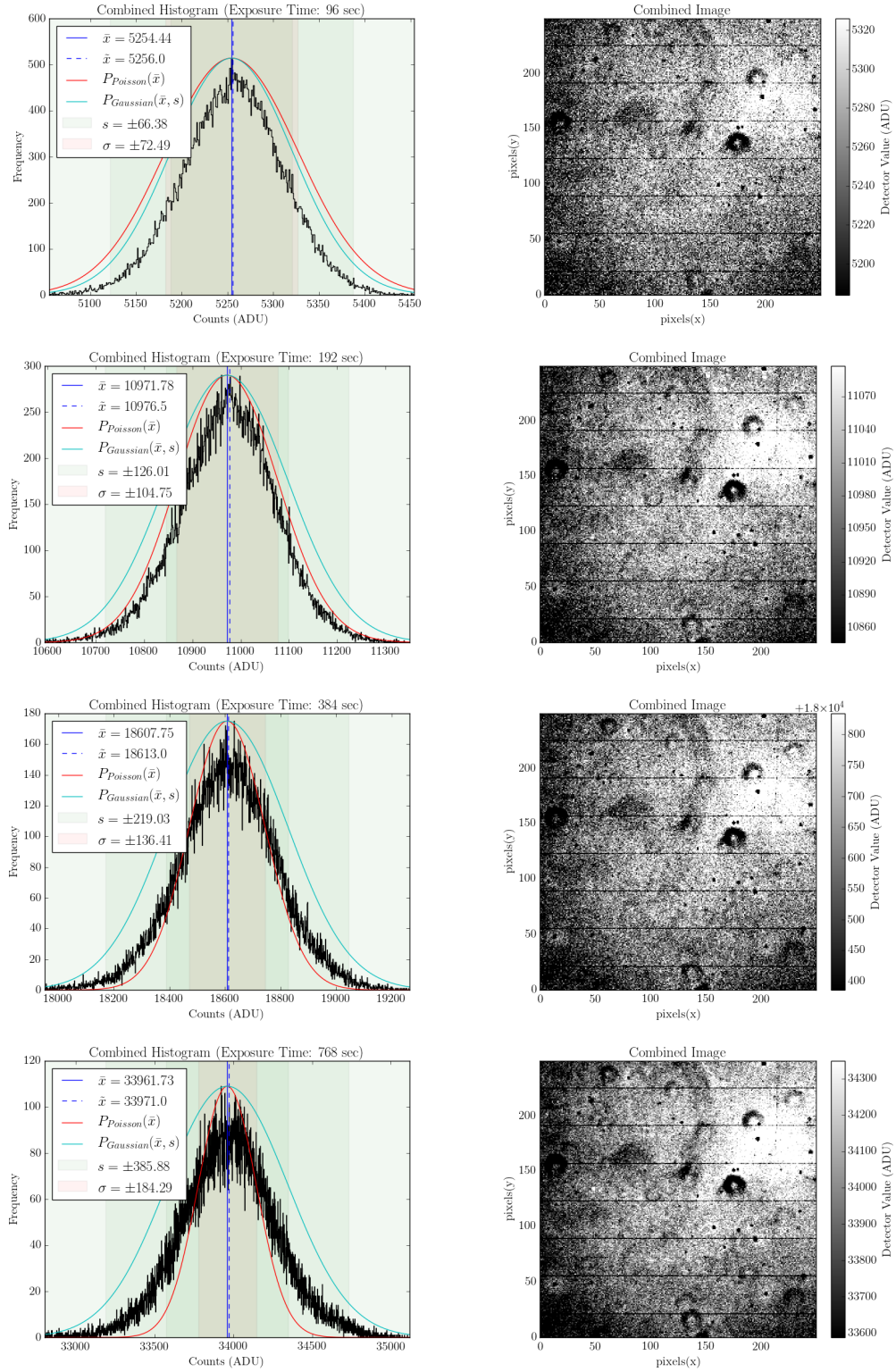


Figure 5. Combined, bias-subtracted, flat frame histograms and images for varying exposure times (96, 192, 384, and 768 sec).

8. APPENDIX

8.1. *Statistical Code*

```

1 def mean(array):
2     return np.sum(array)/len(array)
3
4 def stdev(array):
5     return np.sqrt(sum((array - mean(array))**2)/(len(array) - 1))
6
7 def poisson_approx(data, mean):
8     pdist = np.array(np.exp((data*np.log(mean)) - (data*np.log(data)) + data - mean))
9     return pdist
10
11 def gaussian(data, mean, sigma):
12     gdist = np.array((1/(sigma*(2*math.pi)**(.5)))*np.exp(-.5*((data-mean)/sigma)**2))
13     return gdist

```

8.2. *Data Processing code*

```

1 def readData(folder):
2     """
3     Read all frames from a given directory into one matrix.
4     input:  directory to folder containing frames
5     output: array3d (dimension: xpixel x ypixel x # frames)
6            array2D (dimension: 1D flattened img x # frames)
7     """
8
9     files = os.listdir(folder)
10
11     array3D, array2D = [], []
12     for ff in files:
13         arr = fits.getdata(folder + ff)
14         array3D.append(arr)
15         array2D.append(arr.flatten())
16
17     return np.array(array3D)
18
19 def combineFrame(data_array):
20     """
21     Combine image frames and return 1D array.
22     input:  data array (dimension: # 1D detector counts x # frames)
23     output: 1D combined array of detector counts (ADU)
24     """
25
26     return np.median(data_array, axis=0)

```

8.3. *Histogram/Image code*

```

1 def plotAll(array2D, **kwargs):
2
3     arr = array2D.flatten()
4
5     avg = mean(arr)
6     std = stdev(arr)
7     med = np.median(arr)

```



```

8     Npix = len(arr)
9
10    sigma = kwargs.get('sigma', 2)
11    low = int(np.round((avg-sigma*std)))
12    high = int(np.round((avg+sigma*std)))
13    rng = kwargs.get('rng', [low, high])
14    exp = kwargs.get('exp')
15    if 'nbins' in kwargs:
16        nbins = kwargs.get('nbins')
17        bin_size = (rng[1]-rng[0])/nbins
18    else:
19        bin_size = kwargs.get('bin_size', 1)
20
21    fig, (ax1, ax2) = plt.subplots(1,2, figsize=[18,6])
22
23    # Histogram
24    #=====
25    hr = np.arange(rng[0], rng[1]+1, bin_size)
26    hist = []
27    for i in range(len(hr)):
28        try:
29            counts = len(np.where((arr >= hr[i]) & (arr < hr[i+1]))[0])
30        except:
31            counts = 0
32        hist.append(counts)
33    ax1.step(hr, hist, color='k')
34
35    #mean and median lines
36    ax1.axvline(avg, color='b', label=r'$\bar{x}=%s$'%(np.round(avg,2)))
37
38    #sigma levels
39    if kwargs.get('show_level', True) == True:
40        for i in np.arange(1,sigma+1):
41            if i == 1:
42                ax1.axvspan(avg-i*std, avg+i*std, facecolor='g', alpha=0.05, \
43                    label=r'$s=\pm%s$'%(np.round(std,2)))
44            else:
45                ax1.axvspan(avg-i*std, avg+i*std, facecolor='g', alpha=0.05)
46
47
48    #poisson distribution
49    xarray = np.arange(rng[0]-10, rng[1]+10, 1)
50    pdist = poisson_approx(xarray, avg)
51    pdist = max(hist)/max(pdist)*pdist
52    ax1.plot(xarray, pdist, color='r', label=r'$P_{Poisson}(\bar{x})$')
53    std_expected = math.sqrt(avg)
54    ax1.axvspan(avg - std_expected, avg + std_expected, facecolor='r', alpha=0.05, \
55        label=r'$\sigma=\pm%s$'%(np.round(std_expected,2)))
56
57    #gaussian distribution
58    gdist = gaussian(xarray, avg, std)
59    gdist = max(hist)/max(gdist)*gdist
60    ax1.plot(xarray, gdist, color='c', label=r'$P_{Gaussian}(\bar{x},s)$')
61

```

```

62 ax1.legend(loc='upper_left')
63 ax1.set_xlabel('Counts (ADU)')
64 ax1.set_ylabel('Frequency')
65
66 if 'exp' in kwargs:
67     ax1.set_title('Combined_Histogram_(Exposure_Time:_%s_sec)'%(exp))
68 ax1.set_xlim(rng)
69
70 # Image
71 #=====
72 hrng = kwargs.get('hrng', [np.percentile(arr, 10), np.percentile(arr, 90)])
73 pl = ax2.imshow(array2D, origin='lower', interpolation='nearest', \
74     cmap='gray', vmin=hrng[0], vmax=hrng[1])
75 fig.colorbar(pl, ax=ax2, fraction=0.046, \
76     pad=0.04).set_label('Detector_Value_(ADU)')
77
78 ax2.set_xlabel('pixels(x)')
79 ax2.set_ylabel('pixels(y)')
80 ax2.set_title('Combined_Image')
81
82 if 'save_dir' in kwargs:
83     save_dir = kwargs.get('save_dir')
84     plt.savefig(save_dir + 'exposure%s.png'%(exp))
85 plt.show()

```

8.4. Data Reduction

```

1 files = natsorted(os.listdir('data/'))[1:]
2 exp_times = np.array([int(f.split('exp')[1]) for f in files])
3
4 #combined frames, and combined frames in counts per second
5 counts, counts_ps = [], []
6 for exp in exp_times:
7
8     #Read frames, subtract combined bias, combine frames
9     array3D = readData('data/exp%s/'%(exp))
10    bias_sub_3D = [(frame - bias_2D) for frame in array3D]
11    array2D = combineFrame(bias_sub_3D)
12
13    array2D_cps = array2D/exp
14
15    #cut out safe part of image
16    cut = Cutout2D(array2D, window[0], window[1]).data
17    cut_cps = Cutout2D(array2D_cps, window[0], window[1]).data
18
19    counts.append(cut.flatten())
20    counts_ps.append(cut_cps.flatten())
21
22    #Plot histogram and image & save to .png
23    plotAll(cut, sigma=3, exp=exp, show_level=True, save_dir='writeup/plots/')

```

8.5. Gain Calculation

```

1 xbar = np.mean(counts, axis=1)
2 s = np.std(counts, axis=1)

```

```
3
4 #fit polynomial
5 x_arr = np.arange(0, 10**5, 10**2)
6 fit = np.polyfit(xbar, s**2, 2)
7 y_arr = np.polyval(fit, x_arr)
```
