Basler ac1640 – 750um Gain and Read Noise

Chris Bohlman

10/23/2017

**Abstract**:

The purpose of this experiment was to determine how data trends changed with different exposure times. As the Basler camera's use may be greatly expanded in the near future, we wanted to examine whether the cameras were a good fit for our goals. To begin with, a closer look at the Pylon5 SDK (that came with the camera) and example programs was necessary. From there, we were able to write a program that obtained an image at a certain exposure time and save the result to a fits file. We then changed the program in order to take 10 images at different exposure times, and then wrote programs to collect the 2 data sets necessary. The first data set, taken on September 12, 2017, consisted of 1000 images, with 100 images taken at each exposure time of {600, 1175, 1750, 2325, 2900, 3475, 4050, 4625, 5200, 5775} microseconds. This shall henceforth be refereed to as the 'earlier' data set, or the first data set. The second data set, taken on October 17, 2017, consisted of 1000 images, with 100 images taken at each exposure time of {3000, 5700, 8400, 11100, 13800, 16500, 19200, 21900, 24600, 27300} microseconds. This shall henceforth be refereed to as the 'later' data set, or the second data set. The reason for the two separate data sets is because we wanted one set taken with shorter exposure times with a brighter light, and one set taken with longer exposure times with a dimmer light in order to see if there was a difference in the data trends when all the analysis was conducted. After the data was acquired, we wrote a program to come up with the variance and mean pixel count for each pixel set in each set, which we then plotted and got values for gain and read noise.

The gain and read noise values for both sets are detailed at the end of this report. At this point, it is unclear as to which points shall be used.

**Introductory Information:**

To begin with, we started this work on the Windows 10 OS, using Microsoft Visual Studio 2017. We later migrated to CentOS 7, and we will explain the functions of each program after the migration. To begin with, we installed the Basler SDK, which ended up being the Pylon 5 SDK. Out of the Python SDK, the C SDK, and the C++ SDK, we chose to work in the C++ environment. For general programming, we installed devtoolset-6. To work with fits images, we installed cfitsio, which is built for C (but works with C++). To mathematically work with data (generating best fit lines, etc.), We installed GSL, and to plot data, we installed plotutils. Optionally, to view fits files, we installed ds9.

**Taking Images:**

Initially, we started off making the usb\_grab program. This program accepts an integer input and outputs an image with the exposure in the image name. The main purpose of the program was to see how to change the exposure time to another value other than the default value. Also, the program served as an introduction to writing fits files. The file write\_basler\_fits.cpp as written in order to automatically write the fits file from a struct of values. The write\_basler\_fits function was reused several times in other programs.

Next, we wrote a program, cycle\_exposures, that allowed us to cycle through 10 exposures and output 10 images, all as fits files. The 10 exposures were set in an array in order to have the program take in image at a time, and then move onto the next image.

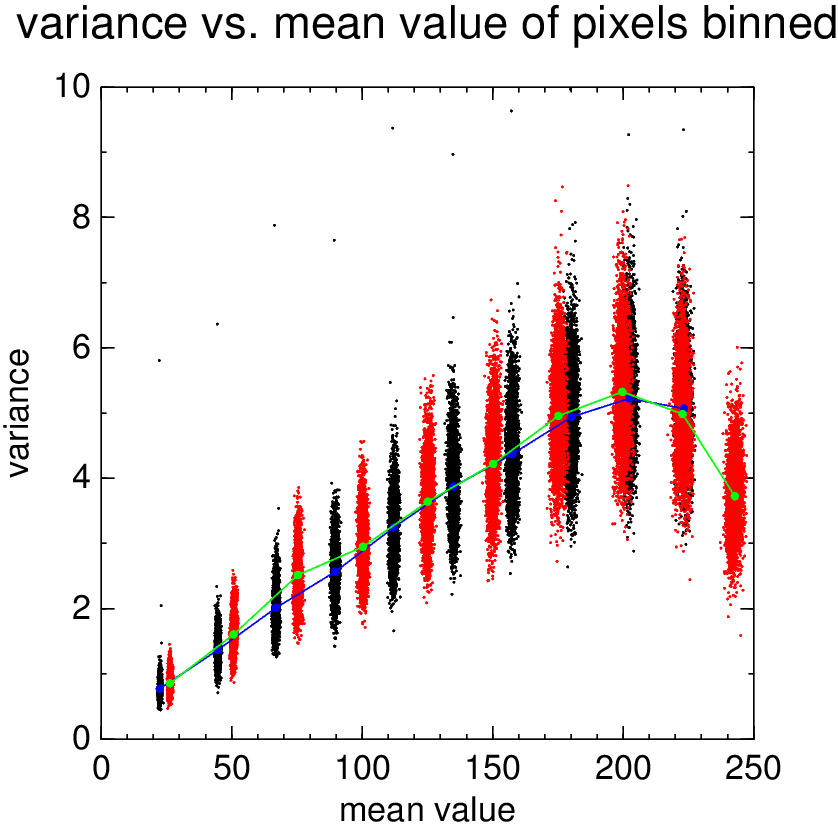
Next, we wanted to create a median flat image. In order to do that, we wrote gain\_read\_blank, which took 100 images at the lowest possible exposure time (59 microseconds). The program wrote 100 files, and then found the median pixel count at each pixel location. After all median values were found, the program outputted a file called median\_lowexp\_image.fits. In the context of this project, the median file was mainly just 1 count or 0 counts, so it wasn't an imperative step.

Next, we took an actual data set for this project. In gain\_read\_mixed\_exp, we added 10 different exposure times for the program, and then took 100 images at each of those exposure times, for a grand total of 1000 images. After all of the images were taken, we subtracted the median image from each of the images, and rewrote each image file. After that, we analyzed the data set.

**Analysis of the Data:**

In the sigma\_clip\_mixed program, we did quite a lot with the data in terms of programming. To summarize, we first opened every single one of the images previously taken. We then looked through every pixel for every set of exposure times, and found the mean pixel count value and the standard deviation for each of these pixel value sets. We then used mathematical formulas to find the variance for each of the standard deviation values, and outputted both the mean and variance values.

Now that we had the mean and standard deviation for different exposures at different pixels, we could plot the data and examine basic trends in the behavior of the data. In graph\_bins, we binned all of the data into their mean counts value, and found the in place median and in place variance in each of the bins. We then plotted the graph using a shell script built for plotutils, and this was the result.



Identification:

Red dots: higher exposure/lower light level binned data

Black dots: lower exposures/higher light level binned data

Green line: median of each mean bin of variances for higher exposure/lower light level binned data

Blue line: median of each mean bin of variances for lower exposure/higher light level binned data

The overall trend of the data is interesting. While there is a pretty linear relationship between variance and mean value, that changes at around 200 counts. Once there, the relationship actually shifts to an decreasing variance value as the mean value increases. In order to test this relationship out after one data set was taken (the first data set), we decided to test a second data set as well (longer exposure times). In the end, the trend was the same between both data sets, so we concluded that this was a camera quirk instead of a result of data collection/coding error.

In order to extract the gain and read noise from the raw data, we took the value of the in place mean values and in place variance values that had been extracted from graph\_bins. In the program linear\_fit, we graphed a line that best fit to the given data set. However, we gave multiple values for the data set in order to extract multiple line coefficients.

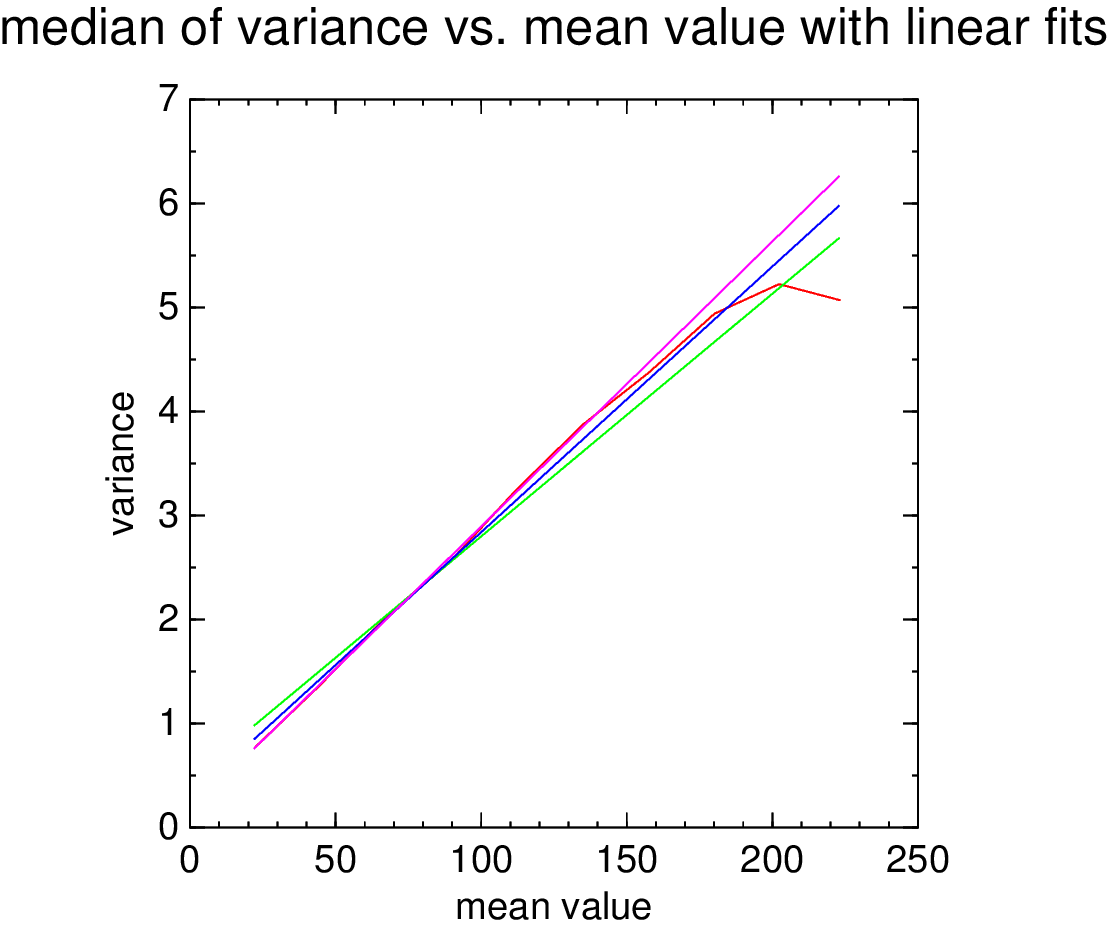
For the first data set, we got the best fit line for 3 regions:

1. The entire data set

2. Up to 200 counts

3. Up to 134 counts

The results were graphed in the linear\_fits program and are shown here:



Identification

Red line: original data medians

Green line: linear fit of all data

Blue line: linear fit up to 200 counts

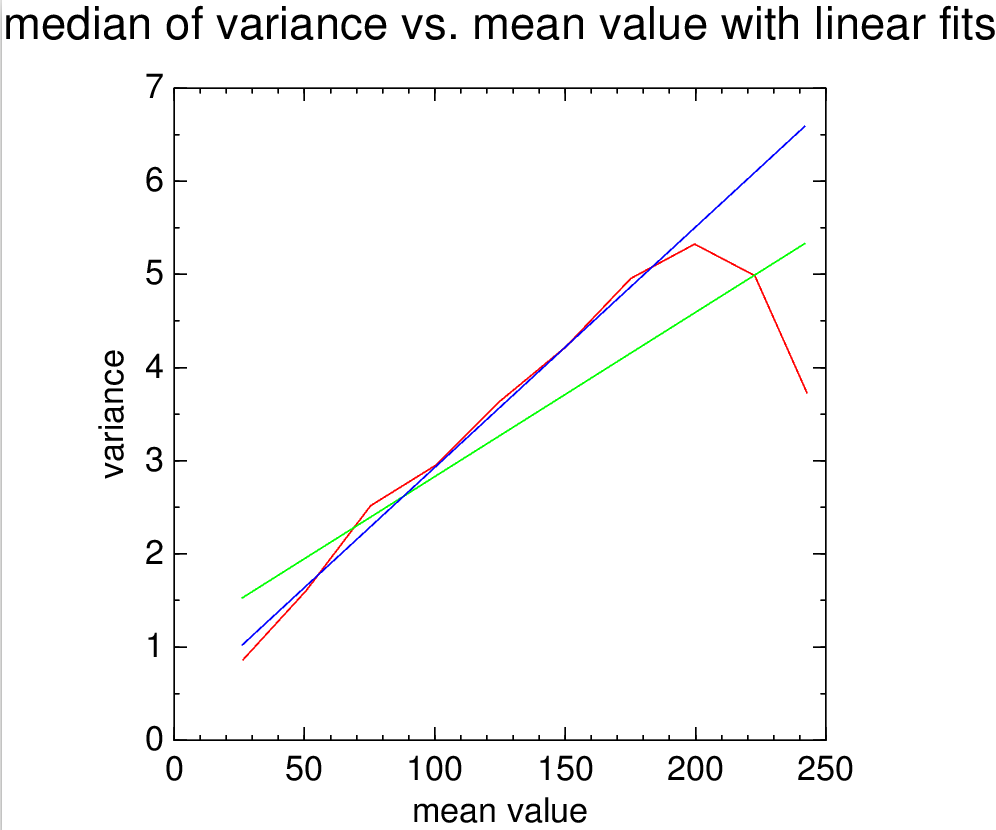
Magenta line: linear fit up to 134 counts

For the second data set, we got the best fit line for 2 regions:

1. The entire data set

2. Up to 200 counts

The results were graphed in the linear\_fits program and are shown here:



Identification:

Red line: Raw medians graphed

Green line: Linear fit to all data

Blue line: Linear fit to all data up to 200 counts

Therefore, from each of those best fit lines coefficients, we could extract gain and read noise coefficients. We decided to separately use all 5 coefficients in order to see if they were close in value or far apart, and what the average value overall would be.

For the first data set, the values for the gain ended up being:

For the entire data set: Gain: 42.8568

Read Noise: 19.9365

For only up to 200 counts: Gain: 39.151

Read Noise: 11.1406

For only up to 134 counts: Gain: 36.3025

Read Noise: 5.29139

For the second data set, the values end up being:

For the entire data set: Gain: 56.7046

Read Noise): 60.4323

For only up to 200 counts: Gain: 38.7462

Read Noise: 13.4454

Taking all of these values into account, I am unsure what to use for the overall gain and read noise of the camera. It looks like most values converge around 39 for the gain and 12 for the gain. The obvious data outlier in this set is in the second data set with all of the values being read in, because it diverges the most from a line for the last two values of counts. However, if the counts are only measured up to 200, we got very similar values. Therefore, which values we need to use to read the correct values for gain and read noise remain to be seen.