Afrah A. Ali

Physics 139

27 October 2022

Lab 3 Report

In this lab, I investigate strategies to identify pixels in image frames that have been impacted by cosmic rays through the construction of a median bias frame as well as the measurement of dark current for the DEIMOS detector.

Note: There are two Python notebooks that contain the code for my lab. The first notebook is titled "L3_AfrahAli" and the second notebook is titled "L3P3". I separated my lab into two different notebooks because attempting to run all of the code from one single notebook was overworking my computer. Please review the Python notebooks in the listed order.

PART ONE

We are provided with 9 bias frames that each have an exposure time of ~1 second. These bias images provide a measure for the baseline counts of the DEIMOS detector. For each of these bias frames, we are asked to determine the median bias level for each chip. To accomplish this, I wrote a for loop that runs through all 9 bias frames and collects the median value for chips 1-8 in each frame by opening the bias frame files and acquiring data from a specified index. Essentially, the code that I wrote to complete this task is looping through each bias file, locating a specified chip, and calculating the median for that chip. The information below specifies the median bias level for each frame and chip:

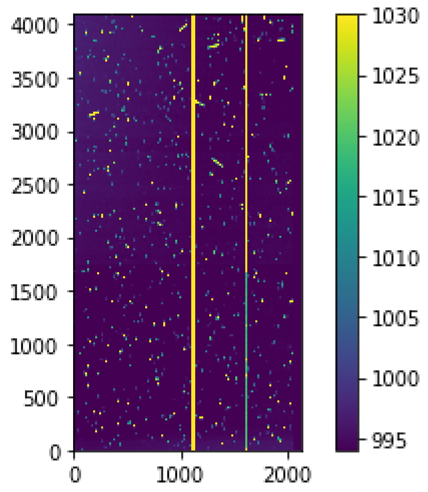
```
The median bias level for chip 1 in d0221_0090.fits.gz  is  994.0
The median bias level for chip 2 in d0221_0090.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0090.fits.gz  is  1019.0
The median bias level for chip 4 in d0221_0090.fits.gz  is  998.0
The median bias level for chip 5 in d0221_0090.fits.gz  is  1007.0
The median bias level for chip 6 in d0221_0090.fits.gz  is  1001.0
The median bias level for chip 7 in d0221_0090.fits.gz  is  1001.0
The median bias level for chip 8 in d0221_0090.fits.gz  is  991.0
The median bias level for chip 1 in d0221_0091.fits.gz  is  994.0
The median bias level for chip 2 in d0221_0091.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0091.fits.gz  is  1019.0
The median bias level for chip 4 in d0221_0091.fits.gz  is  998.0
The median bias level for chip 5 in d0221_0091.fits.gz  is  1006.0
The median bias level for chip 6 in d0221_0091.fits.gz  is  1000.0
The median bias level for chip 7 in d0221_0091.fits.gz  is  1000.0
 The median bias level for chip 8 in d0221_0091.fits.gz  is  991.0
 The median bias level for chip 1 in d0221_0092.fits.gz  is  993.0
 The median bias level for chip 2 in d0221_0092.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0092.fits.gz  is  1019.0
 The median bias level for chip 4 in d0221_0092.fits.gz  is  997.0
The median bias level for chip 5 in d0221_0092.fits.gz  is  1006.0
The median bias level for chip 6 in d0221_0092.fits.gz  is  1000.0
The median bias level for chip 7 in d0221_0092.fits.gz  is  1000.0
 The median bias level for chip 8 in d0221_0092.fits.gz  is  991.0
 The median bias level for chip 1 in d0221_0093.fits.gz  is  993.0
 The median bias level for chip 2 in d0221_0093.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0093.fits.gz  is  1019.0
 The median bias level for chip 4 in d0221_0093.fits.gz  is  997.0
The median bias level for chip 5 in d0221_0093.fits.gz  is  1006.0
The median bias level for chip 6 in d0221_0093.fits.gz  is  1000.0
The median bias level for chip 7 in d0221_0093.fits.gz  is  1000.0
 The median bias level for chip 8 in d0221_0093.fits.gz  is  991.0
 The median bias level for chip 1 in d0221_0094.fits.gz  is  994.0
 The median bias level for chip 2 in d0221_0094.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0094.fits.gz  is  1019.0
 The median bias level for chip 4 in d0221_0094.fits.gz  is  997.0
The median bias level for chip 5 in d0221_0094.fits.gz  is  1006.0
The median bias level for chip 6 in d0221_0094.fits.gz  is  1000.0
The median bias level for chip 7 in d0221_0094.fits.gz  is  1000.0
 The median bias level for chip 8 in d0221_0094.fits.gz  is  991.0
 The median bias level for chip 1 in d0221_0095.fits.gz  is  993.0
 The median bias level for chip 2 in d0221_0095.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0095.fits.gz  is  1019.0
 The median bias level for chip 4 in d0221_0095.fits.gz  is  997.0
The median bias level for chip 5 in d0221_0095.fits.gz  is  1006.0
The median bias level for chip 6 in d0221_0095.fits.gz  is  1000.0
The median bias level for chip 7 in d0221_0095.fits.gz  is  1000.0
 The median bias level for chip 8 in d0221_0095.fits.gz  is  991.0
 The median bias level for chip 1 in d0221_0096.fits.gz  is  994.0
 The median bias level for chip 2 in d0221_0096.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0096.fits.gz  is  1019.0
 The median bias level for chip 4 in d0221_0096.fits.gz  is  997.0
The median bias level for chip 5 in d0221_0096.fits.gz  is  1006.0
The median bias level for chip 6 in d0221_0096.fits.gz  is  1000.0
 The median bias level for chip 7 in d0221_0096.fits.gz  is  999.0
 The median bias level for chip 8 in d0221_0096.fits.gz  is  991.0
 The median bias level for chip 1 in d0221_0097.fits.gz  is  993.0
 The median bias level for chip 2 in d0221_0097.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0097.fits.gz  is  1019.0
 The median bias level for chip 4 in d0221_0097.fits.gz  is  997.0
The median bias level for chip 5 in d0221_0097.fits.gz  is  1006.0
The median bias level for chip 6 in d0221_0097.fits.gz  is  1000.0
 The median bias level for chip 7 in d0221_0097.fits.gz  is  999.0
 The median bias level for chip 8 in d0221_0097.fits.gz  is  991.0
 The median bias level for chip 1 in d0221_0098.fits.gz  is  994.0
 The median bias level for chip 2 in d0221_0098.fits.gz  is  996.0
The median bias level for chip 3 in d0221_0098.fits.gz  is  1019.0
 The median bias level for chip 4 in d0221_0098.fits.gz  is  997.0
The median bias level for chip 5 in d0221_0098.fits.gz  is  1006.0
 The median bias level for chip 6 in d0221_0098.fits.gz  is  999.0
 The median bias level for chip 7 in d0221_0098.fits.gz  is  999.0
 The median bias level for chip 8 in d0221_0098.fits.gz  is  991.0
```

There is variation in the median bias level from chip to chip. There is very minimal variation in the median bias level from frame to frame for the same chip; the variation is so small that it can be treated as negligible.

The next part of the lab asks us to construct a median bias image for each of the DEIMOS chips (1-8) by taking the median bias level at each pixel position across the 9 bias images. To do this, I create a loop that finds the median bias image for each chip in each frame number and compiles them into an array. The result is 8 2D arrays that each hold information about the median bias image for their specified chip. These arrays are then written to a file named DEIMOS_bias.fits. This is done by creating an empty primary HDU with the median bias image array for each chip stored in extensions 1-8. This packs all the image arrays into a single fits file.

PART TWO

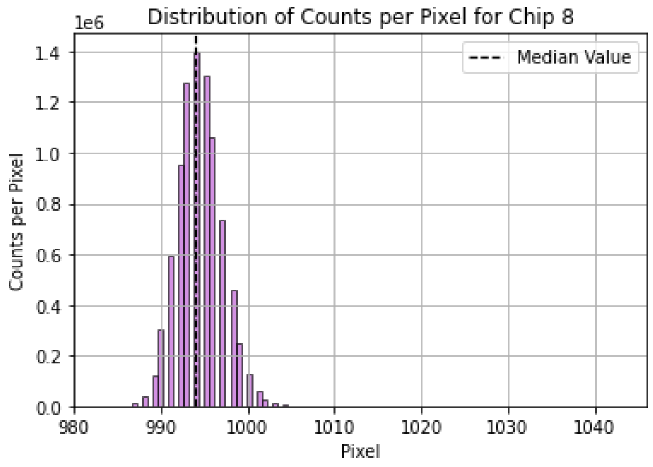
I have displayed the image for chip 8 on frame *d0221_0099.fits.gz* below:



This image has two distinct lines, a dark background and several scattered marks of different colors littered throughout it. Some of these marks appear to be bigger and more notable than others.

For the image displayed above, I have plotted the distribution of counts per pixel for pixels on chip 8 with values within ±50 of the median pixel value for the image. To do this, I first initialized a variable called *m* to carry the median of the specified image. From *m*, I added and subtracted 50 to obtain the minimum and maximum values of the range pertaining to my plot. I found that the minimum value was 944 and the maximum value was 1044. Then, I used the *np.where* function to slice the image above by finding the x and y locations for the pixels within the specified interval.

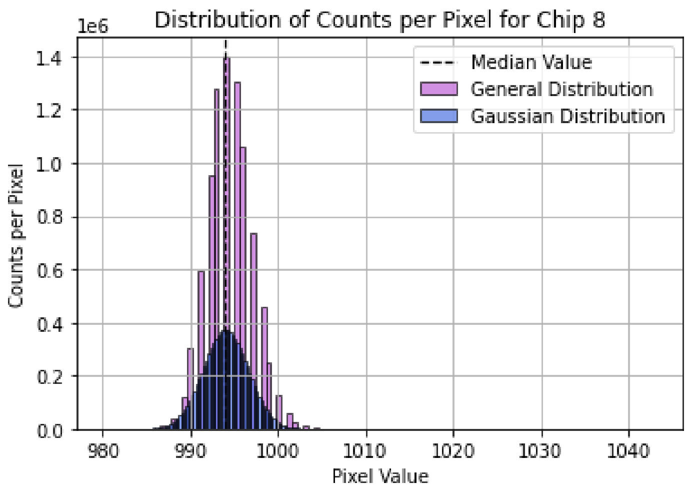
The following histogram displays the distribution of counts per pixel for pixels on chip 8 with values within ±50 of the median pixel value for the image. The dashed line on the histogram denotes the median value for chip 8 in frame 99.



To find the fraction of total pixels on chip 8 that have a value within ±50 of the median pixel value, I found the length of the image slice that I took using the *len()* command. I divided this value by the total number of pixels in the entire image of frame 99. I found this value by finding the shape of the image on frame 99 and multiplying the x and y dimensions together. The final expression was: (Length of Image Slice)/(Total # of Pixels on Image). After calculating the result of this expression, I multiplied the result by 100 to convert it into a percentage value. 99.71% of pixels have a value within ±50 of the median pixel value.

Next, we are asked to find the standard deviation of counts per pixel for the subset of pixels that are within ±50 of the median pixel value. To calculate this, I used the *np.std* function to calculate the standard deviation of the image slice that I created previously. The standard deviation for the image slice is 2.58 pixels. I used the same function to calculate the standard deviation of the entire image which is 1367.43 pixels. These measures of dispersion are significantly different because the total data set being used to calculate the standard deviation for the entire image is much greater than the data set being used to calculate the standard deviation for the sliced image. The disparity in the dispersion between the entire image and the sliced image is proportionate to the amount of data being used to calculate each dispersion value.

To further analyze the nature of the way pixels are distributed in the sliced image, we are asked to overplot a Gaussian distribution on top of the general distribution of counts for pixels with values within ±50 of the median pixel value. To plot this Gaussian distribution, I used the *np.random.normal* function with the following arguments: the median of the image slice, the standard deviation of the image slice, and the total number of pixels in the entire image of frame 99. The Gaussian distribution is shown along with the general distribution on the following graph:



It is clear that the general distribution emulates the general shape of a Gaussian distribution. Both of the distributions' median values are in roughly the same place.

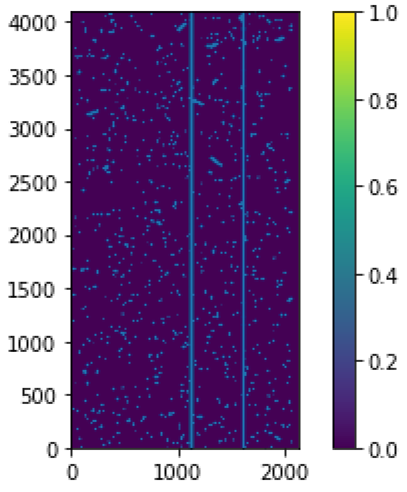
After creating this graph, we are asked to find the fraction of total pixels on chip 8 that have a pixel value that is 1σ above or below the median pixel value using the value for σ calculated from the image slice. To do this, I first found all the pixels that are above 1σ using the *np.where* function with the following argument: *Image Slice > Standard Deviation of Image Slice + Median of Image Slice*. In doing this, I am telling Python to combine the median of the image slice with one standard deviation into a single value, and find all the values of the image slice that are above that value. The values that satisfy the condition specified in the argument above are saved into an array. I repeat this process to find all the pixels that are 1σ below the median value by tweaking the argument inside the *np.where* function to be: *Image Slice > Median of Image Slice - Standard Deviation of Image Slice*. This argument tells Python to subtract one standard deviation from the median value, and find all the pixels in the image slice that are less than the resulting quantity. The values that satisfy the condition specified in this new argument are saved into a separate array.

Now, to find the fraction of pixels that are outside of 1σ, I found the sizes of the two arrays created through the previous process. One array holds information about all the values above 1σ, and the other holds information about all the values below 1σ. I added the sizes of both of these arrays, divided that sum by the total number of pixels in the image slice, then multiplied the resulting number by 100 to obtain a percentage value. I found that 31.38% of the total pixels on chip 8 have a pixel value that is 1σ above or below the median pixel value of the image slice. This means that about 68.62% of the total pixels on chip 8 have a pixel value that is above or below one standard deviation of the median pixel value. This finding aligns with the general breakdown of standard deviations of a Gaussian distribution;

generally, 68.2% of the data being used to generate a Gaussian distribution lies within 1σ of the average value of the data set. Therefore, the percentage that I found agrees well with the expectation for a Gaussian distribution.

We are then asked to repeat the previously delineated steps to find the fractions of pixels that lie outside of 5σ, 10σ, and 100σ. To find these fractions, I followed the exact same process used to find the fraction of pixels outside of 1σ. The only difference in the code used to find the specified fractions is that I multiplied the standard deviation of the image slice in the arguments of the *np.where* function by either 5, 10, or 100. I found that 0.012% of the total pixels lie outside of 10σ, 0.065% of the total pixels lie outside of 5σ, and 0% of the data lies outside of 100σ. The fractions calculated align very closely with the expected values for a Gaussian distribution.

Next, we are asked to make a figure illustrating the location of the pixels with values that are outside of 10σ. To do this, I created two arrays; one holds values for the number of pixels that are above 10σ in the entire image for frame 99 and one holds values for the number of pixels that are below 10σ in the entire image for frame 99. Each slot in these arrays contains an ordered pair with an x and y position that pertains to the location of a pixel either above or below 10σ. The zeroth index of each array holds the y positions of pixels and the first index holds the x positions of pixels. I plotted the zeroth and first indices of both arrays onto the same scatter plot, effectively producing an image that illustrates the locations of pixels with values that are outside of 10σ. The image is attached below:



Because this is an image and not a graph, I could not include axes labels or a plot title so I will include them in this lab report. The x-axis presents the x-positions of the pixels outside of 10σ, and the y-axis presents the y-positions of the pixels outside of 10σ.

Next, I examine the data from chip 8 in the other 4 dark frames provided; frames 100-103. For each image, we are asked to find the fraction of pixels that are outside of 10σ. To do this, I created two arrays for each image frame; one array holds the number of pixels that are above 10σ and the other holds the number of pixels that are below 10σ. To create the arrays containing the values above 10σ, I used the *np.where* function with the following argument: *Image (100-103) > 10 * Standard Deviation of Image Slice from Frame 99 + Median of Image (100-103)*. To create the arrays containing the values below 10σ, I used the *np.where* function with the following argument: *Image (100-103) < Median of Image (100-103) - Standard Deviation of Image Slice from Frame 99*. After creating these arrays for each image, I calculated the size of each array using the *np.size* function, added the sizes together, and divided that number by the overall size of the image for each frame. For frame 100, 0.605% of pixels lie outside 10σ. For frame 101, 0.598% of pixels lie outside 10σ. For frame 102, 0.598% of pixels lie outside 10σ. For frame 103, 0.611% of pixels lie outside of 10σ. The following table concisely summarizes this information:

Frame Number	**Fraction of Pixels Outside of 10σ**
99	0.012%
100	0.605%
101	0.598%
102	0.598%
103	0.611%

Next, we are asked to find the pixels that are outside of 10σ in every given frame. To accomplish this, I created arrays that store information about the number of pixels above 10σ in each frame. These arrays have varying y-dimensions but the same x-dimension. To standardize the shape of the arrays to be more similar, I transposed each array to have the same y-dimensions but differing x-dimensions. These arrays are titled *sig99t - sig103*t. I then wrote the following code:

```
q = np.array([x for x in sig99t if x in sig100t])
o = np.array([x for x in q if x in sig101t])
w = np.array([x for x in o if x in sig102t])
```

```
v = np.array([x for x in w if x in sig103t])
print(v)
```

The first line of this code finds the values that are similar between *sig99t* and *sig100t* and saves them into a value *q*. The next line of codes finds the values that are similar between *q* and *sig101t* and saves them into a value called *o*. This process is repeated until there are no more transposed arrays to make comparisons with. The final value that contains information about all pixels that are outside of 10σ is the variable *v*.

After this, we are asked the following question: "What would you expect for pixels belonging to a bad pixel column versus those impacted by a cosmic ray?". Each pixel is a semiconductor that allows current to flow through. A damaged pixel leads to a malfunctioned semiconductor that has less controlled current flowing through it. A detector would assume that there is an image on the damaged pixel because of this random current flow through the damaged pixel. I would expect that a detector would process a bad pixel the same way it would process a pixel impacted by a cosmic ray; there is little to no distinction between the mechanisms by which each type of damaged pixel is analyzed by the detector.

The following questions ask about what fraction of pixels on chip 8 have been impacted by a cosmic ray in at least 1 of the 5 dark frames, and about what fraction of pixels on chip 8 are likely associated with general defects in the detector. I am unsure about how to find either quantity, but it is my expectation that once you have the percentage of pixels that are bad for one of these cases, you can subtract that percentage from 100% to get the number of pixels that are bad for the other case.

The remainder of this lab has been excluded from the report. This decision was made on the basis that any and all attempts made to produce viable completions of these sections were deemed inadequate for inclusion. The work that has been included is a reflection of the data and findings for which I am confident in reporting. In contrast, attempts at the remaining section proved to be in juxtaposition to the quality of the included work and subsequently detracted from its merit overall. It is the hope that the reader finds the included reporting to be a sufficient attempt at summarizing the viable outcomes of this lab, and that it is taken in consideration in its own right, rather than relative to the extendible form that this lab might have taken (provided additional resources and capacity to produce such an extensive report).