Project #3: CCD Noise

Digital imaging is primarily based on the Charge Coupled Device (CCD). In a digital camera, light is focused onto a CCD chip, which consists of a 2D array of photon detectors. These semiconductor detectors use the photoelectric effect to count the photons incident upon them; incoming photons knock electrons into the conduction band where they contribute to a current that can be read electronically. A digital image is a 2D array of pixel values, with each pixel value corresponding to the count in a particular detector. High counts correspond to bright pixels and low counts to dim pixels, thus forming a black and white image. For color images, each pixel counts red photons, green photons, and blue photons, and these RGB counts provide a color for each pixel in addition to a brightness.

CCD cameras, which are about 70 times as sensitive to light as film cameras, have led to a revolution in astronomy, where it is important to have high sensitivity in order to image dim objects. However, CCD images contain several types of noise that need to be understood. In this project you will study the statistics of two types of CCD noise in order to understand the process of removing this noise from CCD images. This is an important part of processing CCD images (called image reduction) that must be done before they can be used for science.

First we will study bias and readout noise in a CCD. Readout noise is due to the limitations of the electronic process through which pixel values are read; instead of outputting the exact number of photons that hit the detector, the output pixel value instead scatters around the exact value with some standard deviation. In this project we will look at both the standard deviation and the distribution to the readout noise.

Readout noise leads to a second problem. CCDs are designed to only output positive pixel values. This means that if a pixel has a zero count, readout noise will cause it to scatter only above its true value, so on average the output pixel value will be higher than the true value. The solution to this problem is to bias all the pixels so that their zero is actually a positive number; in other words, a pixel with no incident photons will yield a pixel value that scatters around a positive number rather than zero. The bias value can then be subtracted from the image, thus zeroing the pixels so that a zero pixel value corresponds to zero incident photons. Unfortunately, each pixel has a different value of bias, so we must perform this process pixel by pixel.

The second type of noise we will consider is called Dark noise. This is due to the fact that the camera itself produces infrared thermal photons which are counted by the pixels in the same way as pixels from the sky. Thus we would get nonzero pixel counts even if we took an image with the lens cap on. Dark noise can be reduced by cooling the camera, hence the fact that state of the art telescopes like the Keck on Mauna Kea have CCDs that are cooled with liquid Helium to temperatures of just a few Kelvin. One of the advantages of space telescopes is that space is an intrinsically cold environment.

In order to subtract Bias and Dark noise, astronomers take calibration frames along with images of the sky. Astronomers typically take three types of calibration frames, Bias frames, which are zero second exposure images, Dark frames, which are images taken with the same exposure time as the sky images but with the shutter closed, and flat frames, which we will not discuss here.

Bias and readout noise in Bias Frames

In order to study bias and readout noise (and figure out how to correct for them), you will examine so-called Bias frames (images) taken with a CCD camera. Bias frames are essentially zero-second exposures, so their pixel counts are just the bias values plus readout noise. Astronomical images are stored in FITS format (Flexible Image Transport System) and so must be read differently than text files. On Canvas there is a python program that demonstrates how to read the pixel values from a FITS image into a 2D array. Images from this CCD camera are 1472 by 2184 pixels. It also includes a demonstration of how to read the set of 9 Bias images into a list of 2D arrays. Use np.shape() to look at the shape of this list; the shape of an array tells us the dimensions of the array. For your list the shape should be (9,1472,2184). Thus the list is a 3D array of nine 2D images stacked on top of each other. First, you need to create an average of the 9 bias frames to create a frame containing only the bias values. To do this, use np.mean() with the option axis=0 (look up the numpy.mean page on the web). This tells python to only average over the first dimension (the zeroth axis) so that the result is a 2D array containing the average of the nine values for each pixel. Subtract the average frame from the list of Bias frames (just subtract it directly from the list). All of the pixel values in the list of frames should now be pure readout noise.

You can find the mean and the standard deviation of the readout noise by using np.mean() and np.std() directly on the list of frames without specifying an axis; this should give you the standard deviation of the read noise. You should get the mean to be very close to zero; if its not, then something's wrong. To see the distribution of the read noise, make a normalized histogram of the pixel values. To do this, you need to make the 3D array of values into a 1D list. To do this, use the np.ravel() command, which collapses an array into a 1D list. Since you have a LOT of values, you should be able to get a very smooth looking histogram. Of the distributions we have studied, what does the histogram most look like? (hint: think Gaussian). Plot the function on top of the histogram and see how well you can make it fit. What can you conclude about the amplitude and the distribution of readout noise?

Dark Frames

In order to study dark noise, you will examine dark frames with various exposure times. Although these images are taken with the shutter closed, during the exposure the pixels will count infrared photons coming from their surroundings. These photons will also be counted when the shutter is open, so the dark frames allow us to determine how many dark photons we expect to be in an image so that we can subtract them. Now, each pixel has a constant probability per time of receiving an infrared photon, so the distribution of counts in the pixels of a dark frame should be Poisson distributed. However, Dark frames also have bias and readout noise. To correct for the bias, you should subtract the average bias frame from each dark frame. Once the bias is subtracted, the counts in each pixel will be a combination of Poisson distributed infra-red photon counts and Gaussian distributed readout noise.

Read in the shortest exposure time dark frame. Subtract the average bias frame. Unfortunately, we can't reliably calculate the mean and standard deviation of the pixel values due to the presence of so called hot pixels. These are pixels on the chip that for some reason always give abnormally high values; these values can throw off our standard deviation calculations in particular, since the squares of these large values can dominate the average. Thus we will mostly be working with histograms of the pixel values, which are more reliable. Using np.ravel() as before, make a histogram of the pixel values in the dark frame with the bias subtracted. Write a function to generate the probability distribution of a Poisson distribution and show that it doesn't fit the histogram well.

How do we calculate the distribution of pixel values that are the sum of numbers drawn from two different distributions? This requires the use of a convolution of the two distributions. In order to understand this, lets consider a pixel that has a value x, which is a combination of the Poisson distributed photon count and Gaussian distributed noise. Now, suppose that the noise had a value y and the photon count was $x - y$ (these sum to the value x). The probability of this case is given by the probability of the Gaussian noise being y, given by $P_G(y)$, times the probability that the photon count was $y - x$, which is $P_P(y - x)$, where P_G is the Gaussian distribution and P_P is the Poisson distribution. We can sum over all the possible values of y using an integral, so that the total probability of the count being x is given by

$$P(x) = \int P_G(y) P_P(y - x) dy \tag{1}$$

This is called the convolution of the two probability distributions. In Python, we can calculate the convolution in the following way:

- Define functions for both the Gaussian and Poisson distributions, make sure you use the numpy versions of all the power function and import the factorial function from scipy.special (from scipy.special import factorial).
- Generate a list of x values for the Poisson distribution. This list will also be the x values for the convolution function, so you will want it to cover the same range as the histogram you are going to fit.
- Generate a list of y values for the Poisson distribution. You can use the count at which the histogram peaks as a guess for the value of λ. Now, since the Poisson distribution isn't defined for negative values you will have infinities (inf) in your list. To remove these infinities, use the command list[list == inf] = 0, where list is the name of your Poisson y list. This will replace any inf in your list with zero.
- Generate a list of y values for the Gaussian distribution using a shorter x list centered on zero. For σ use the standard deviation you determined for the readout noise above and use zero for x_0. Make sure that you use the same spacing of x values for this list as you used for the Poisson y list.
- Import convolve() from scipy.signal. Input the y lists for the two distributions and use the option mode='same'. The output will be the convolved probability distribution at the values of x that were used for the Poisson y list.

- Plot the convolved distribution on top of the histogram and tweak the values of λ and σ to get the best fit.

Calculate the best fit λ value for each of the different Dark frame exposure times. If there really is a fixed probability per time of a photon being counted in a pixel, then the value of λ should be proportional to the exposure time. Plot λ vs. exposure time and see if this is indeed true.

Discussion: Discuss the distributions of the readout noise and the dark frames with the bias removed. Is the readout noise distribution fit well by a Gaussian? Are the dark frame pixel values with the bias removed well modeled by a Poisson distribution convolved with a Gaussian? Does the value of λ increase proportional to exposure time? Suppose you had an image of the sky taken with an exposure time t. Discuss how you would correct the image for bias, readout noise, and dark noise using what you have learned in this project. In particular, discuss what effects can be corrected for and which can't.