Lab 2: CCD/Digital Imaging

(Due: 2017 Feb 22)

CCD Cameras

The purpose of this lab is to explore the characteristics of the CCD camera, measure the signal to noise ratio in your images, learn about calibration (bias and dark frames), and learn how to manipulate images in Python. This writeup describes two different CCD cameras, by two different manufacturers: the Apogee Alta U260, and the SBIG (Santa Barbara Instruments Group) STL-1301. We will be using the latter for this course. The hearts of these cameras are their CCD chips, both made by Kodak. Details for the two cameras are given in the table below:

Camera	**Apogee Alta U260**	**SBIG STL-1301**
CCD Chip	KAF-0261E	KAF-1301E
Number of Bits	16 bit (max val. 65536)	16 bit (max val. 65536)
Number of Pixels	512 x 512	1280 x 1024
Size of Pixel	20 mm	16 mm
Read Noise	22 e^-	17 e^-
Dark Current	1 e^-/p/s @ –30 C	0.5 e^-/p/s @ –30 C
Peak QE	63%	73%
Full Well Capacity	200,000 electrons	120,000 electrons
Gain (e^-/ADU)	?	1.6

The term CCD is short for Charge-Coupled-Device, and can be thought of as an array of electron wells (or buckets). A photon striking one of these buckets has some rather high probability, given by the quantum efficiency (QE) of releasing an electron into the well. Once released, the electron is trapped in the well until it is read out. One can expose the CCD to light for some length of time, and during the exposure the well begins to fill with electrons. On readout, the number of electrons in each pixel well are counted, and a number (called an ADU, or Analog-Digital Unit) proportional to the number of electrons is recorded for that pixel. The image consists of the array of these ADU numbers. For example, when 100,000 photons hit a particular pixel of the STL-1301, 73% of them (assuming peak QE) will release electrons, giving 73,000 electrons. On readout, these electrons will be converted to a number (ADU) at 1.6 electrons/ADU, so the readout would be 45625. These are called “counts,” and the units are ADU (also referred to as DN, which stands for “digital number”). In the final image, the brightness of that pixel *should be* represented by the number 45625. However, the number will actually be higher because there are other sources of excess electrons, which can be regarded as noise (unwanted counts).

Let's take a look at some of the sources of unwanted "signal," i.e. noise. A general equation for the number of counts *C* [ADU] measured by the camera is:

$C(T,t) = [I\ \mathrm{QE} + d(T)]\ G\ t + b(T)$ (uncalibrated image) [1]

where *I* [photons/s] is the intensity of incident photons on the CCD, QE [e^-/ photon] is the quantum efficiency, *G* [ADU/e^-], *t* [s] is the exposure time, *d* [e^-/s] is the dark current, *b* [ADU] is the bias, or read-out noise.

Bias

During the reading process, noise is generated in the readout circuitry that has nothing to do with photons. The amount of this noise is fixed in time, but varies from pixel to pixel. This is because the individual pixels are not all precisely alike. You can measure this noise, called *bias* noise, and subtract it from your CCD images, by exposing for 0 seconds (an instantaneous exposure) and reading out the chip. The shutter of the camera is closed, and the chip is read immediately, so that no electrons build up. Take a bias image in *MaxIm DL* by selecting "Bias" on the "Expose" tab. Notice that the exposure time (minutes and seconds) adjustment is grayed, because a bias is an instantaneous exposure. Click "expose" and you will see a rather uninteresting looking, noisy image appear. Every image you take will have this kind of noise in it. Set the camera temperature to –10 C, and take another Bias frame. **Use the Information window (from the View menu) to measure the average value in the image, by setting the mode to Area. Write it down in your log book. Save the image as Bias.fit.**

To correct for bias, you will subtract the bias image, taken with the same temperature, from the science image, which leaves:

$C'(T,t) = [I\ \mathrm{QE} + d(T)]\ G\ t$ (bias corrected image) [2]

Dark Current

In addition to this readout noise, there is also thermal noise. When you expose a CCD chip in the dark (i.e. there are no photons hitting the chip), a small number of electrons will slowly build up each second, so that in 10 s you will have 10 times as many as at 1 s, etc. The number of "leaky" electrons depends on the temperature, so at high temperature you will have more leaking electrons and at low temperature you will have fewer. This is called dark current. Dark current is the main reason why we cool our CCD chip to as low a temperature as possible. With the camera temperature set to –10 C, take a 300 s dark frame and measure its average value, as before. **Write down the average value, and make a note of the temperature setting in your logbook. Save the image as Dark-10C.fit.** Now lower the temperature by 20 C in steps of 5 degrees (e.g. –15, –20, –25 and –30 C) and take a 300 s dark frame at each setting. **Write down the average from the information window, and the temperature setting for each image, then save the image with Dark-15C.fit, etc.** You should note a steady decrease in the average level. We take dark frames like this in order to subtract this unwanted noise from the images, but there is another important purpose. For the last image that you took, scale the brightness (the screen stretch) until you see a few bright pixels scattered around. These are called "hot" pixels because they show a lot more sensitivity to temperature than most pixels. When we subtract a dark frame from an image, these noisy variations are largely eliminated. However, because this noise depends on both temperature and exposure duration, we have to make sure that the dark frame that we subtract was taken at the same temperature and the same exposure duration as the image we apply it to. Dark frames become increasingly more important when the chip temperature is high, and/or when the exposure time is long.

To correct for dark current, you will subtract the bias-corrected dark image, taken with the same exposure time and temperature, and representing the quantity $d(T)\ G\ t$, from the science image, which leaves:

$C''(T,t) = I\ \mathrm{QE}\ G\ t$ (dark corrected image) [3]

Calibration

The dark frames we took have both dark current and read noise (bias). We can remove the read noise by subtracting the bias frame from the dark frame. Let's do this by hand using the Pixel Math tool in *MaxIm DL*. First, load the Bias image that you saved, and the dark frame taken at the same temperature. If the Bias was taken at –10 C, load the Dark-10C.fit image. With the dark frame image selected, choose Pixel Math from the bottom of the Process menu. Choose "Subtract" as the Operation, and in the Image B drop-down list, select the Bias image. Make sure that "Add Constant" is zero, and click OK. The operation we have done is Dark – Bias, and if you look at the resulting image, you should see that the right edge, which used to be rather bright, is now "flat" (uniform, like the rest of the image). However, the histogram in the screen stretch window is cut off. That is because the Dark image values (noise) averaged somewhat higher than the Bias, but in some pixels (almost half), the values were actually lower than the Bias. Since the values cannot go negative, values Dark – Bias < 0 are lost. To avoid this, select Undo from the Edit menu, and do the Pixel Math operation again, but this time enter 100 in the "Add Constant" box. Click OK. The operation this time is Dark – Bias + 100, and now the histogram looks normal—the values range from around 50 to 180 or so. This subtraction of the Bias from the Dark frame is a type of calibration. *MaxIm DL* has some tools to make standard calibration easier, which we will see in a moment.

Select a Star Field and Take Images

Find an available star around 8th magnitude and point the telescope, focus, and prepare to take a series of images. It will be more interesting if there is a small galaxy or other nebula in the frame, but do not choose a field with a large nebula. We will want some flat areas in your image. **Write down the designation of your chosen star, and its magnitude, in your log book.** Set the CCD temperature to the lowest possible temperature for the observing session (the camera can cool to about 40 C below ambient). Usually, you should be able to reach –30 C. Wait for the temperature to stabilize. Everyone should use the same temperature, so that we can all use the same bias and dark frames, which we will take as a class.

Using *MaxIm DL*'s "sequence" ability, you will take a series of images of your star. In the camera's Expose tab, select "Autosave." Click the "Options" button and select "Set Destination Path…" to choose where to save your files. Then click "Options" again and select "Setup Sequence." In the new window that pops up, type in a suitable Autosave Filename. Make sure that only the first line is checked, select "Type" as "Light," select the "luminance" filter, use a "Suffix" of "L" (for luminance), an "Exposure" time of 20, "Binning" 1, and "Repeat" 32. This will take a series of 32 exposures of 20 s each, and will append your autosave name with "_001L" "_002L", etc. Make sure that the "Delay First" and "Delay Between" settings at the bottom are 0. Click "OK" to close the window. In the "Expose" tab, where you should now be, click "Start" to start taking the images. In about 15 minutes the images should be done.

Write down in your log book the details of what you did (filenames, folder, temperature settings, time you began each series) and anything else that happened, such as false starts and restarts, etc.

Calibration of Images

We are going to calibrate the star images, which basically means subtracting the bias and dark frames. This is something that we will do for ALL images in the future, and is required to get nice-looking and photometrically accurate images. As a class, we will take a series of 20 bias and 20 dark frames (each dark having 20 s duration), so you should find those in a standard place on the computer. We take 20 frames in order to average them and obtain a more representative and less noisy result. To apply them, we will first "Set Calibration…" in *MaxIm DL*. In the "Process" menu, select "Set Calibration…" and at the bottom of the window that opens, enter the Source Folder in the Auto-Generation area. Then click "Auto-Generate" and after a minute or so the area at the top will contain all of the image sets (darks and biases) needed for calibration. Later we will learn about another part of the calibration, called a Flat Frame, but for now we will not use flats.

Once the image sets are found, make sure that "Combine Type" is set to "Median," and click the button near the bottom that says "Replace with Masters." It may take some time to do the processing. When it is done, click "OK" to close the window. To see the application of calibration, simply read in one of your star images ("Open" from the "File" menu), then under the "Process" menu select "Calibrate." You should see the image improve substantially. If you missed it, do "Edit" "Undo" and "Edit" "Redo" a few times to toggle the application of calibration on and off.

As you might imagine, applying the calibration to a lot of files could get tedious, but *MaxIm DL* has a shortcut. Under the "File" menu, choose "Batch Save and Convert…" to apply calibration to all of your star images. In the window that opens, use "Select Files…" to select all of the files you want to calibrate. Make sure that the "Perform calibration" box is checked. Click on "Path…" in the "Destination" area, and create a new subdirectory under the main directory that your files are in. Call this new subdirectory "Calibrated." When all is ready, click "OK" and in a minute or two all of the files will be calibrated and written to the new directory.

Combining Images

For your now-calibrated star images, we will combine images in series of 2, 4, 8, 16, and 32 images. By combining images, we are effectively increasing the exposure time. Combining two 20 s images gives an effective exposure time of 40 s. Combining 32 such images gives an effective exposure time of 640 s. To combine images, in *MaxIm DL* choose "Stack…" from the "Process" menu. Choose "Add Folder…" and navigate to where you saved your Calibrated images, and select it to load all of your calibrated images. After the group is created, click the Align tab. Open the group list and select the 16th image, right-click and choose "Reference Image" to set this as the reference. Make sure Align Mode is "Auto – star matching." Go through all of the images one at a time by clicking "Next Image," to check that they all look okay. The images may shift slightly from one to the next. To compute the alignment needed to bring them all to the same location at the reference image, choose "Compute All" in the Image Alignment drop-down list. If you go through the images again, you should see that they do not shift.

You will now combine different sets of images using the Combine tab, using "Sigma Clip." The procedure is complicated, so you will be shown how to do it in class. You will end up with a series of images containing 2, 4, 8, 16 and 32 of the individual images. Save the images by adding a suffix 2, 4,..32 to indicate how many images are combined in the resulting image.

Be sure to include the "Reference" image number 16 in all cases. When you are done, you should have 5 images representing a combine of 2, 4, 8, 16, and 32 images. If you look at these images, it should be quite obvious how the noise decreases as you increase the number of images.

Analyzing the Images in Python

We will use *Anaconda Python*, which is available at https://www.continuum.io/downloads, to do some further analysis and make some plots. Once installed, there are several options for running python. We will use the Jupyter QTConsole that comes with the Anaconda distribution. You can get a detailed introduction to the software by doing the Python Tutorial from the course web page. The key to using Python for image processing is to understand that images are simply arrays of numbers. Open the Jupyter QTConsole window and change directory to the folder that your files are in. You can read in one of your images by the same process described in the tutorial. You will need both the image and the header information in the file. You can get those using the commands:

```
from astropy.io import fits
hdu = fits.open('<filename>')
header = hdu[0].header
img = hdu[0].data
```

where <filename> is the name of your file (with its .fit extension). The filename has to appear in quotes. The first command imports the fits library. The second command opens the file and returns a "handle" in the variable `hdu`. All FITS images contain a header that lists important information, and this information will be returned by the third command as a list of strings in the variable name `header`. The fourth command will read the image and place it in a variable called `img`. You access the image by referring to this variable name. Before we go further, we have to correct the image for an offset applied by *MaxIm DL*. To keep values from going negative, *MaxIm DL* adds an offset to all pixel values (called the PEDESTAL). To see the value of PEDESTAL for your image, type

```
header
```

and look through the output for a line starting with the word PEDESTAL. It should have a value of −100. You can access the PEDESTAL value directly by header['pedestal']. Let's add this (negative) value to the image, by typing

```
img += header['pedestal']
```

To display the image, type

```
imshow(img,'gray')
```

You will probably see a nearly blank screen, because the brightness scale is too broad. You can display it with a narrower range by typing something like

```
imshow(clip(img,100,3000),'gray')
```

which clips the image to make the brightness range from 100 to 3000. Make the window larger to see it at full resolution. Retype this command, adjusting the clip range (the low value of 100 and high value of 3000 above) as you wish, until you can see the background well.

We are going to look at the statistics of a small region of the background of the image that has no stars or other obvious objects. Use the Python figure window's zoom function to zoom in to such a region, and when you have a nice clean background, type:

```
x = map(int,axis())
```

This command reads the axis limits of the zoomed portion of your image into a variable called `x`. Type `x` to list its values. We will use these coordinates as arguments to extract the region of the image outlined by your zoom:

```
sample = img[x[3]:x[2],x[0]:xp[1]]
```

so that the array named `sample` will now contain this extracted portion of the image. Display this extracted region by typing

```
imshow(sample,'gray')
```

Note that this will rescale the brightness of the sub-image. To scale the brightness as before, include the low and high values you used in your previous `imshow()` command.

Signal to Noise

You should see a noisy looking rectangle or square showing your extracted region. Now we will measure the "signal to noise ratio" in this region by using some statistics commands, the MEAN and STD commands. The MEAN command returns the *mean*, or average, of a distribution of values. The STD command returns the *standard deviation*, often expressed using the greek letter sigma (σ). We will define the *signal to noise ratio* as the *mean* divided by the *standard deviation*. First, print the value of the mean and standard deviation by

```
mean(sample)
std(sample)
```

Now calculate the signal to noise ratio and save it in a new variable, `snr`, by the command:

```
snr = mean(sample)/std(sample)
```

Automating the Process

We now want to do this same sequence of commands for each combined image that you took, and plot the result. This would be tedious to do by hand, so let's write a small program to do it. We will do this as a class, and each of you can use the program to create your plot. We will plot the signal to noise ratio for our images as a function of total observing time that went into the images. You will see that the signal to noise ratio increases along some curve. This curve can be well approximated by a parabola of the form $SNR^2 = at$, where t is the exposure time, and a is a constant related to the number of photons/s arriving at the detector. This dependence indicates that the signal to noise ratio increases as the square-root of the time. Try plotting your measurements as SNR^2 vs. t to see the linear dependence. **Make a single, 6-panel image consisting of your chosen region extracted from each of the 6 images and include it in your log book and in your written report. Make a printout of your plot and include it in your log book and written report.**

Conclusion

You should easily see that combining images improves the signal to noise ratio, allowing fainter stars or details of objects to be seen. Quantitatively, the signal to noise ratio increases as the square root of the observing time (gaussian statistics) or as the square root of the number of photons measured (Poisson statistics—also called counting statistics, or photon statistics). Since the rate of incoming photons is presumably constant, the number of photons will increase linearly with time, so in this case gaussian and photon statistics give the same result—signal to noise ratio increases as the square root of the observing time. You can increase observing time by taking longer exposures, or by taking a larger number of short exposures and combining them. However, due to image rotation and telescope motion, long exposures result in star trails unless you have a very accurate telescope tracking system. In our case, we will typically take exposures of no more than 20 s. For a really good image (low signal to noise), we might want a total exposure of 1 hour, which would require 180 images! Do not be afraid to take lots of images if you want good signal to noise.

Bulletized Synopsis

Purpose: Learn about CCD cameras and calibration (bias and dark)

- Take a bias image and measure its average value.
- Take a series of dark frame images at five different temperatures and note the change in dark level with temperature.
- Choose a star around 6th magnitude, point the telescope at the star, focus, and verify pointing. Write down star name and particulars in your log book.

- Set the temperature to as cold as possible (e.g. –30 C) and take a sequence of 32 images of 20 s exposure time.
- As a class, take a series of 20 bias frames and 20 dark frames (of 20 s duration) at the same temperature.
- In *MaxIm DL*, apply these calibration frames to your mages.
- Combine your calibrated images in a series of 2, 4, 8, 16, and 32 images.
- Use Python to explore one of your images, to learn that an image is nothing more than an array of numbers. Learn to open a fits file, display an image, and select a portion of an image.
- Use Python to measure an area of your images (the same area in each image) and make a plot of signal-to-noise ratio (SNR) vs. exposure time. Compare with the expected sqrt(time) dependence.