ASTR:4850 - An Introduction to Image Processing with Python

Introduction

In this course, we will write our own python programs (as opposed to using a commercial software) to analyze astronomical data. Python is an interactive and extensible programming language. It has extensions to load images from FITS files, process images using compact code without loops, and display images on the screen. In this first assignment, you will become familiar with the basics of python and some of the particulars about handling images in python. You'll be using python heavily during the semester, so it would be good to install it on a computer that you have access to outside of class (like a laptop if you own one).

I recommend installing a Python distribution called Anaconda, because it is designed for data scientists and most of the packages for scientific analysis are pre-installed in Anaconda. It also provides an easy interface to manage virtual environment and packages. To install, go to `https://www.anaconda.com/download/` and download the latest Python 3 version.

I recommend using the Integrated Development Environment (IDE) called Jupyter to compose, edit, and run your code, because it lets you easily combine Markdown text and executable Python source code on one canvas called a notebook. Jupyter is included in Anaconda, to launch it, simply open Anaconda-Navigator and click on Jupyter. By default, Jupyter launches in your web browser. If you don't like using a browser, you can check out Virtual Studio Code.

To get started, download the zip file to obtain the example FITS images and the tutorial Jupyter notebook, and then unzip/extract the files in your chosen work directory. Use Jupyter to open the tutorial notebook (the .ipynb file in the unzipped folder).

Here is a todo list to set up this lab:

- Download **python_intro.zip** and extract its content in your work directory
- Install Anaconda3
- Launch Jupyter and navigate to your working folder
- Open the example notebook `python_intro.ipynb`

1 Plotting an image from a FITS file

In this section, we will go through the first code block.

The first line is a comment about what the program does. It is good practice to liberally comment your code.

First, we import a few packages, which are pre-written modules to do various tasks. numpy, or 'Numerical Python' is a package to define mathematical constructions like matrices (or arrays) and make it easy to handle them in python. matplotlib is a scientific plotting package. astropy.io.fits is a package to read and write FITS files.

Then, we open a FITS file with **astropy.io.fits**. In the initial import lines, we have imported the FITS I/O package "as pyfits". This allows us to use the shorthand "pyfits" to refer to the package. The program currently opens one of the FITS files you downloaded. If you are running python in the same directory as your data file, you need to input only the file name. Otherwise, you will need to add the file's full path.

Next, we copy the image data array from the FITS file to a variable called img. img is actually a numpy 2-dimensional array and later on we'll discuss how to do math operations on numpy arrays.

Finally, we plot the image using matplotlib.pyplot. Back in the import section, we imported the plotting package matplotlib.pyplot "as plt". This allows us to use the shorthand "plt" to refer to the plotting package. All four lines call routines in the matplotlib.pyplot package using plt as shorthand. The first line sets the plotting to interactive mode; this just means that the program continues on after making the plot rather than stopping for the user to look at the plot and then close it. The second line sets up the color map. Since the camera is black and white, we use a gray scale map to make realistic looking images. The third line uses the "imshow" routine to make a plot of the image data using the selected color map. The fourth line actually puts the plot up on the screen.

Now play with the notebook:

- run this block of code by pressing Shift + Enter while the cursor is inside the code block. It should regenerate the plot.
- modify the color map (colmap) and the contrast (vmin and vmax) parameters, rerun the code until you get a picture that pleases your eyes.
- learn to use Markdown textboxes to provide formatted text. Move your cursor just slightly above a code block and select to add a Markdown textbox. To edit an existing Markdown textbox, double click it. The basic syntax of Markdown is on this webpage: `https://www.markdownguide.org/basic-syntax/`.
- For more plotting examples, read this notebook example.

2 FITS header

FITS is a "self-documenting" file format. In addition to the image data, the file contains a header that gives the 'meta-data' describing the image parameters and how and when the image was obtained. These were written by the camera control software when the file was created. The meta-data consists of a keyword name, a value, and an optional comment. For example, the NAXIS keyword gives the number of axes, it should be 2. The NAXIS1 and NAXIS2 keywords give the number of pixels along the two axes of the image. The meta-data in FITS files provides a nice way to check that you are working with the correct data files.

The next section of the notebook prints out the FITS header of the 0th extension (h[0].header), and then uses the header saved in the 1st extension to work out the celestial coordinates of the central pixel, so that we know where the image is on the sky. It uses the astropy.wcs package to do the coordinate transformation from pixel coordinates to sky coordinates.

Now using the list of keywords in the header, answer the following questions and write your answers in a Markdown textbox in the notebook.

- Which telescope and instrument took the image? (where is the telescope and how large is it?)
- How long was the exposure time?
- Which filter was used (and Google its central wavelength)? Is it in optical or near-IR?
- Finally, use `http://ned.ipac.caltech.edu` to find out the common name of the object.

3 Subtracting images

This code block reads in two other images of the same object, extracts the image data, subtracts the background, calculates a new array which is the difference in the two images, and then plots all three images.

Difference imaging is incredibly useful in astronomical observations. Describe studies of two types of celestial objects where difference imaging is beneficial.
Write your answers in a markdown textbox in Jupyter Notebook.

4 Plotting a histogram of pixel values

This section calculates some statistics and plot a histogram of the pixel values in our images. The first parts should be familiar. After setting up figure 1 with plt.figure(1), we clear the figure using plt.clf(). This resets anything that was done to the plotting window.

The numpy/matplotlib routines to make histograms and calculate statistics need to have one-dimensional arrays as input. We find the size of the 2-d array img with img.shape routine, which returns two values in a 'tuple'. Python tuples are sets of values separated by commas and sometime enclosed in parenthesis. On the line, nx gets set to the first value in the tuple and ny gets set to the second value. In the next line, we make a new 1-d array imgh. It has the same values as the 2-d image array img, but arranged like ducks in a row.

Following this are a few lines to calculate statistics of the pixel values. min, max, and mean should be self explanatory. The standard deviation (std) is a measure of the fluctuations of the data around the average value. It is the square root of the average of the squared deviations from the mean, i.e., std = sqrt(mean((x-mean(x))**2)).

Then, we plot the histogram. A histogram is a graphical representation of the distribution of a set of values. In this case, the values are the intensity readings from the CCD for each pixel. If you arrange these in a 2-d array, you get an image. A histogram discards the spatial information and only shows the distribution of the values themselves, in particular the frequency with which each value (or range of values) occurs. The matplotlib routine **hist** plots histograms. The first argument is a 1-d array of values, the other arguments set the plot parameters. The most crucial of these is 'bins' which sets the number of bins in the histogram. By default, the bins will evenly spaced between the lowest and highest input values.

In calculating the statistics of a data set, one often prefers to discard outliers, e.g. pixels that have high values because they are saturated or 'hot' or pixels that have low values because they are damaged. Examine the histogram of your pixel values. Are there outliers? If you don't have any hot pixels or dead pixels (with very low values), then you have a good detector.

Following the first histogram, we set an allowed range for 'good' pixels values (between **plow** and **phi**) and then make a new array, **imghcut**, keeping only the values in that range. A new histogram is then made and its statistics are printed.

Adjust the values for **plow** and **phi** accordingly. Re-run the code block and observe how the statistics of the pixel values change when keeping only the 'good' pixels.

5 Now Try Python for yourself

- Write a code block to make an array of the integers from 1 to 10, find their squares, and then plot the square versus the value.
- Write a code block to calculate the sum of a 9x9 box pixels centered on pixel (100,100) in the difference image.
- Write a code block to overplot the mean, the median, and the mean ± stddev of the pixel values on the histogram as vertical lines of different styles. Adjust the plotting range so that the lines are clearly separated. This plot gives a nice graphical illustration of how the standard deviation is a measure of the fluctuations of the data around the average value.

6 Submission

The preferred format for this assignment is Jupyter Notebook. When you are done and before quitting Jupyter, save the notebook file (.ipynb) as an HTML file (.html). Submit the HTML file on ICON, not the .ipynb file.

7 Resources for learning Python

The resources below are a good place to start. If you want to do something specific in python, a good first step is to search on the internet, e.g. try Google 'python median'. If any of the links are broken, try Google the keywords.

- Astropy affiliated packages: `https://www.astropy.org/affiliated/`
- NumPy tutorial: `http://cs231n.github.io/python-numpy-tutorial/`
- Matplotlib tutorial: `http://matplotlib.org/users/pyplot_tutorial.html`
- PyFITS documentation: `https://docs.astropy.org/en/stable/io/fits/`
- Code Academy Python tutorial: `http://www.codecademy.com/tracks/python`
- Question and answer site for programming: `http://stackoverflow.com`