Tyler James Metivier

Professor Whitaker

Undergrad. Research

February 26, 2017

Python Tutorial on Reading in & Manipulating Fits Images and Creating Image Masks

(with brief introduction on DS9)

Abstract:

This tutorial involves reading in an image of a galaxy and its model, reading in a previously constructed segmentation map and creating a mask, and creating a "residual" image while writing it to a new fits file. In essence, this tutorial represents how one may selectively remove unwanted objects/values from an image. A smaller tutorial on viewing these images in the program DS9 will be included at the end.

Introduction:

When galaxies merge, morphological structures known as "tidal features" may be formed. These features are mostly composed of stars. Tidal parameters are the mathematically quantified values of tidal features in a galaxy. Quantifying these tidal features can be incredibly helpful in understanding the nature of galaxies within our cosmos. With greater knowledge of these structures, we can better understand gravitational properties on a larger scale. Galactic mergers may also be a catalyst for the shutoff of star formation within galaxies. This tutorial will include the first parts of quantifying tidal features. The image and model of the galaxy used is from the OBEY survey (Observations of Bright Ellipticals at Yale). This survey includes galaxies, "selected from the Nearby Galaxies Catalog (Tully, 1988), consisting of all elliptical galaxies at a declination between -85 and +10." An elliptical galaxy is one with an ellipsoidal shape and a smooth light profile. These galaxies are practically featureless in comparison to a spiral galaxy. (A spiral galaxy being one with a spiral morphology). Elliptical galaxies also have little to no star formation occurring within. The OBEY team derived an objective tidal parameter from the galaxies within their sample. As was done by the OBEY team, this tutorial contains the steps leading towards the quantization of said features.

Because this tutorial is using nearby galaxies, there is no issue with detecting the tidal features. As one looks farther and farther away, it is harder to detect tidal features. This is due to surface brightness dimming. Surface brightness dimming is quite self-explanatory- it is the dimming of the surface brightness based on redshift (z). This redshift is in direct correlation with the distance to a galaxy. Redshift (in this situation) is the stretching of light as it travels across the cosmos. This occurs because the universe is expanding. The fact that surface brightness dimming affects all objects makes it so that a huge majority of galaxies cannot be used for analysis, only ones with a very high surface brightness. This tutorial is part of a larger project that addresses this issue. Once the tidal parameters are calculated, we can simulate high redshift galaxies (while taking into

account the cosmological surface brightness dimming given by: $(1 + z)^{-4}$). The simulated high redshift galaxies can be utilized in finding how large a telescope must be in order to detect these tidal features. This can also give us insight as to what possible future telescopes like LUVOIR can give us. LUVOIR stands for Large UV/Optical/IR Surveyor. LUVOIR is a NASA concept for a multi-wavelength space observatory to be "sent up" in the 2030's. The mirror aperture may be anywhere from 8 to 16 meters in diameter. This tutorial will explain the process of reading in images of a galaxy and creating an image mask in order to isolate the tidal features.

Tutorial:

[This tutorial is for Linux users]

Before the .py file is written containing the main components of the project, a few steps must be taken first.

Some Background

If Anaconda is not installed, please follow this link and install Python 2.7 for your machine. (2.7 is not the newest version but it is more stable with existing astronomy software)

https://www.continuum.io/downloads

Make sure you know which directory Anaconda is downloaded to. Open up a terminal and type:

```
bash ~/whereveryourfileisstored/Anaconda3-4.2.0-Linux-x86_64.sh
```

Obviously, replace "whereveryourfileisstored" with its actual location.

Note: *If you are using a 32 bit machine, make sure you install the correct file and do not put "_64" at the end of the extension (as shown above).*

Now simply let Anaconda install. With the installation of Anaconda, Astropy should also be installed.Astropy will be a vital tool in the future. Open up a terminal and type:

```
conda update astropy
```

This will ensure you are using the newest version. Now you may type:

```
import astropy
astropy.test()
```

This will run and print out any failures. Hopefully, everything will work as it should and you can move on to the next step. The program 'Emacs' will be utilized for the writing of our code in this tutorial.To install this program, type into your terminal:

```
$ sudo apt-get install emacs
```

And just like that, we can begin. The first fits images we will be using can be downloaded from this link:

http://www.astro.yale.edu/obey/cgi-bin/catalog.cgi?1852

To install the images; right click on “Reduced data frame” and “save link as” into a directory that will be most convenient. For this tutorial, the directory will be (‘/home/tyler/research/’). Do the same for “Model fit” on the web page. This will be the model. Onto the fun part.

Part I – Reading in the Model and Image

Open up the terminal and type the following:

```
emacs -nw tidalresearch.py
```

This will create a new .py (python) file within the text editor.

The -nw part simply stands for “no window.”

There should be a blank file open in front of you now.

First, we will add two lines to this document. These lines will import the needed tools from astropy that will allow us to view and manipulate FITS files.

FITS stands for “Flexible Image Transport System.” FITS images are commonly used in astronomy due to the fact that they can contain so much more than other, more common digital formats (like JPEG). FITS images can contain multi-dimensional arrays and descriptive information regarding the data.

To differentiate between the terminal and our program we’re writing, the program blocks will be colored light blue.

Now that we’ve opened a blank .py file, proceed to add:

```
from astropy.io import fits
```

Now it’s time to read in the model and the image. Add the following lines:

```
model_header = fits.open('/home/tyler/research/1852_b2_mod.fits')
```

```
model = model_header[0].data
```

The ".data" is included simply because we are telling it that we want to access the data portion of this file. As the text suggests, we just opened the model. What we called "model" is set equal to to the part of the data that we want to utilize. (denoted by model_header[0])

Now we are going to do virtually the same thing with the image.

```
image_header = fits.open('/home/tyler/research/1852_b2.fits')

image = image_header[0].data
```

You may be asking, "why did we put a [0] after header both times?" That's simply because [0] is where the desired information in this FITS image is located. Putting something else like [1] or [4] would access the first or fourth (respective) extension. But today, we're just sticking with the primary set [0].

Part II – Reading in Segmentation Map and Creating a Mask

The segmentation map utilized in this tutorial was created in Source Extractor[1]. The segmentation map was created in this program by identifying a group of connected pixels that exceed a certain threshold above the background as a detection. Any "detection" is simply given its own identification number. These detections can be stars or galaxies or anything contained within the image. If, for example, a galaxy registers as a detection (as it should in our case)- all of the pixels associated with that object have the same identification number. These values can be discovered in DS9 by hovering the pointer over the object on the screen. (More on the software later)

The specific file we will be using here can be downloaded here:

https://www.dropbox.com/s/bdum4orcgkqsq8j/1852_b2_seg.fits?dl=0

First, we are going to open the file and set it equal to seg_header:

```
seg_header = fits.open('home/tyler/research/1852_b2_seg.fits')

seg = seg_header[0].data
```

As you will likely notice, these two lines are very similar to the previous four that were added. That is simply because we are virtually doing the same thing- simply reading in the various different fits images.

[1] Source Extractor, Bertin & Arnouts 1996

Next, we are going to identify the elliptical galaxy of interest within the segmentation map. The galaxy was assigned an id number (along with all of the other objects within the map) when it was put through source extractor. This id number was found using the program DS9. A short tutorial on how to use DS9 will be included at the end of the python section. If you insist on verifying this id number yourself, you are welcome to skip to the end, read the DS9 section, and come back to this step.

The galaxy was assigned the number 229.

```
id_galaxy = 229
```

Because we've now identified the main galaxy, we must remove all of the other background sources within this image. We will do this by creating a mask.

```
selection_1 = ((seg > 0) & (seg != id_galaxy)) # the != means "is not equal to"
selection_2 = (seg == id_galaxy)
```

Within the segmentation map, any value equal to 0 corresponds to sky background noise pixels. Here, selection_1 is equal to the pixel values associated with all of the detected objects except our galaxy. And selection_2 is where seg is explicitly equal to the id_galaxy. And if you remember, id_galaxy was already defined as the where the pixel values equal 229.

This mask has the same x,y dimensions as the other images, but the sky background and elliptical galaxy have a value of 0. Any other pixel that belongs to an object is now equal to 1. We will now define this mask as equal to "seg" so that we can further manipulated the values and then apply it to our image.

```
mask = seg
mask[selection_1] = 1
mask[selection_2] = 0
```

Now we are going to make it so that the values that equal 1 in the image actually equal "NaN." (Not a number). We are doing this because we don't need these values moving forward. All we care about is the residual leftovers. When the galaxy, stars, and other objects are removed- what we have left is our tidal features. Now that the other objects pixel values are not a number, they won't accidentally be included in any sort of calculation.

```
mask_selection = (mask == 1)
masked_image = image
image[mask_selection] = float('NaN')
```

The values in the model that equal -1 will also be changed to NaN. This is because it was noticed that the pixels outside of the pixels of interest (associated with the model) had a value of -1. This can be dangerous when quantifying tidal parameters. They are converted to NaN so python doesn't perform mathematical functions with them.

```
wrongnumbers = (model == -1)
model[wrongnumbers] = float('NaN')
```

Part 3 – Creating the New Fits File

This is the final section of the python tutorial. Only two lines remain.

To create our residual, we must divide the masked image by the model and then subtract 1 from that ratio. This calculation was taken from the paper:

"THE FREQUENCY OF TIDAL FEATURES ASSOCIATED WITH NEARBY LUMINOUS ELLIPTICAL GALAXIES FROM A STATISTICALLY COMPLETE SAMPLE."

$$T_{galaxy} = \overline{\left| \frac{I_{x,y}}{M_{x,y}} - 1 \right|} \qquad (1)$$ (Tal et. al 2009)

T_{galaxy} stands for the tidal parameter of this specific galaxy. However, in this tutorial, we're not actually quantifying the tidal parameter yet. So in our case, we are just making a new FITS image that contains *only* the tidal features. $I_{x,y}$ and $M_{x,y}$ are the overall pixel values of the image and model, respectively. This equation tells us that the tidal parameters are calculated by taking the absolute mean (the bars on the side mean "take the absolute value" and the bar on the top means "take the mean').

```
image_header[0].data = abs([masked_image / model] -1)
```

Now we will simply write this to a new fits file. After we write the directory and name of our new file, we will also add a line that will always overwrite this file anytime we run our script.

```
image_header.writeto('home/tyler/research/1852_b2_res.fits', clobber = True)
```

Now you can save this file that you've written and run it. To save and exit emacs, press ctrl-x-s and then ctrl-x-z. You should be back at the terminal now after saving

NOTE: *We are no longer writing our program, we are now in the terminal*

In your terminal, type:

```
python tidalresearch.py
```

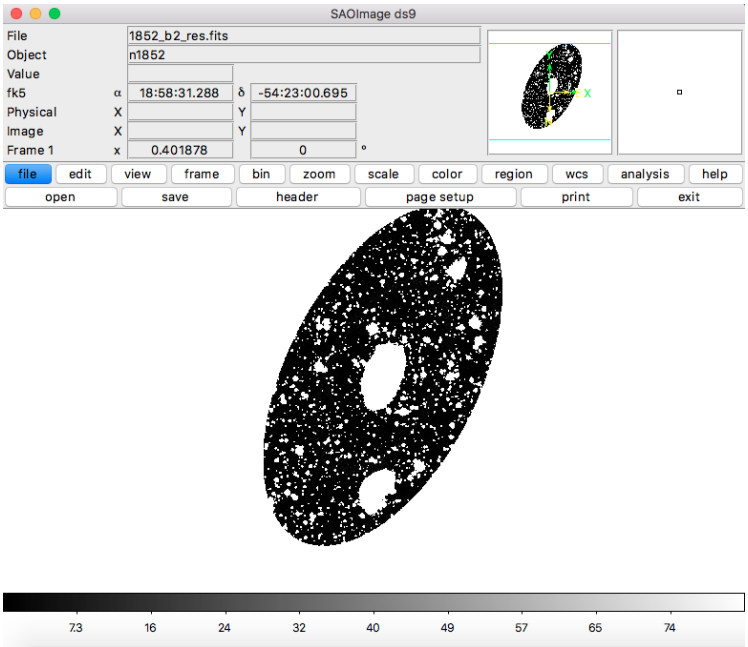
Hopefully it will run with no issues. The new image we've created can be opened and viewed with ds9. To this, type:

```
ds9 ('/home/tyler/research/1852_b2_res.fits') &
```

The "&" symbol will make it so that it will run in the background and won't clog up the terminal.

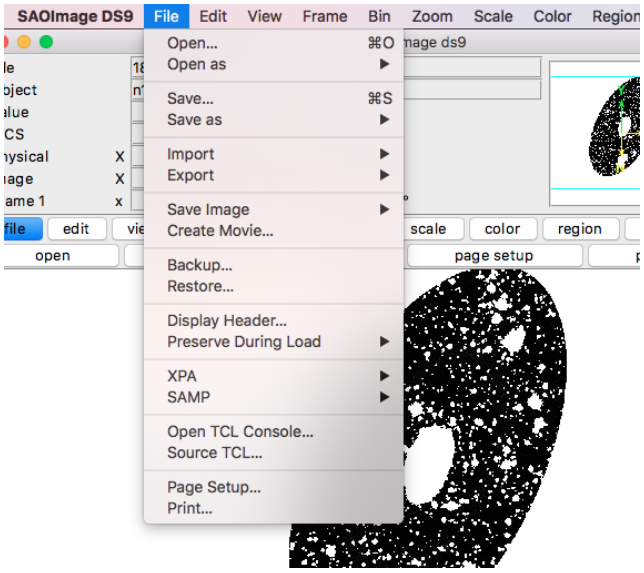
On How to Use DS9:

DS9 is a powerful tool when one wants to view a FITS file. **see page 3 for definition of FITS image*

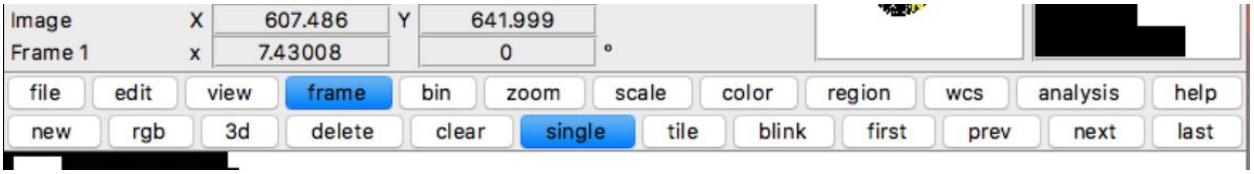


1. **Adding New Frames and Tiling Images**

   First, open the image by going to “File -> Open” or hit ctrl-o. This will give you a drop down menu with your directories.

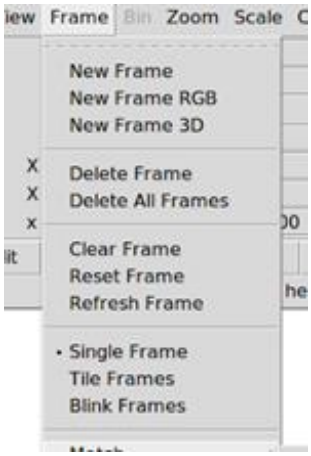


Now, to open more files, go to the middle section of buttons and select “frame -> new.”



Now the first step may be repeated for opening the second file. The image should be opened first followed by the model, segmentation map, and finally the residual. The residual is the file that we created earlier (1852_b2_res.fits) that contains ONLY the tidal features of our galaxy.

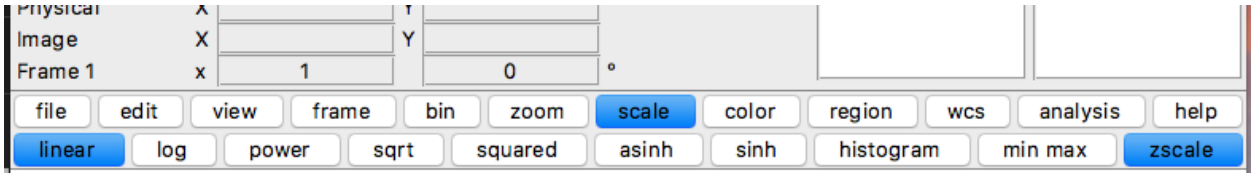
Once all of the images are loaded in, the “tab” key may be hit to flip between the images quickly. If you’d like to view all of the images at once, go to “frame -> tile.” (in the middle section)



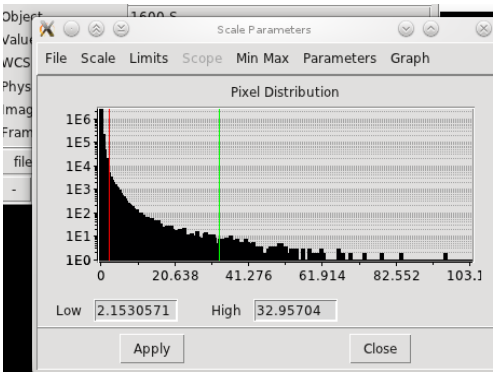
If too many frames are accidentally opened, click on the frame you want to delete and then select “Frame -> Delete Frame.” This can also be done using the middle set of buttons. Go to “frame -> delete.”

2. Scaling Images

Click the frame you’d like to scale and then select zscale.



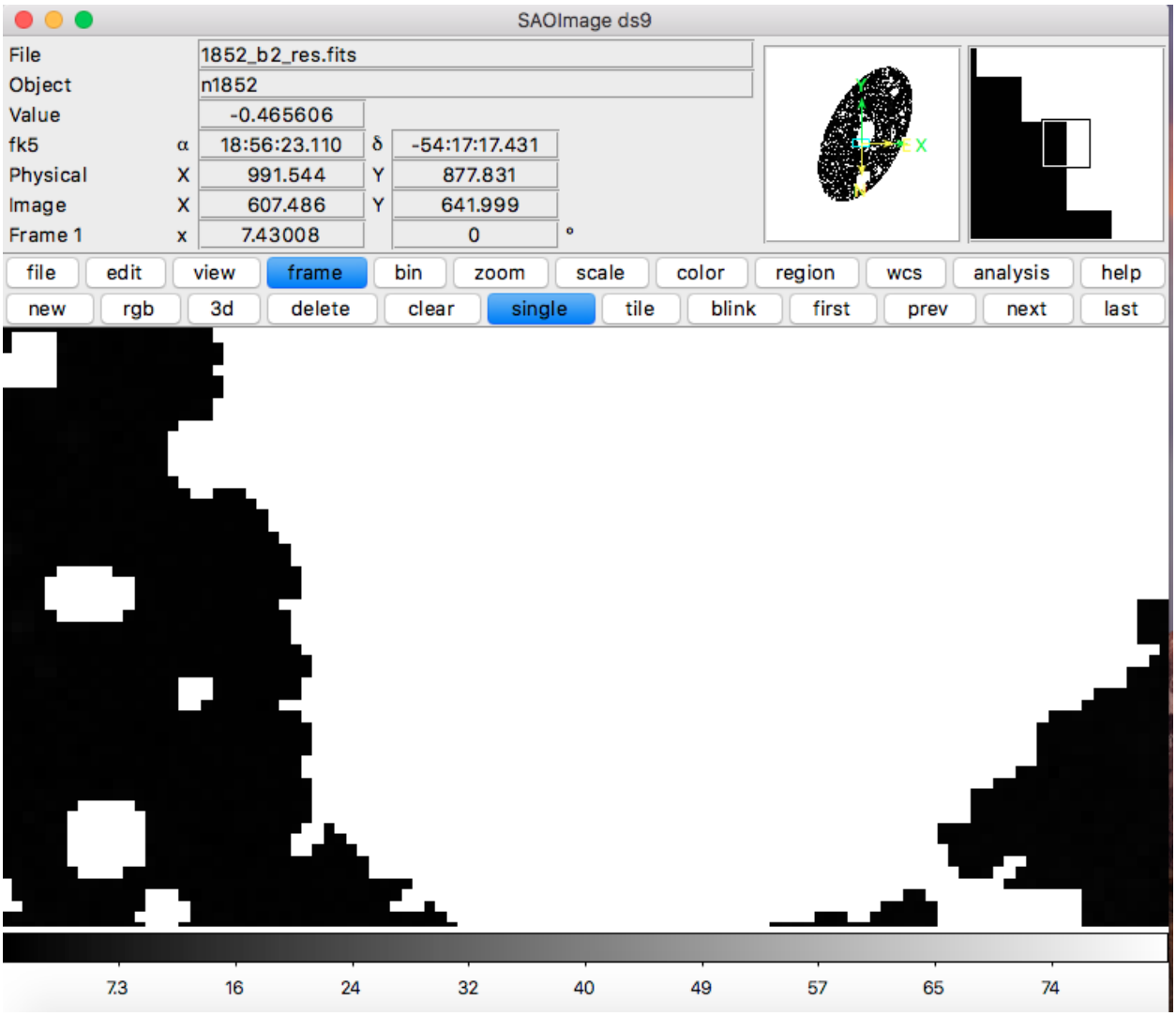
Zscale usually works very well at selecting a good contrast, but you can also set scaling parameters yourself. Go to Scale -> Scale Parameters (it is at the very bottom). Here, you may set your own values.



3. Zooming

You can zoom by selection “Zoom -> Zoom In (or Zoom Out)”. The box in the upper right corner can be used to move to different areas of the zoomed in image.

[*Zoomed in image below*]



If you'd like to align all of your images up in order to tab through them quickly (and see the same corresponding area when comparing them), select "Frame -> Match -> Frame -> Image." This will match all of the images by the level of zoom of the original frame and x,y coordinates. If they have the same world coordinates but different pixel scales, you can select WCS instead. This is a quick and easy method when one wants to cycle through the aligned images.