Using Python to Assist in Characterizing and Modeling Circumstellar Debris Disk

Marcos Martinez
Consortium for Undergraduate Research (CURE)
Los Angeles City College (LACC)
Los Angeles, USA
marosmartinezedu@gmail.com

***Abstract*—As part of an effort to characterize and model circumstellar debris disk, 376 stars are chosen as targets on which to collect largely photometric data for plotting a spectral energy distribution (SED) for every target. Python version 2.7.13 is used to match data from various ground- and space-based telescopes to its corresponding target. A formatted file containing largely photometric data is created for every target. In addition, a new Python algorithm was devised to mosaic 56 images in the flexible image transport system (FITS) format, for those spatially resolved debris systems. The mosaic of FITS files help visualize the extent, orientation, and position angle of the outer cold ring around the mature stars studied.**

***Index Terms*—Python, formatting, debris, disk, FITS, mosaic, photometry, sorting, file.**

I. Introduction

When looking up at the night sky, what we observe to be stars are often not just a star, but one with a complete planetary system and orbiting debris, that can only be faintly detected at infrared wavelengths. The debris disks of these stars can be many magnitudes brighter that any individual planet within the disk. Single belt and especially double belt circumstellar debris disks can provide insight on what is contained in a star system. For example, it is theorized that the features of mature, two debris belt star systems are impacted by the terrestrial and gaseous planets within them; i.e., the solar system. Circumstellar debris disks are being studied extensively in the last decade and two-belt debris systems have become a focus for the exoplanet search effort and it may be interesting to note that we live in one.

The project I am involved with this summer attempts to characterize and model circumstellar debris disks using realistic dust grain properties and data taken from several instruments. The project is organized into several modular components that ultimately calculate best physical model fits using Chi Square fitting routines. This method is also used in Morales et al. 2013 and Morales et al. 2016. As part of the project, the code to accomplish this is to be written in Python, which is free to use, well documented and portable across several platforms.

I am tasked with the gathering of photometric data on the targeted stars from several ground- and space-based observatories and organizing that data into a single uniformly formatted file for each star-debris system. Files contain largely photometric data as well as peripheral data in a header format; such as distance, age, spectral type and temperature. Each of these data files will be used to model the spectral energy distribution (SED) for its corresponding star-debris target. The SEDs will be used to find a "best fit" to models of many different varieties to probe for disk and planetary system architecture and composition. I am also tasked with devising an algorithm in Python that will tile flexible image transport system (FITS) images in any given number of dimensions for quick analyzation and publication purposes. Flexible image transport system files are files widely used in the field of astronomy to store telescope images with data that the operator may want to include with the observation. I will have to analyze this data myself with Python's FITS file

handling capabilities to account for each image's scaling parameters, rotation and resolved location.

II. Methods

A. Data Compilation

When given my first task of compiling data, I started with a list of 376 target stars given to me by my mentor. I am also given a list of data from different sources that shall be included in the final file. Much of this task is to make sure that all the data that corresponds to a star is matched to the correct star. There are two major methods to identify a star in the catalogs that I am working with; stars are identified by their name, which most have more than one of, or they are identified by their coordinates, which presents a degree of uncertainty. With so many inconsistencies, it may seem daunting at first to have to compile data that will be constantly analyzed, but an important principle of computer science suggests that the task should be broken down further.

Fundamentally, there is an object (the star) and attributes (the data) to be associated with that object. In computer science, an object is rather a special abstraction of a data structure with attributes and attributes are what that object "has". Because we have a situation as such, we can use object-oriented programming as a basis to begin the data crunching. Python allows its users to take advantage of its object-oriented programming features to accomplish tasks of that nature.

A class in computer science is like a blueprint in construction. An object of a class is like a house from a blueprint. The attributes of a blueprint may be spatial dimensions or features of the proposed house. Attributes for every object are not always necessarily the same value but always the same type. Python allows users to create "blueprints" or classes for just about anything, so I created a target star class whose attributes will act as place-holder variables that will be populated by the data retrieved from every catalogue of interest.

It is necessary to first create a star object for every target with an associated name or names and coordinates so that the correct data may be matched to the correct star. A series of control statements is used to accomplish matching the data. Python's built in csv module assists in extracting data held in comma separated value (CSV) files. The CSV files I did work with all held data slightly differently, therefore some values in the code were hardcoded to handle the formatting variations. Other files were organized in IPAC format, which is a format widely used in the field of astronomy and astrophysics to store and process data. Python also has IPAC format handling capabilities through the astropy module, which is a tool kit written in the Python programming language specific to applications in astronomy, that I used to extract data stored in this format.

Once all of the desired data has been collected for each star, I used Python's built-in file writing functions to create a file for each start, resulting in 376 unique files, containing peripheral data in the header of the file and photometric data form different sources in the columns. Since the files I made would be processed by someone else, I elected to write the files in IPAC format. Clear benefits of IPAC format are that the files are human-readable as well as easy to process my machine, as shown (Figure 2.1). This format may not be as widely used as it should be, even within the field of astronomy or astrophysics but it is a standard established nonetheless. Electing to use such a standard is an attempt to further standardize the way data is stored and processed in astronomy.

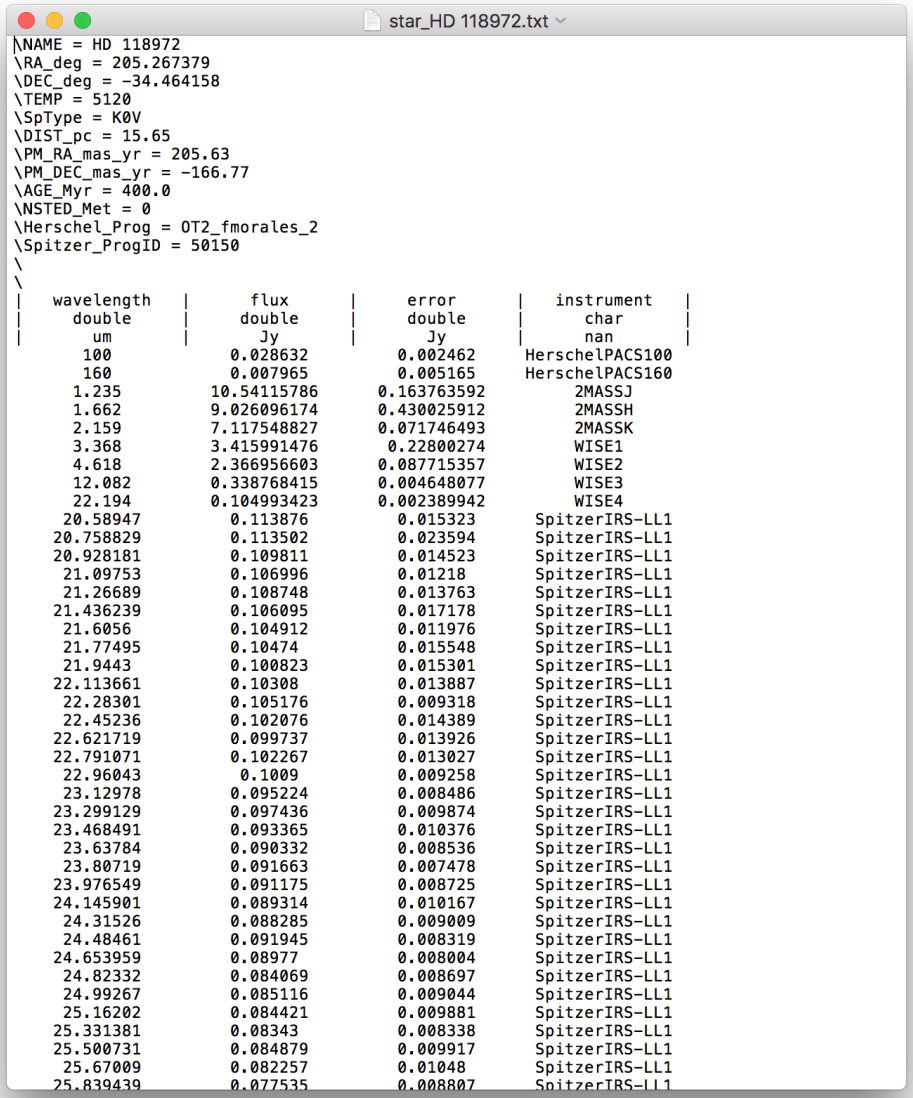

Fig. 2.1. Example of file produced. As shown, peripheral data is placed in the header and the main data to be processed for SED modeling is organized in the columns.

B. Mosaicking

When given the secondary task of mosaicking FITS files, I first started with an example FITS file to test Python's capabilities to manipulate and view it. Fortunately, Python has a tool kit to manipulate FITS file data in the astropy module that may be used to view the FITS image with the matplotlib module, which is a module in Python that is widely used for graphics.

To begin, I started by using the SAOImage DS9 (DS9) FITS file viewing application to be able to manipulate FITS images by hand. After making the FITS image look presentable, I attempt to replicate the same presentable version of the same FITS image in Python. This process went through several iterations until I was comfortable changing several parameters in the Python program to be certain of how it would affect the image.

Once I had familiarized myself with viewing the FITS image, I could then begin working with the data included in the file along with the image. The image itself is a two-dimensional matrix whose units represent pixels holding energy values. The value contained in the pixel represents a color on a color scale that is defined by the user and this allows the image to be viewed. The data also included a conversion between a pixel and a distance in arc seconds as well as an angle of rotation if the image is not oriented with north being up, east being left. In Figure 2.2, we can see an early iteration of my manipulation to the first example FITS file. Note that the image has an x-axis and a y-axis where both are in units of pixels. The reason for the axes is that matplotlib is largely used for traditional graphing in Euclidean space. Since our image data is two-dimensional matrix with values associates in its coordinates, matplotlib can plot the data in Euclidean space.

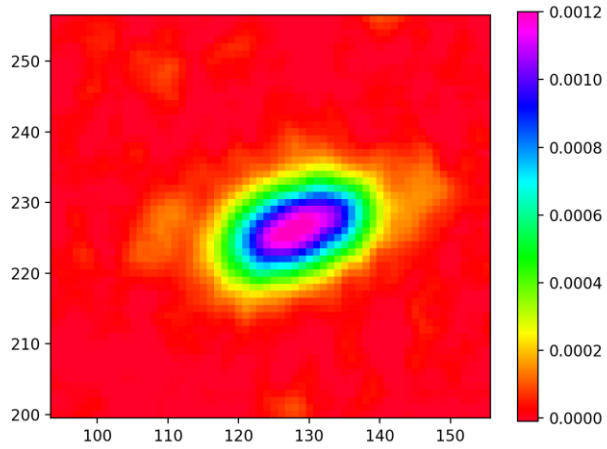

Figure 2.2. The example FITS file, which was used to become familiar with Python's FITS file handling capabilities.

Now that plotting the image has become more or less trivial, it becomes time to devise an algorithm to tile to fits files in any desired dimension. Python's matplotlib simplifies the process by providing a grid object called gridspec in its number of features. The gridspec feature allows for individual images to be plotted within the grid, organized like tiles on a wall. At first, the same image is tiled repeatedly, but then I am given a directory containing about a thousand FITS files. Each FITS files contains a modification to the an image of a particular star. Each star will have 5 main modification. In turn, each modification will represent a column in the mosaic and each row will represent a different star. At this point, the star class is reintroduced. After all the unique stars have been examined by another algorithm I made for the sole purpose of identifying unique stars, the FITS files are contained within the star object. Ultimately, there are 55 stars with all 5 before mentioned modifications. Those 55 stars are then tiled and revised. Figures 2.3 through 2.6 are to demonstrate the differences in handmade mosaics from [Morales 2013] and those made by the algorithm at different stages of their development.

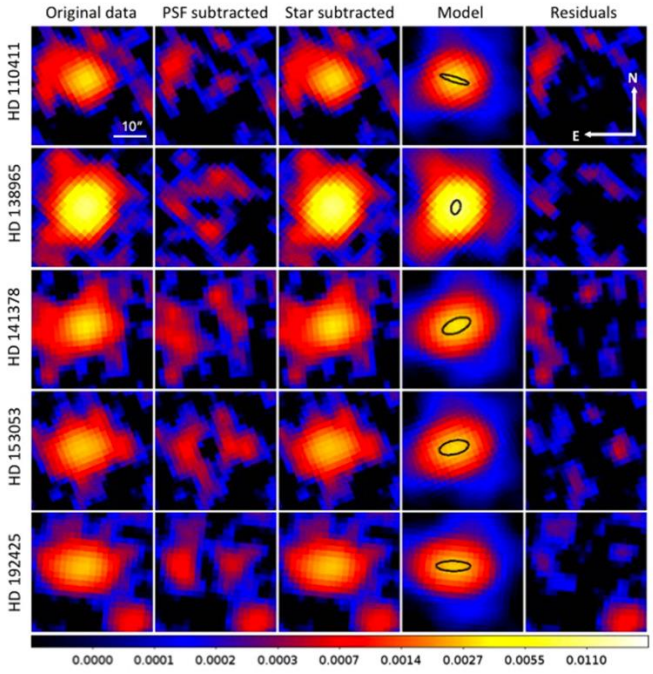

Figure 2.3. A handmade mosaic of FITS images made using DS9.

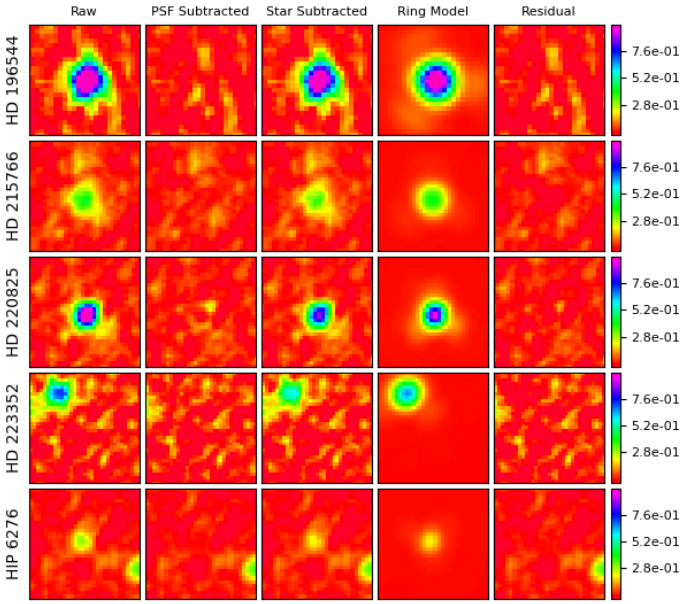

Figure 2.4. An early mosaic produced by the algorithm.

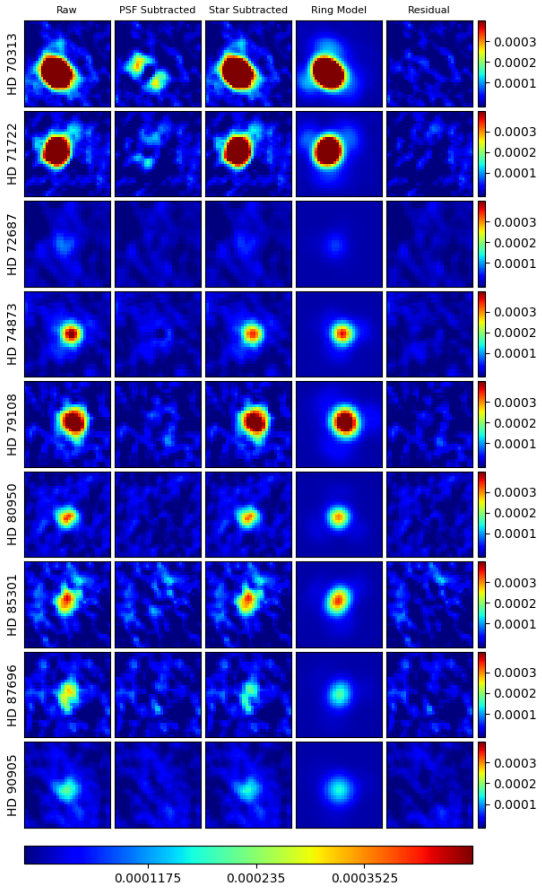

Figure 2.5. A revision of the mosaic. Note the change in color scale and color bar at the bottom of the figure.

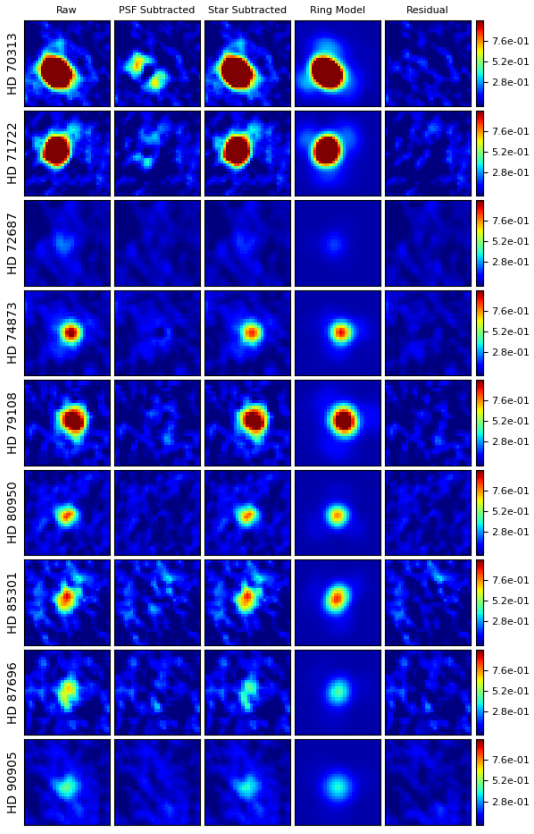

Figure 2.6. A revision of the mosaic. Note the color bars' ticks in scientific notation.

III. RESULTS

As a result of my work, 376 were produced for SED modeling that will assist in resolving more debris disks around nearby, bright stars. The code is made deliverable and is general enough to be used on other targets with minimal changes to the code. As more data is collected by future missions, my code may be used to compile new data with existing data in hopes of resolving as many debris disks as possible.

Also, the tiling algorithm I created may be used for publication purposes as well as to be able to analyze many FITS images at once. The code is concise and may be modified to meet the needs of anyone using it. Changes or improvements may still be made to the algorithm but as it stands, it is a useful tool that is generalized enough to meet different needs.

ACKNOWLEDGMENT

I would like to thank my mentors Farisa Morales and Geoffrey Bryden for their excellent guidance and inspiration. I would like to thank Paul McCudden and James Somers for their dedication to the CURE program and their efforts to ensure the success of its students. I would like to thank my group members Justin Bracks and Jonathan Acuna as well as Marlyd Mejia and Alessandra Capotosto for their contributions.

This work is supported by NSF grant #1460538 to Los Angeles City College.

REFERENCES

[1] F. Y. Morales, G. Bryden, M. W. Werner, and K. R. Stapelfeldt, "*HERSCHEL*-RESOLVED OUTER BELTS OF TWO-BELT DEBRIS DISKS AROUND A-TYPE STARS: HD 70313, HD 71722, HD 159492, AND F-TYPE: HD 104860"

[2] [2] F. Y. Morales, G. Bryden, M. W. Werner, and K. R. Stapelfeldt, "*HERSCHEL-RESOLVED OUTER BELTS OF TWO-BELT DEBRIS DISKS—EVIDENCE OF ICY GRAINS*"