**Summer 2020 UROP Summary**

*Dahlia Dry*

*8.28.2020*

1. **Abstract**

Comae are the nebulous structures surrounding comets consisting of ice and dust that form due to sublimation as comets pass close to the sun. Cometary nuclei are often obscured by these comae, making it difficult to determine where exactly the nucleus is. Because comets and comet-like objects such as Centaurs are so small and far away, accuracy is needed on the scale of milli-arcseconds (less than 1 pixel in a typical image), causing even the smallest differences in centroided location of the nucleus have significant consequences on the accuracy of astrometric calculations and ephemerides.

The process of deriving the position of the cometary nucleus in an image involves fitting a circular gaussian point spread function to the image signal in order to obtain sub-pixel precision. However, this process does not take into account the asymmetric nature of cometary comae, and literature suggests that signal from the coma drops off at a rate proportional to 1/r from the nucleus. This project aims to apply this finding to the creation of a new model of cometary comae that can more accurately and consistently determine the true location of the comet nucleus. In doing so, the goal is to potentially improve the accuracy of astrometric calculations of comets and Centaurs as well as their associated ephemerides, which can allow for better prediction of the stellar occultations of these bodies.

1. **Project Overview**

I very much enjoyed the opportunity to work on this project throughout the summer; it not only served as a great way for me to further explore my passion for astronomy but also allowed me to really work on improving technical skills that will prove valuable to me in school and my later career. This project required intensive work in both python, a language I was already comfortable with, and mathematica, a language which I had never worked in before. Because of this, it was a really interesting challenge to start and end by coding in python while spending an extensive stretch in the middle of the project really struggling to get the hang of mathematica, but ultimately coming out of it feeling confident that I could apply this new knowledge to other projects and work in mathematica in the future if the task demanded it. This project also involved a lot of math, from working with multivariable optimization to interpolating field distortion patterns and ephemerides, and it was a great way for me to apply a lot of mathematical knowledge that I had very recently gained in my coursework from the 2019-2020 academic year. One aspect of research I enjoy is that, in a way, it feels like tackling one small challenge after another in pursuit of a larger goal, but if I had to pinpoint the greatest challenge I faced during my work this summer it would without a doubt be getting started with mathematica. It was very difficult to spend hour after hour trying to resolve errors without really making much progress at all, especially when oftentimes I would eventually discover that the problem was rooted in simple syntax errors that were very easily made when I made the mistake of assuming that the simplicity of python carried over to this new language. However, eventually succeeding in building the model in mathematica was a great validation that I was learning from these errors and I believe I will certainly be better off from that struggle in the long run. It also helped that I found the premise of this project really fascinating from the get-go; I’ve always really loved creating plots and models, and being able to compare the models I built to the vast body of data in the pipeline was a really interesting way to spend the summer that leaves me excited about the research I might do in the future.

1. **Code Documentation**

* model.py: Synthesizes all the separate steps required to centroid an object in an image and apply the combined skirt model. Uses the wolfram.client python library to wrap mathematica notebooks for the skirt model, calculating ra/dec, and interpolating an ephemeris to get ra/dec offsets, so that the offsets for an entire night of images can be calculated by running this script once. Outputs a to directory named by the user (eg. Output) with the following file structure:
  + Output
    - Date
      * Frame (for frame in Date)
        + input.csv: dataframe containing the (x,y) coordinates graphically selected by the user using the model script which correspond to the three calibration stars and the object
        + plots.png: an image with 4 subplots showing, clockwise from top left: image, residual plot, contour plots, 3d model/data mesh
        + psf\_data.csv: mathematica output of the image data, with columns corresponding to (x,y,value)
        + psf\_model.csv:mathematica output of the model data, with columns corresponding to (x,y,value)
        + psf\_resids.csv:mathematica output of model residuals, flattened 1D array of values
        + psf\_xsection.csv: mathematica output of model cross section, flattened 1D array of values
      * coords.csv: dataframe indexed by frame number with columns (r,rb) corresponding to the associated r and rb file paths, (x0,y0) corresponding to the mathematica model centroid, (x0\_adj,y0\_adj) corresponding to the adjusted coordinates from interpolating the user-specified field distortion pattern, (ra,dec) corresponding to the right ascension and declination in radians calculated from the model centroid, and (ra\_offset,dec\_offset) corresponding to the right ascension and declination offsets from JPL calculating the user-specified ephemeris (.eph) file.
* utilities.py: a lengthy set of functions that I’ve developed over the past year that help with various things relating to the pipeline, like grabbing specific sets of data or clicking on a star in an image to find its associated line in the rb file
* PSF\_fitting\_v7.nb: latest version of the combined skirt fit, in mathematica notebook form. This functionality of this file is fully encapsulated within model.py