Colibri Transition Document

Tristan Mills (tmills8@uwo.ca)

August 20, 2019

All of my files are located in `/mnt/data/tristan_data/PycharmProjects/Colibri` on Mango and `/home/tmills24/PycharmProjects/Colibri` on student327 (the computer downstairs) unless otherwise specified. For more information on the pipeline's execution see Emily's paper (Pass et al. 2018) and her transition document.

1 Pipeline Update

I made a number of updates to the Colibri data pipeline. Notable changes are 1) adding fixed field functionality, 2) changing the file format, and 3) tweaking the data archival. Notes on each of these changes are listed below, as well as notes on input/output and file structure.

1.1 Fixed Field Changes

The pipeline has been optimized for a fixed field. The pipeline now only performs star finding when looking at a new field during a single night, instead of performing star finding for each data file. This improves runtime and still allows for non-perfect positioning of the telescopes.

After the initial star search the positions of each star are saved in the current date folder as `fieldname_pos.npy`. Note that a star's number may be different between telescopes. Care should be taken during analysis of occultations to make sure stars are compared to themselves across the three telescopes and not different stars.

Additionally, a drift check is still performed. If there is not any significant drift, then the pipeline assumes the positions of stars are near stationary and only performs centroid refinements. If there is drift, the motion of the stars is taken into account.

Astrometry.net support has been removed. Field data is contained in the regions.p file that outputs from Richard and Jill's field finding algorithm. The reason Astrometry was used in the first place was because the cameras Emily tested on weren't positioned at the same point, and it was used to check for the overlap in the fields. Since our three telescopes will all be looking at the same field this step is no longer necessary. For the purposes of analysis, we know which stars are which thanks to the field finding algorithm and can be found on Astrometry only as needed.

1.2 Fits File Format

The original pipeline reads in .vid files, a custom format created by the meteor group. For the update I also added functionality for .fits files, however this introduces a lot of overhead through file I/O. This change was implemented while keeping the previous format intact. The only difference is in the `getSize()` and `importFrames()` methods. (I made versions for the fits file format named `getSizeFITS()` and `importFramesFITS()`.) If the file format changes again it will be an easy change, as only these methods will need editing, along with how the pipeline is called. For example, right now the pipeline runs with a list of file names contained in a single folder, where each file is a frame. If a new file format collapses multiple frames into a single file then the way the pipeline should be called on a file by file basis, rather than folder by folder with a list of file names.

1.3 Data Archival

The pipeline currently saves 5 seconds of data on either side of an occultation event, or up to the ends of the time series if it is near the beginning or end. The relevant file and star name are saved

in the filename of the .npy file. A metadata file is also saved at the same time containing an array of the exact time of each frame. The pipeline does not currently delete processed data without occulltations.

1.4 Input/Output

The pipeline runs for a single time series at a time. The location of this time series (or the location of each frame in the case of fits files) is passed as one of the arguments for the `main()` function. The original pipeline also split a single time series in half to keep the ram usage down. This split has been removed now that we have control over the length of the time series. Testing should be performed to determine how long times series can be for optimal pipeline execution.

The original pipeline required a flat and dark image, because the field changed over the course of a night. By using a fixed field we no longer need to use flats or darks, which speeds up execution slightly and reduces the number of files needed to run.

Another input when calling the `main()` function is the field name. The field name is used for star positions, and saving occultation events. The names for the various fields obtained in the field finding algorithm can be as simple as unique numbers. No files pertaining to the fields are needed for execution.

When an occultation is identified the surrounding 5 seconds of data are saved in a date folder as an .npy file. The precise timing of each frame is also saved as an .npy file. The occultations can then be easily explored in these files. Note that the pipeline is designed to run separately for each telescope, this is particularly important when saving files to avoid overwriting. Analysis of occultations is to be performed separately.

1.5 File Structure

In the working directory the pipeline looks through the fits folder for time series. Using fits files, each folder under the fits folder represents a single time series to be searched through. The name of the fits folder can be changed so long as the directory variable when calling the pipeline is changed accordingly.

The pipeline will create a folder named using the current date. In this folder positional data and detected occultations are saved (along with the occultation's time data).

1.6 Misc. Changes

A check was added such that dips in the light curve below %40 are flagged as occultations. Dips below %40 are geometrical (rather than fresnel) and are worthy of analysis. Note that dips due to noise will not pass this check since they are found after the `MexicanHat` convolution. This was suggested by Emily quite a while ago.

2 Next Steps for the Pipeline

2.1 Decide on a File Format

Mike is working on ways to save a time series as a single file as output from the cameras. When this is achieved the `getSize()` and `importFrames()` methods will need to be changed to work with this new file type. The `getSize()` method should return the width, height, number of frames, and a time list containing the exact time of each frame. The `importFrames()` method should return an array of data containing the specified number of frames, starting from a specified start point. By changing only these methods, and the way the pipeline is called, no other changes are needed to the `main()` method.

2.2 Optimization

Aside from basic computational optimization there are a few ways that the pipeline will need to be optimized.

First, the detection thresholds can be refined further to select the stars most useful to our investigation. The pipeline currently uses the original values chosen by Emily. With working telescopes these values should be investigated again and refined.

Another major step will be optimizing the kernel set. This should be done using Emily's `fresnelModeler.py`, such that the kernels cover the entire parameter space and minimize the number of false positives and maximize true positives. See Emily's paper (Pass et al. 2018) for a similar investigation. Note that large occultations will be flagged by the %40 dip detection, and thus we don't need many kernels for larger objects.

I am not well versed in parallel processing, but there may be work to be done here as well. The parallel processing should be changed from `threaded` to `cored`, depending on the computer setup at Elginfield. There may also be further opportunities to make use of parallel processing in the pipeline that are not currently implemented.

2.3 Rolling Capture

The pipeline does not currently handle rolling capture. However the exact time for each star can be found using the start of each exposure, and the y-position of each star. This change should be made to the time list that gets saved upon finding an occultation (simply add the time correction on a case by case basis for a given star). The rolling shutter only matters when comparing light curves during analysis, so there's no point editing the pipeline to handle varying time lists.

3 Sensitivity Estimates

In addition to pipeline updates I performed sensitivity estimates for the Colibri simulation. These estimates can all be found in the `photometry_main.py` file, along with various plots and figures.

3.1 Airmass Estimates

Using the first light images I determined the approximate airmass extinction. Richards report makes the assumption that for every unit of airmass we lose 0.1 magnitudes of brightness in our stars. My calculations push this number up to a loss of 0.33 +/- 0.09 magnitudes per unit airmass.

The six fields were all observed at different times of night with different cloud conditions. However, each pair of fields (taken chronologically) appear to have similar conditions. Thus, I was able to compare the magnitude corrections of three pairs of fields to obtain the above value.

3.2 Limiting Magnitude

By finding the calibrated magnitudes of our 25ms images and plotting against the stars' signal-to-noise ratios, I found that the limiting magnitude (near zenith) is 12.40 +/- 0.27.

3.3 Distortion

To account for severe distortion found at the corners of our images, priority was given to the centres of fields when field finding. The central third of the image has no distortion effects, while the far corners can lose as much as 0.7 magnitudes when measured using apertures. See `photometry_main.py` for more details.