Colibri Main Pipeline

Rachel A. Brown

March 2022

Contents

This document describes the main data reduction pipeline for the Colibri telescope array. The main pipeline was originally written by Emily Pass in 2018. It has since been modified by Tristan Mills (adding .fits capability) and myself. It also includes functions written by Mike Mazur to add .rcd file handling (2021). A basic overview of the pipeline flow is shown in the below flowchart:

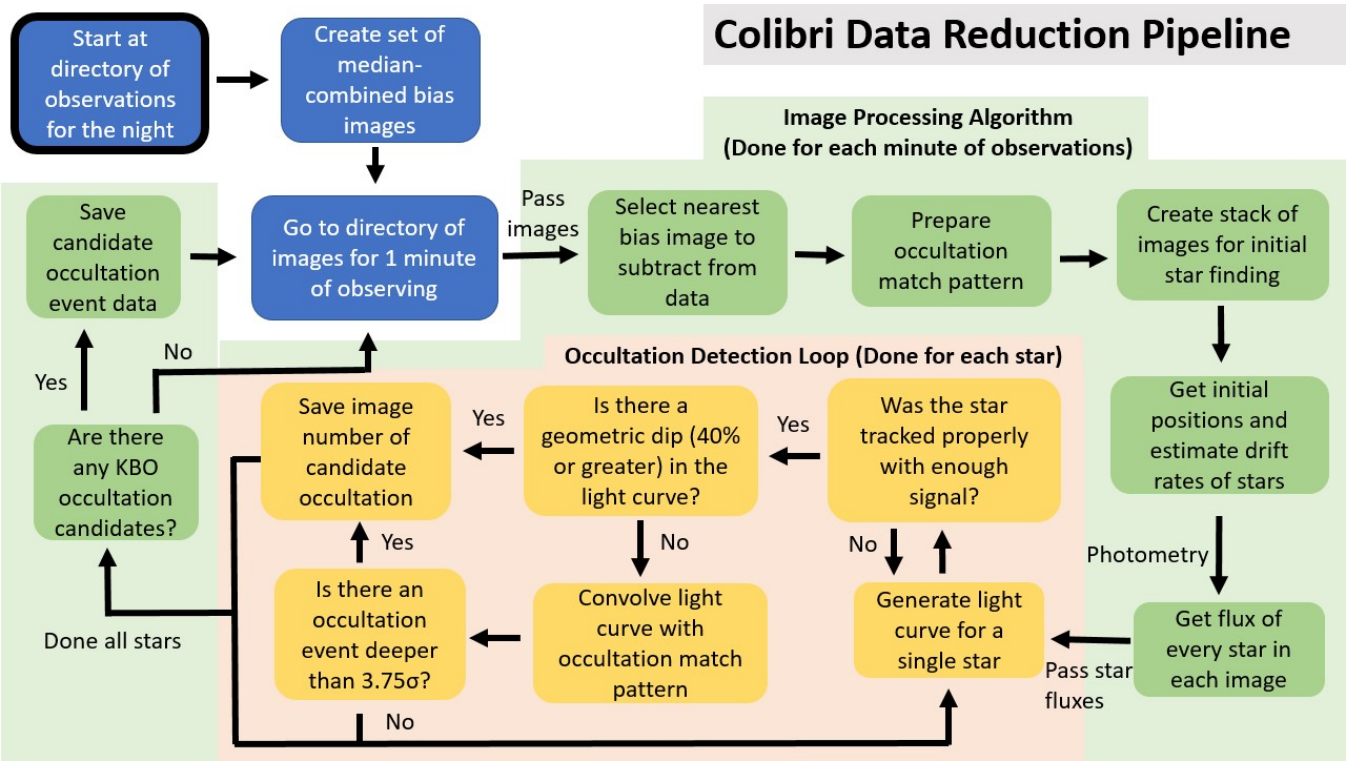

Figure 1: Primary pipeline flow

1 Need to Run

1.1 File Structure

1.2 Required Modules

2 Algorithm

The following is a basic overview of the steps the Colibri Main Pipeline takes to reduce the data. For a more detailed description of each function, see Section 3.

1. Parameters for running the pipeline are defined.

   RCDFiles (boolean) is set to *True* if the pipeline should run on .rcd images. If set to *False* the pipeline will run on .fits images.

   runPar (boolean) is set to *True* to run the pipeline in parallel mode, which will process minute-long datasets on different cores.

   The name of the telescope, the gain (*high* or *low*) for .rcd files, the date of observations, and the base path to the observation and archived data is set here.

2. Filepaths to the directories where the data is stored are defined. Note that filepaths in the pipeline are managed using python's **pathlib** library. This is so the pipeline can be run on both Linux and Windows operating systems, and to make file structure navigating and accessing more straightforward.

3. Bias images are prepared using the **makeBiasSet** function (Section 3.14). This creates a set of median combined "master bias images" to be subtracted from the science images before they are run through the pipeline. During data collection, a set of 50 bias images are taken every 15 minutes to mediate the effects of changing bias levels throughout the night.

4. The Ricker wavelet ("Mexican Hat") kernel is prepared using **RickerWavelet1DKernel** module in **astropy.convolution**. The width of this kernel is determined by the exposure time of an image and the characteristic scale length of a KBO occultation. The kernel will later be convolved with individual star light curves to search for occultation patterns in the data (Section 3.4).

5. The main part of the pipeline is run for each minute-long data set. If running in parallel, the pipeline runs on minute-long directories simultaneously on multiple cores. Otherwise, the directories are run in sequence. If the pipeline is running on .fits files, there is a check in place to see if the conversion has taken place. If not, the .rcd files are converted to .fits files before calling **firstOccSearch**. The function **firstOccSeach** (Section 3.5) is really the main function of the pipeline which runs through all the steps necessary to process the data and search for occultation events. It runs on one minute-long data directory at a time.

   5.1 The filepath to a minute-long dataset (**minuteDir**) is passed to the **firstOccSearch** function (Section 3.5), along with the list of master biases, the generated kernel from Step 4, the exposure time and gain of the images.

   5.2 The folder in which results are saved is created if it doesn't exist already.

   5.3 The median combined bias image that is closest in time to the current minute is selected using the function **chooseBias** (Section 3.2).

   5.4 The detection threshold for star finding and the circular aperture radius for photometry are chosen.

   5.5 A list of image filepaths within the current **minuteDir** is made.

   5.6 The dimensions of the images and the number of images in the directory is determined using either **getSizeFITS** (Section 3.8) or **getSizeRCD** (Section 3.9), depending on the **RCDFiles** parameter set in Step 1. The current minute directory is skipped if there are too few images (set to 3x the kernel length).

   5.7 Star positional data is created from a median combined stack. Image stacking is performed using the **stackImages** function (Section 3.20). This image is passed to **initialFind** (Section 3.12) to perform object detection using the **sep.extract** module. The object positions and half-light radii are saved in a .npy file labeled ***minuteDirName_detectThresh****sig_pos.npy* in the results directory. If there are too few stars found, this step will be repeated until enough detections are made.

   5.8 The average drift rate of the star field over the minute is calculated. This is done by importing the first and last images in the set using either **importFramesFITS** (Section 3.10) or **importFramesRCD** (Section 3.11). Each of these images are passed to **refineCentroid** (Section 3.17) along with the initial star positions to refine the positions using **sep.winpos**. The average drift rate of stars over the minute-long data set is calculated using **averageDrift** (Section 3.1). If the average drift in either the x or y direction is greater than the threshold, drift will be accounted for when doing photometry.

   5.9 A time series for each star over the minute-long data set is created, accounting for star field drift if necessary. For each image in the series, the image file is imported using the function **importFramesFITS** (Section 3.10) or **importFramesRCD** (Section 3.11). This returns the image array and the header time. This information along with the previous image's star positions is passed to **timeEvolve** (Section 3.22) or **timeEvolveNoDrift** (Section 3.23) if the drift is significant. These functions perform aperture photometry using the **sep.sum_circle** module, including local background subtraction in an annulus. The output of this step is a multidimensional array with dimensions [frames, star number, X coord, Y coord, list of fluxes, list of unix times].

   5.10 The time series of each star is checked for candidate occultation events. This is done by sending each star's time series to the **dipDetection** function (Section 3.4). This function runs through the following steps:

      5.10.1 The light curve is checked for tracking failures, and a Signal-to-Noise Ratio threshold of 5 is set.

      5.10.2 First the light curve is checked for any dips greater than 40% of the light curve normalized by the median. If found, the time series and the image frame number of the event are returned to **firstOccSearch**.

5.10.3 If no events of this type are found, the light curve is convolved with the Ricker Wavelet kernel created in Step 4. A check is performed to make sure the time series isn't cut off at the beginning or end of the minute.

5.10.4 The convolved time series is checked for a dip due to diffraction (candidate occultation event). A significant dip is considered to be

$$(lightcurve\ minimum) < (mean\ background\ level) - 3.75\sigma \quad (1)$$

Where the background is taken to be the entire convolved lightcurve excluding a specified number of buffer frames at the beginning and end of the curve. σ is taken to be the standard deviation of the background. If a dip of this significance is found, the original light curve and the frame at which the dip is found are returned to **firstOccSearch**.

5.11 Light curves containing events are archived to be run through the secondary pipeline. A list of frame numbers where events were detected is created from the results of the dip detection step. A list of light curves containing events is also created. The number of frames on either side of the detected event to archive is calculated as well.

For each detected event a .txt file is created that saves the important data about the event. This file has a header with the name of the image where the event occured, the header date and time for the event, the star coordinates, telescope name, and star field identifier. It also contains a table with the list of image filepaths, the image time (seconds only), and the star flux.

5.12 Printout statements with the number of detected stars and percentage of events are printed to the screen.

6. Steps 1-5 are completed for each minute-long dataset until the entire night is processed. When the pipeline has finished, a directory labelled by the processing date in *ColibriArchive* will contain .npy files with star coordinates for each minute, any master bias images created, median combined images for star detection in each minute, and all the detection .txt files.

3 Functions

3.1 averageDrift

- Determines the median x and y drift rates (px/s) of all stars across a minute-long data set.
- Compares the star positions between two frames, and uses the time difference between these frames to determine the drift rate.

Input:

1. 3D array of [x,y] star positions - [# frames (should be 2), # stars, # position columns (should be 2 - X & Y)]
2. list of header times of each position

Returns:

1. median x drift rate [px/s]
2. median y drift rate [px/s]

Algorithm:

1. Convert header times to unix times (for simple subtraction).
2. Get number of pixels each star has drifted in x and y separately by subtracting the last frame positions from the first frame positions.
3. Get the time interval between the first and last frame.
4. Calculate a drift rate for each star by dividing the amount of drifted pixels by the time. Take the median in x and y.

3.2 chooseBias

- Choose the correct master bias by comparing time to the observation time

Input:

1. Filepath to the directory for the current minute-long dataset.
2. 2D numpy array of [master bias image datetimes, master bias image filepaths]

Returns:

1. Bias image array that is closest in time to observation

Algorithm:

1. Get a datetime object for the current directory using **getDateTime** (Section 3.7).
2. Make an array containing the differences between each master bias datetime and the current datetime. Select the array index with the smallest difference. This is the master bias image that is nearest in time to the time the current directory was created.
3. Load in the image data for the selected master bias image and return this image array.

3.3 clipCutStars

- When the photometry aperture is near the edge of the field of view sets flux to zero to prevent fadeout

Input:

1. List of X coordinates for each star in pixels
2. List of Y coordinates for each star in pixels
3. Length of image in x direction
4. Length of image in y direction

Returns:

1. Indices of stars to remove

Algorithm:

1. Set a threshold for the number of pixels within the edge of the image to remove stars
2. Make arrays of star x and y coordinates.
3. Get indices of stars which have their x coordinates outside the threshold on the right side of the image and on the left side. Get indices of stars which have their y coordinates outside the threshold on the top and the bottom.