Colibri Secondary Pipeline

Rachel A. Brown

May 2022

Contents

This document describes the secondary data reduction pipeline for the Colibri telescope array. It includes functions written by Mike Mazur to add .rcd file handling. There is also a function written by Emily Pass that was in her original version of the Main Pipeline. This pipeline is designed to run on the output from the Colibri Main Pipeline. Once candidate events have been identified, this pipeline matches the events to a set of pre-made occultation kernels. A basic overview of the pipeline flow is shown in the below flowchart:

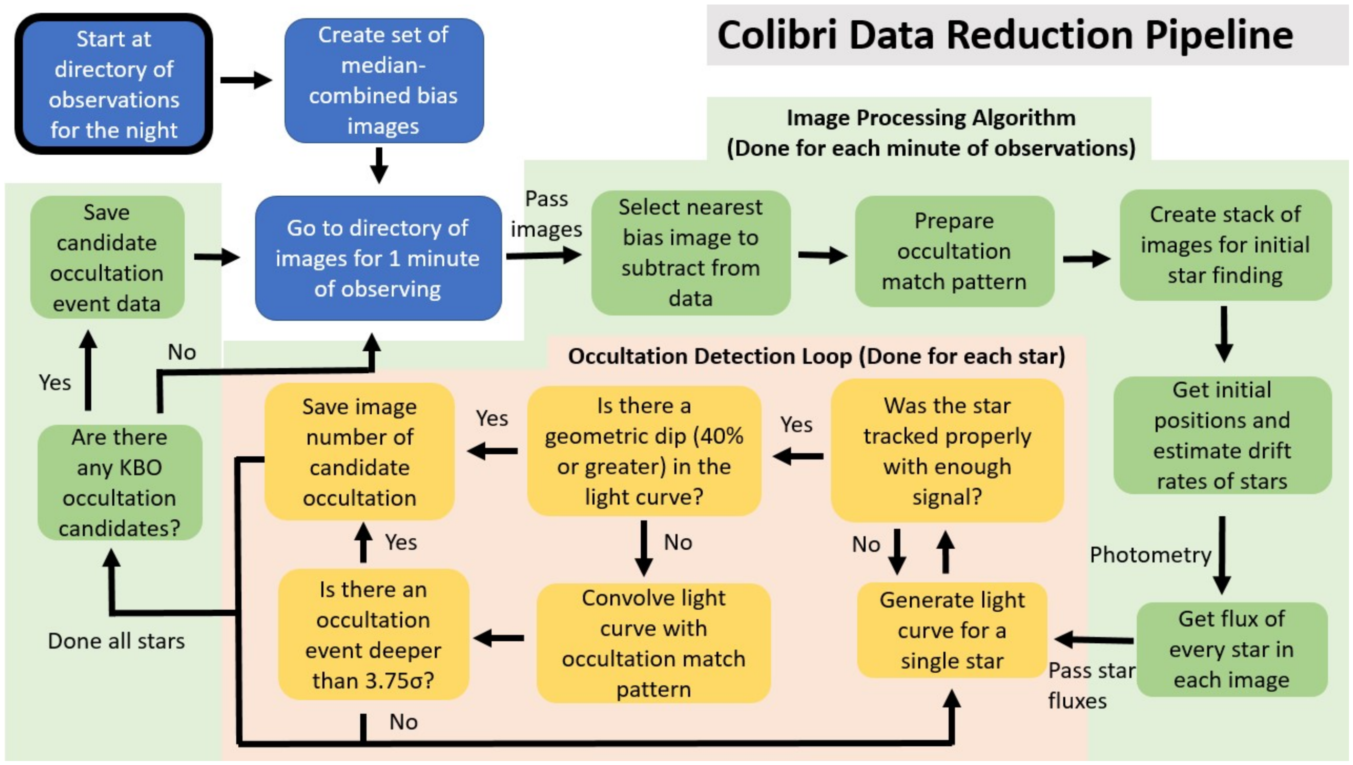
Colibri Data Reduction Pipeline
Start at directory of observations for the night
Create set of median-combined bias images
Go to directory of images for 1 minute of observing
Pass images
Image Processing Algorithm (Done for each minute of observations)
Select nearest bias image to subtract from data
Prepare occultation match pattern
Create stack of images for initial star finding
Get initial positions and estimate drift rates of stars
Photometry
Get flux of every star in each image
Pass star fluxes
Occultation Detection Loop (Done for each star)
Generate light curve for a single star
Was the star tracked properly with enough signal?
Yes
No
Is there a geometric dip (40% or greater) in the light curve?
Yes
No
Save image number of candidate occultation
Convolve light curve with occultation match pattern
Is there an occultation event deeper than 3.75σ?
Yes
No
Done all stars
Are there any KBO occultation candidates?
Yes
No
Save candidate occultation event data

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 Secondary 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. This includes the name of the telescope, the gain level for the .rcd images (*high* or *low*), the solution polynomial order for astrometry.net, the date of observation, the date the main pipeline was run, and the base filepath for accessing data and saving observations.

2. A pre-made set of occultation kernels is loaded into the program. These should be made with KernelGeneratorGUI (see ColibriPipeline GitHub repository) and saved in a single text file. The corresponding kernel parameters should be saved in a separate file. The kernels are checked to see if they contain dips that are detectable with our system. If the deepest trough of the kernel is shallower than our expected noise level, the kernel is removed from the set.

3. The correct filepaths to the input files are set up. A list of all the output detection .txt files from the main pipeline is made. A list of all the median combined images saved to the archive directory is obtained. An

empty dictionary to hold coordinate transformations is created. Empty lists to hold results for both diffraction and geometric events are created.

4. Each detection file is processed.

   4.1 Data from the detection .txt file is read in. This includes the light curve (flux values over time), event frame number, star x coordinate, star y coordinate, event time, event type, star med, star std, and star number for identification.

   4.2 A coordinate transformation from image coordinates (X, Y) to ecliptic coordinates (RA, declination) is calculated using astronometry.net's web service. This is done automatically in the function **getTransform** (Section 3.1). The transformation is passed to the function **getRADecSingle** in the script **getRADec.py** to calculate the current star's coordinates in RA and Dec.

   4.3 A query is sent to Gaia using **VizieR_query.py** to get a catalog of stars in the region around the current field of view.

   4.4 The current star is matched with a star in the Gaia catalog using **match_RADec** (Section 3.2). Note that on rare occasions this function is unable to find a match and will return a table of 'nan' values. This can be due to the star being fainter than the magnitude limit given by Gaia (for example two faint angularly close stars appearing as one brighter star in our images), or because the star is a missed detection on our end. These should not happen very frequently.

   4.5 The event is matched with an occultation kernel from the set generated earlier. Events catagorized as *geometric* and *diffraction* are processed separately here to keep track of the type of events. Changes may need to be made to the processing method for *geometric* events in the future. The best match is determined by the function **kernelDetection** (Section 3.3) based on the kernel with the lowest chi-square. This returns the best matching kernel index, the Chi-square value for this match, the best starting frame index for the match, and the parameters of the best matching model (TNO radius [m], stellar angular diameter [mas], impact parameter [m], shift adjustment [frames]). This information is appended to the results list, and is also saved in a plot during the matching process.

5. All of the results are written to text files; one for *diffraction* events and one for *geometric* events.

3 Functions

3.1 getTransform

3.2 match_RADec

3.3 kernelDetection