ARIES Analysis Pipeline

by Elisabeth R. Adams
2012-08-24

Goal: The *analysis pipeline* will take you from a single, combined image for each object and filter (the end result of the *reduction pipeline*) and find nearby companions and detection limits. Summary tables and plots (suitable for talks/papers) are produced for desired sets of objects.

Some comments:

All of the scripts below are python scripts that run from a regular command line. Several of them call IRAF internally (using pyraf), typically to find the FWHM or to locate and characterize stars using daophot or apphot.

This pipeline assumes that your images were reduced using the *reduction pipeline*. In particular, you must have files named “K12345_Filter.fits” stored underneath the “objects/” directory, by filter.

Several python modules have been created to assist with this package: ao.py, aries.py, and koiPlusFilter.py are the most important, and you will need to add information to them as listed below (item 2).

NEW: all processing scripts (but NOT the DATA) are in a github repository.
Let me know if you want to be able to make commits (bug fixes and updates to this README are both greatly appreciated!)
https://github.com/elisabethadams/ao-pipeline.git

General trouble shooting:

In the “objects/” directory, it is assumed that all subfolders correspond to a real object with data, e.g. “objects/K12345/”. Avoid making additional subfolders!

1. Getting started

a. Got raw data? Go work through the *reduction pipeline* first!

b. It’s the same directory structure; if the main directory is “aries/”, then:

(i) “aries/objects/ARIES/” holds the input data for each object and filter

(ii) “aries/ao-pipeline/” has the processing scripts (this one lives under scripts/pipeline-analysis)

(iii) “aries/tables/” and “aries/plots/” are where the output will go

(iv) “aries/info/” has other useful information (e.g. 2MASS tables)

(v) “aries/batches/” is created in this pipeline for bulk uploads to KFOP

c. If you have pre-reduced data, you can still use this pipeline

Put your images in folders by object and filter: "aries/objects/OTHER_INSTRUMENT/K12345/K/"

Your file should be called “Object_Filt.fits”

2. Modify the python modules as needed

a. *ao.py* contains most of the functions needed for the AO reduction and analysis. This is where the directory structure is defined; every other script should refer back to *ao.py* instead of making new definitions
b. *aries.py* contains information specific to ARIES observations, such as the file prefix on a given night and which objects were observed
c. *koiPlusFilter.py* contains information specific to each final image (.fits), to handle the flips/rotations and scaling needed to produce images

3. Run the reduction pipeline (7 steps)

There are XX pipeline steps. Each script has basic instructions at the top of the .py file, including the pre-reqs for running the script. In normal operation, you do not NEED to edit any .py file except for options in the following: step0-setup.py (custom set of objects)

a. step0-setup.py (also calls createSettings.py, ds9list.py)

Goal: To choose objects for analysis and create input settings.
Setup: Make sure all your images is present in the “objects/” folder.
RUN: "./step0-setup.py" from the command line in “scripts/pipeline-analysis/”
Time: quick
Output:

(1) Lists all available targets and lets you choose subset to use:
1 -- All; 2 -- K*; 3 -- not K*; 4 -- subset listed *in this script*
(2) creates file “usingObjects.txt” so later scripts use the same subset
(3) creates a settings file for each object (using createSettings.py)
(4) [optionally] displays every final image in subset using ds9list.py

b. step1-findstars.py

Goal: To find every real star on a frame, quasi-automatically. Makes .coo files.
Setup: Step 0
RUN: "./step1-findstars.py"
Time: variable (interactive)
Output:

(1) creates OBJECT_J.fits.fullcoo using daophot
(2) will create OBJECT_J.fits.coo with coords of each object. THE TARGET SHOULD BE LISTED FIRST or else later scripts will be confused. You really want the .coo file to be right! Make sure no noise sneaks in (e.g. ghosts, cross-talk).

b. step2-getmag.py

Goal: To find delta-mag and distance of all stars on a frame. Makes .mag files.
Setup: Steps 0-1.
RUN: "./step2-getmag.py"
Time: not too long (a few seconds per frame)
Output:

(1) creates OBJECT_J.fits.mag using apphot

c. step3-findlimits.py (calls magLimits.py)

Goal: To find the detection limits from 0.1-4” from target. Creates lim.tsv files.
Setup: Steps 0-2. Also, you will need to query 2MASS to get the 2MASS mags the first time you use an object (see catalogs.py for details)
RUN: "./step3-findlimits.py"
Time: not too long (a few seconds per frame)
Output:

(1) creates OBJECT_J.fits_limt.tsv using apphot via magLimits.py

d. step4-compstars.py

Goal: To list every real point source on a frame
Setup: Steps 0-3.
RUN: "./step4-compstars.py"
Time: quick
Output:

(1) A file (objects/OBJECT/OBJECT_stars.tsv) listing the distance, delta-mag, and X/Y pixels of all stars (including the target). The FWHM is also listed. Information for all filters appears in this file.

e. step5-tables.py

Goal: To create summary tables (text and LaTeX for papers)
Setup: Steps 0-4
RUN: "./step5-tables.py"
Time: quick
Output:

(1) a table of the object name, fwhm, 2MASS mag, and limits from 0.05-4” (one table per each filter, outputting both .tsv and .tex)

f. step6-plots.py (uses koiPlusFilter.py and finderPlots.py)

Goal: To plot each image and to make summary graphs for papers
Setup: Steps 0-5, plus edit koiPlusFilter.py for fine-grained control of plots. Make sure the Shiftlist file exists and that the files it refers to are in your data directory!
RUN: "./step6-plots.py"
Time: not so long to run (a few seconds per image), but a while to figure out the right settings (esp. the scaling). Also depends on which plots you want.
Output (all are optional):

(1) single plots for each image (showing full field of view and zoomed-in)
(2) plot of zoomed-in version of all images you want
(3) plot of full field of view in pretty publishable form of ONE object

g**. step7-bulkUpload.py**

Goal: To make a .tar archive that KFOP can ingest
Setup: Steps 0-4 (tables and plots are optional)
RUN: “./step7-bulkUpload.py”
Time: quick
Results:

(1) copies final images and limit files to batches/ folder and makes tar file (e.g. ea-20120731-002.tar)

(2) creates an observational summary tar (e.g. obsnotes-20120731-003.tar)

Congratulations! You have run your objects through the entire analysis pipeline! You have images, lists of detected sources, and limits on additional objects. Go publish something with it...