ARIES Reduction Pipeline

by Elisabeth R. Adams
2012-09-21

Goal: The *reduction pipeline* will take you from raw data to a single, combined image for each object and filter.

Some comments:

Most of the scripts below are python scripts that you run from a regular command line. Several of them call IRAF internally (using pyraf). However, Steps 4 and 5 create .cl scripts that you will need to run in IRAF after running the python scripts.

I find it helpful to have three terminals open: if your main directory is “aries”, one in "aries/ao-pipeline/pipeline-reduction/", and both a regular and an IRAF window in "aries/data/yyyymmdd/".

Steps 1-4 only need to be run once (to calibrate and sky-subtract the data). Steps 5 and 6 combine all the images for a given object and frame, and may be run multiple times as you figure out which images you should include and which to omit. Step 7 cleans things up when you are done making the images, so that you can proceed to the next major step: the *analysis pipeline*!

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 shootin*g:

The sky-subtracted frames (“qtargetNNNN.red.sub.fits”) get written TWICE, once in Step 4 and again in Step 5/6. Sometimes there are... problems. (If you abort or crash something, the files may get deleted.) It is often easiest to start over by wiping out all of the output from the previous step(s).

The bad pixel mask (BPM1.fits in the data directory) sometimes gets corrupted and should be recopied (from blankBPM.fits in the pipeline directory). Common symptoms include lots of big white blobs around your dither pattern (something is going wrong and the bad pixel mask mistakenly thinks it can fix it). You can turn OFF using the bad pixel mask in either the first or mask pass (fp_xzap=no, mp_xzap=no) and/or updating it, i.e., just use the one you have. and/or updating the bad pixel mask in either the first pass (fp_badpixupdate=no, mp_badpixupdate=no) or the mask pass if it starts acting up).

These scripts will have bugs, particularly in edge cases! Tell me what breaks... (bonus points for getting on github and fixing it yourself).

1. Getting started

a. Find somewhere with a lot of disk space (e.g. /data/dupree7)
    Name the overall folder (e.g. "aries").
    Make the subdirectory "aries/data/"

b. Where to put your data
    Put your data in by the date: "aries/data/20111008"
    This includes: (1) target data, (2) sky flats, (3) darks
    (Note: the date can either be a single date or the first night of the run.)

c. Put the processing scripts in "aries/scripts/", which has two subdirectories:
    (1) "scripts/pipeline" (v1.0) take raw data and returns a final combined image by filter and object
    (2) "scripts/analysis" (coming soon!) take an image for an object and filter and extract information (number/location/magnitude of companions, detection limits, plots)

d. Later, the directory “aries/objects/” will be made (in Step 7) to store results.

2. Update information for night of data

a. ALL of the important definitions are in the python package "aries.py"
    Make sure the directory structure matches what you set up in Section 1!

b. Create a section for your data following the examples
    nightDir = "20100503"
    targetBaseName[nightDir] = "star" # data prefix; e.g. “star” or “target”
    framesForNights[nightDir]=[1,692]

c. Additional information can be specified later as needed

d. Make sure the variable "useThisNight" is set to the desired value!

3. Run the reduction pipeline (7 steps)

There are 7 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 Step 6.
“*Output” below lists the files created in “aries/data/” unless noted otherwise.*

a. step1-corquad.py

Goal: To remove (mostly) the cross-talk in each quadrant
Setup: Make sure all your data is present in the “data/” folder.
RUN: "./step1-corquad.py" from the command line in “scripts/pipeline-reduction/”
Time: a
Output:
    (1) creates and runs shell script corquad.sh
    (2) creates files “qtargetNNNN.fits” (for files “targetNNNN.fits”)
    (3) archives “targetNNNN.fits” files under “data/raw/”
    (4) copies script ds9list.py (tool to open list of images)

b. step2-lists.py

Goal: To generate lists of all of the darks, skies, and data by filter, exposure time, and night (used by later steps)
First: Make sure your objects are named consistently in headers.
RUN: "./step2-lists.py"
Time: quick (though it takes some tim=======
Output:

(1) creates allDark_1.0.txt (etc.) for all available darks
(2) creates allSky_J.txt (etc.) for all available sky flats
(3) creates allSkyD_1.0.txt (etc.) for sky flats with same exptime
(4) creates allObj_OBJECT_J.txt (etc.) for each object, filter
(5) creates night1_J.txt (etc.) for each night of data in same filter

c. step3-calib.py

Goal: To dark-subtract the calibration data and then sky-correct the target data (we do NOT dark-subtract the targets; sky subtraction does that)
First: Steps 1-2.
RUN: "./step3-calib.py"
Time: a while (flat correcting lots of images adds up)
Output:

(1) creates masterDark_1.0.fits (etc.)
(2) creates skyNNNN.d.fits using list allSkyD_1.0.txt (etc.)
(3) creates masterSky_J.fits using list allSky_J.txt (etc.)
(4) normalizes masterSky_J.fits to normSky_J.fits (etc.)
(5) divides data by flats using allObj_OBJECT_J.txt (etc.) to create files "qtargetNNNN.red.fits" (i.e., the calibrated data)
(6) moves dark and sky frames to "data/raw/darks/" and "raw/skies/"

d. step4-skysub.py

Goal: To sky-subtract the target data (once per night and filter)
Setup: Steps 1-3.
RUN: "./step4-skysub.py"
Then in IRAF (in "data/YYYYMMDD/" directory):

(A) xdimsum [if package is not already loaded]
(B) cl < skysub.cl

Time: a LONG while (sky subtraction takes a few seconds per image)
Output:

(1) copies bad pixel map BPM1.fits
(2) creates iraf script "skysub.cl" (which uses list "night1_J.txt")
(3) iraf script creates files "qtargetNNNN.red.sub.fits" (NOTE: the first-pass red.sub.fits files still have dark spots where stars are; these will be removed in the Step 5 in the second pass. If they are NOT removed, then something is misconfigured.)

e. step5-objects.py

Goal: To combine each individual object into a final image (setup)

Setup: Remove obviously bad frames from “allObj_OBJECT_J.txt” ; it helps to run “./ds9list.py allObj_OBJECT_J.txt red”

RUN: "./step5-objects.py"

NOTE: Step 5 is more interactive. First, choose which objects to run. Next, every frame is centered (only once, to save time) to track the motion of the star with the dither pattern; this is quasi-automatic, but sometimes you are prompted to adjust the daophot criteria. (Common causes: the FWHM has gotten worse; you need to use a lower detection threshold; there is more than one star on the field and you need to pick the right one.) It is helpful to use “ds9list.py allObj_KXXX.txt red" to display all frames for an object.

Time: a while (the first time, quicker subsequently)

Output:

(1) creates iraf script “runObjects.cl”

(2) creates “qtargetNNNN.fits.coo” and “.fullcoo” while centering

(3) creates “Shiftlist_OBJECT_J.txt” with list of centers

(4) creates “saved_step5_objects.py” in “aries/scripts/” (for Step 6)

While you CAN run the iraf script now, you should check Step 6 first!

f. step6-inspect.py

Goal: To reject bad frames before making final image (quality control)

Setup: run Step 5 (will NOT work otherwise)

RUN: "./step6-inspect.py"

Then in IRAF (in “data/YYYYMMDD/” directory):

(A) xdimsum [if package is not already loaded]

(B) cl < runObjects.cl

NOTE: ARIES data can be highly variable from frame to frame. Frames are rejected based on too-large FWHM, as measured with imexam. What “too large” is varies by object and night and is set in this notebook by hand. If you DO get rid of frame(s), you must **rerun step5-objects.py**.

Time: quick

Output:

(1) creates “allObj_OBJECT_J_use.txt” (without bad seeing images)

(2) iraf script creates “qtargetNNNN.red.sub.fits” (updated version) as well as .red.sub.ocm.pl, .obm.pl, .rjm.pl, and .crm.pl (used by xdimsum)

g. INSPECT your data

Are there big white blobs? That means the sky subtraction wasn’t very clean. Things to try: wipe the bad pixel map, if used; tweak some of the xdimsum settings, such as nmean=11 (try combining more or fewer images; note the run time scales with this number); get rid of frames with terrible seeing so you’re only combining the best. This is a bit of a black art. Play around.

h. step7-archive.py

Goal: To archive the final images elsewhere, where they can be analyzed.
Setup: Do you like the looks of the final images? Check again
RUN: "./step7-archive.py"
Time: not too long
Results:

(1) creates "/aries/objects" if it doesn't already exist
(2) creates "/aries/objects/OBJECT/J/" for each object/filter
(3) copies OBJECT_J.fits, allObj_OBJECT_J_use.fits, Shiftlist_OBJECT_J.txt to "/aries/objects/OBJECT/J/"
(4) copies several intermediate files to above directory + "saved/"
(5) finds the estimated seeing of the final image (interactive) and creates summary.txt

Congratulations! You have now produced an AO image for an object and filter. Proceed to the analysis pipeline...