User Guide
for RECONS Astrometry Python Pipeline
("the Pypline")

Eliot Halley Vrijmoet

June 27, 2022

Introduction

The RECONS astrometry pipeline uses many images of one field to measure a given star's position over time and determine its proper motion and parallax. This process is impossible to complete without some human judgement, however, thus this user guide is intended to guide you through running that pipeline.

This guide describes the revised version of the pipeline, now written almost entirely in Python. Earlier versions (3C and earlier) were written in a combination of Bash, IRAF, FORTRAN, and IDL. The new Python pipeline ("the pypline") uses some different file names and has several enhanced functions, but the steps of the process are unchanged from earlier versions. The Appendix at the end of this manual provides a quick translation between the old and new functions and files.

If you are looking for more details about what the pypline code itself is doing and how its many parts fit together, please consult the **Developer Manual**.

The Big Picture

At the telescope we take images of each star of interest several times ("epochs") per year for several years. For each of those target stars, we then extract its position in every image. Our targets are generally nearby stars (within 25 pc), and the goal is to track their motion over time.

Each target star ("pi star") also has in its image a set of background stars. Because these stars are not nearby (> 1 kpc away), they are mostly stationary over time, so we use them as reference points for the pi star's motion. In every frame where we get the pi star's position, we also get the positions of each of those reference stars ("ref stars"). For each pi star we use a single set of ref stars in every image (usually the brightest 5–15 stars), and collectively this set is called the "setup."

To complete the process, we compute how the target star's position is changing over time with respect to its ref stars, and to that data we fit an astrometric model. This model gives us our goal: the target's proper motion and parallax. We also fit a variability model to the photometric data from all the frames, which tells us how the target star changes in brightness over time.

With that in mind, reducing the data for one pi star requires completing five major **stages**:

1. `tagframes`: Check the quality of the images, removing any that could produce unreliable astrometry or photometry. Then "tag" the position of the pi star and ref stars in each image, so the next stage of the code knows where to look for those sources.

2. `centroidframes`: For each image, use the tags from Stage 1 to fit a model to each star's PSF. This determines each star's position ("centroid") more precisely than the initial tags.

3. `parallaxprep`: Prepare the centroid positions for astrometric model fitting. This involves computing the parallax factors and adding corrections for the effects of Earth's atmosphere and the field's rotation.

4. `pisolve`: Fit the astrometric model to the positional data. This result gives the pi star's proper motion and parallax.

5. `varsolve`: Fit the variability model to the photometric data. This result gives all the stars' variability.

Each target star (a.k.a. "pi star") has a full name (such as GJ1005AB or DEN1756-4805) and a 7-character short name (such as gj1005x or den1756). Most files generated by the pypline use the system's short name, although the full name is used primarily for the final data files and plots.

Stage 0 Prepare the data reduction environment

These instructions apply to **reduction updates**, i.e., when a system has been reduced at least once in the past. To reduce a system for the first time, replace these instructions with the alternate Stage 0 for new systems.

1. Log on to the recons machine: enter `recons` at the command prompt and enter your password.

2. If you will be using IRAF instead of Python to view and tag your frames, start that program now:
   (a) At the command prompt, enter: `xgterm -sb &`
   (b) In the xgterm window that opens, navigate to your IRAF folder: `cd ~/iraf`
   (c) From within that folder, launch IRAF with this command: `ecl`
   (d) Back in your original terminal (not xgterm), launch DS9: `ds9 &`

   If you will *not* be using IRAF, you do not need to launch DS9 yet. (If you want to launch it now anyway, you must use this command: `ds9 -title pyplineds9 &`)

3. From your terminal, navigate to the data folder for the system you will be reducing. These folders are located at `/nfs/recons4/CTIOPI/regions/` within subfolders organized by right ascension (RA). For example, DEN1756-4805 is at RA 17^h 56^m 56^s, so its folder is: `/nfs/recons4/CTIOPI/regions/15-17/den1756-4805`.

4. Enter `ls` to show the folder contents (example in Figure 1). Each data folder should include many `*.o.fits` files; these are the images of the pi star's field. There should also be subfolders named with the prefix `pi.` and numbers indicating dates (format: `pi.YYYY.MMDD`).

   Note the name of the most recent `pi.*` folder (you will need it in the next step). In the Figure 1 example, this most recent folder is `pi.2019.1105`.

   Potential issue: there are very few `*.o.fits` files in the data folder.

5. In the terminal, run `setup.reduction` to create a new reduction folder pre-loaded with the essential files. The script will then give you several prompts; in particular, you will need to:
   (a) Enter the name of the most recent existing `pi` folder (for example, `pi.2019.1105` in Figure 1).
   (b) Enter a name for your new reduction folder, using today's date in `YYYY.MMDD` format. For example, if today is 31 March 2022, your new folder should be `pi.2022.0531`.
   (c) Enter information about the system, including its name, RA, DEC, and proper motion. Press Enter to accept the default values of these properties, unless you notice a clear error to correct. Note: the star's name at this prompt must match its name in the RECONS astrometry catalog.

   Potential issue: `setup.reduction` alerts you that frames are missing from the astrometry catalog.

6. After completing the above script, `cd` into your new folder. If you will be using IRAF, `cd` into your new folder from your xgterm window as well.

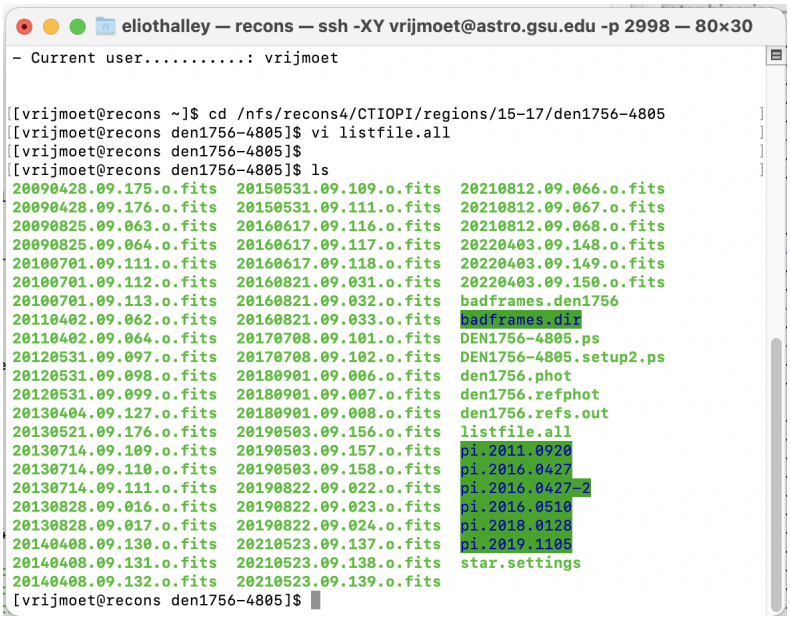

Figure 1: Example star data folder contents. This system is DEN1756-4805 (RA 17^h), and its most recent reduction folder is `pi.2019.1105`.

7. Print a paper log sheet to fill out as you complete the reduction. The log sheet is located here: `http://www.astro.gsu.edu/~thenry/CTIOPI/pi.reduction.logsheet.2021.0205.pdf`

   If you are not working remotely, locate the paper packet for this system in the red/maroon binders.

Stage 1 Inspect and tag the frames

From this stage forward, run every step from within your new `pi.YYYY.MMDD` data reduction subdirectory. Stage 0 above created many essential files, including:

- `listfile`: lists all the frames (`*.o.fits`) used for the reduction.
- `listfile.all`: lists all the frames taken for this system, formatted to let you add notes on each frame.
- `listfile.newframes`: lists the frames that are *new* in your reduction (relative to the previous reduction).
- `star.settings`: gives several important parameters for the system, such as its full name, 7-character short name, coordinates (J2000 and 2MASS), and filter.

In this stage you will inspect the images to identify frames to use in the reduction, then you will tag stars in each frame so their precise positions can be computed.

You may use either Python or IRAF to display the frames in DS9 for this stage. If you are using Python, enter `tagframes.py` at the terminal. If you are using IRAF, enter `tagframes` in xgterm.

(Stage 1) Step 0 Display and inspect the new frames

With `tagframes` launched, you will be presented with a menu and an initial prompt to enter the listed commands. Numbered commands must proceed in order, and named commands can be run at any time. Commands are not case-sensitive. To inspect the frames, we proceed as follows:

(a) At the `tagframes` initial prompt, enter `dis`. If you are using Python, wait for a minute now while it launches the DS9 window.

(b) When prompted for the name of the listfile, enter `listfile.newframes`.

(c) The program will ask if you would like to mark the ref star setup using the `*.setup.reg` file. You should always answer `yes` here, unless you intend to change the ref star set for this reduction.

Marking the stars is *only* for your information — it does not affect the data. The `*.setup.reg` was created for you by the `setup.reduction` script in Stage 0.

If you need to adjust the regions, in the DS9 menu bar select `Edit` and then `Region`. Regions will then be editable by dragging-and-dropping and double-clicking.

Potential issue: DS9 reports that regions cannot be loaded, then crashes.

(d) The pi star will always be the star in the image with the highest number (or the highest numbered *two* stars for a resolved binary). When each frame is displayed, check the pi star and brightest ref stars for the following properties:

- Shape: Press `e` over a star in DS9 to view a contour plot. Verify that the PSFs are round and symmetrical. Slightly elliptical is acceptable, but asymmetrical or skewed PSFs are unreliable for astrometry and should be excluded. See Figure 3 for examples.
- Saturation: Press `r` over a star in DS9 to view a radial plot to check its counts. Verify that the maximum counts are not above ∼65,000 and the top of the PSF is not flat.
- Non-astrophysical anomalies: Use the `e` (contour), `r` (radial), or any other commands to verify that the PSFs have not been altered by cosmic rays, satellite trails, or artifacts from the CCD (bad columns or bad pixels). A bad feature will affect the astrometry if it is on the PSF or its tail less than 7(?) pixels from the peak.

Additionally, check that all ref stars are fully present, as sometimes the frame position is shifted such that some ref stars are missing or have parts of their PSFs outside the field. Partial PSFs are especially important to note.

If you find any issue in a frame, write a brief note (10 words or fewer) to the right of that frame name in the `listfile.all` file.

If the issue affects the pi star or leaves fewer than five good ref stars, the frame must be removed. Do this by marking an `X` one space to the right of the frame name in `listfile.all` and also writing a brief explanation (10 words or fewer).

Figure 2 shows examples `listfile.all` notes.

Note for displaying frames: You can come back to the `dis` command at any point and use it to view a single frame by entering the word `one` followed by the image's file name. For example, to view `20220412.09.160.o.fits`, at this file name prompt enter: `one 20220412.09.160.o.fits`. If a tag file (`*.list.*` file) exists for that frame, the program will ask if you want to display it as well.

(Stage 1) Step 1 Tag the ref stars and pi star in one reference image.

After you have checked all the new frames and marked the bad ones, you are ready to tag the ref stars and pi star(s) in the setup. You will tag a single high-quality frame as a reference, then later (Step 3) run the `autotag` program to automatically tag the rest of the frames using that reference.

(a) Run `tagframes` again and select Step `1` for reference image tagging.

(b) At the reference image prompt, always select the old trail plate as the reference image, unless you know of a better image to use. Good reference images have all the ref stars visible, each with >10,000 counts if possible, and each with high-quality PSFs (according to the criteria in Stage 1 point (d)).

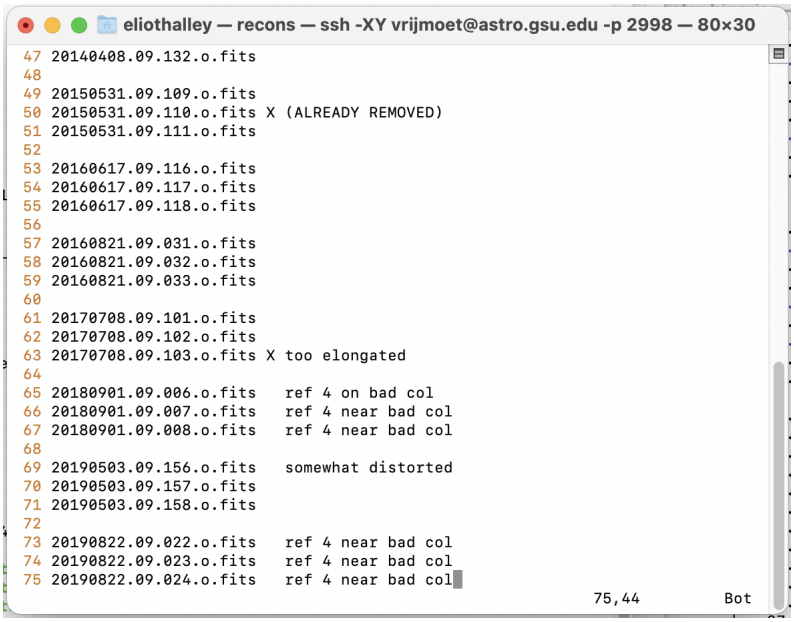

Figure 2: Example `listfile.all` file (for DEN1756-4805). Notes next to frame names indicate issues with those images. Frames marked with `X` will be removed from the reduction before `autotag` tags the frames.

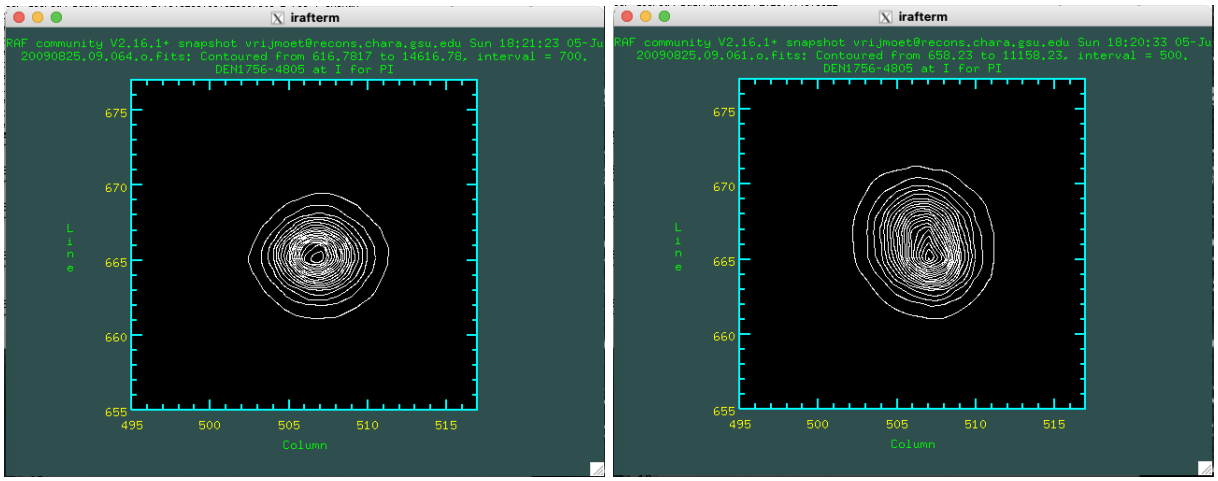

Figure 3: Contour plots (IRAF) for a star with (*left*) a reasonably symmetrical PSF, and (*right*) an asymmetrical or distorted PSF that should be noted in `listfile.all` (and marked as a bad frame).

(c) At the "Display tags" prompt, always select `yes`, unless you intend to change the ref star setup for this reduction.

(d) When the reference frame displays, tag the stars in their numerical order. Create each tag by centering the mouse cursor over the star and pressing `space`. In Python, the tagged position will then print to the terminal (but in IRAF it will not print).

When you have tagged the last star, exit by pressing `q` in DS9 if you are using Python, or `ctrl-z` in DS9 if you are using IRAF.

(Stage 1) Step 2 Verify the reference frame tags.

This step is optional. The program will display the reference frame in DS9 with the tags you created. It will then open the tags file for you (letting you choose your editor: vi, emacs, or nano) and let you edit them. You may adjust the tag coordinates, add lines, delete lines, and change their order (as long as you change their numbers so they are listed in order). Each tag is a set of image coordinates, followed by the star number.

If you need to adjust the tags without using this program, they are stored in the file `autotag1.list`.

(Stage 1) Step 3 Launch the autotag program to tag all the frames.

When the reference frame is tagged, select Step 3 to let `autotag` use it to tag the rest of the frames. This step will print many lines of output to the terminal, and will take longer to complete if there are more frames (up to a few minutes).

When the autotagging is complete, a report will be printed to the screen. This report is also saved in the file `autotag.report` if you need to access it later.

If the autotag report states an unexpected result for a frame — such as many stars outside the field, or a star outside the field that you know should be inside — you should use the `dis` command in `tagframes` to display that frame and its new tags to check that `autotag` tagged the frame properly. If necessary, you can tag a single frame again using the `tag` command in `tagframes`.

Stage 2 Measure the stars' positions

With the frames inspected and stars tagged, we are ready to get more precise measurements of the stars' positions in each frame. To begin Stage 2, enter at the terminal: `centroidframes.py` (for this stage, and all that follow, will use only Python, not IRAF).

The next five steps must be completed in order. You may exit the program between steps; resume it by re-running `centroidframes.py` and following the initial prompt to jump to the step where you left off. You can redo a previous step with no special preparation, but then you must redo all the steps that follow it.

(Stage 2) Step 1 Create a `*.sex.*` file for each frame.

When you run this step, you do not need to provide any input. When it is complete, it will report how many `*.sex.*` files it created.

This process creates the frame-specific parameter files that SExtractor will need in Step 2. It creates one `*.sex.*` file per frame: each of these file names starts with the star's short name and ends with the frame

number. For example, for GJ1005AB, these would be `gj1005x.sex.001`, `gj1005x.sex.002`, etc. Any existing `*.sex.*` files are deleted here before new ones are created.

(Stage 2) Step 2 Run SExtractor on each frame.

This step also requires no input from you after you select it.

This step runs the SExtractor program on each frame using its associated `*.sex.*` file, finding the precise centroid of each tagged star in each frame. It creates one `*.centroid*` file per frame: each of these file names starts with the star's short name and ends with the frame number. For GJ1005AB, these look like `gj1005.centroid001`, `gj1005.centroid002`, etc.

Each `*.centroid*` file contains a 10-line header plus one line per tagged star. If any stars are missing or repeated, then later steps in the pipeline will fail. Steps 3 and 4 (next) will help catch and correct any errors.

(Stage 2) Step 3 Check the SExtractor results/centroiding.

When you run this process, it inspects each `*.centroid*` file to check adherence to this golden rule:

Each ref and pi star must appear once, and only once, in each frame. No repeats, no missing stars.

The output of this process is a table of each centroid file and its associated image, with arrows pointing to any `*.centroid*` files that had problems. [SHOW EXAMPLE FIGURE]

When a `*.centroid*` file has an issue:

Missing or extra stars in a `*.centroid*` file happen because of either the data or the tags.

- **Data problems** are issues with the image itself. These include when the star is too faint relative to the background, or when its PSF is very near to a bad pixel, cosmic ray, or satellite trail.
  We will fix these cases in the next step, so proceed to Step 4 (Frame Fixer).
- **Tagging problems** mean the image has no issues near the stars, but the `autotag` program did not place the tag close enough to the PSF(s) of the star(s). This may happen when the ref frame tags or the proper motion were not accurate enough. You probably have a tagging issue if no stars were centroided in an image, or if many images in a row are missing the same star(s).
  *You **must** fix these cases by re-tagging those frames before proceeding: go back to* `tagframes` *and use* `tag`*. After re-tagging, start* `centroidframes.py` *again from the beginning.*

How do you know which case applies to each of your problem frames? Check each bad frame by dispaying it with its tags (`tagframes` command `dis`). Check that the tag is centered on each star and there are no defects (bad pixels, cosmic rays, etc.) on any of the tagged stars' PSFs.

(Stage 2) Step 4 Fix frame results (missing/extra stars).

This step is a mini-program that lets you add, replace, or delete lines in the `*.centroid*` files. If the Check Step (Step 3) shows that none of your frames are missing stars and none have extra stars, `centroidframes.py` will prompt you to skip this step.

Before skipping this step, check `listfile.all` for frames with unsuitable stars that were not caught by this program. If a ref star's PSF in one frame is not fully visible or is not free from defects (refer to point (d) of Stage 1), you must replace that star in that frame's `*.centroid*` file.

If there is a problem requiring that you add, replace, or remove a star in a frame, enter `add` or `del` and be ready to input (1) the star number(s) you need to add or delete and (2) the frames in which those star(s) must be modified.

Cases you can handle with Frame Fixer:

Missing stars or partially off-field stars: enter `add`. The program will ask for the star numbers and frame numbers. Then it will insert a placeholder line (full of `999` values) in those `*.centroid*` files to substitute for those missing/partial stars. If a specified star is already listed in a given `*.centroid*` file, this command will **replace** its line with the placeholder line.

Repeated stars: enter `del` and follow the prompts for star and frame numbers. Frame Fixer will present all lines with the repeated star in each of the frames you request. If the PSF of the star is good (no overlapping defects) and the extra centroid is actually just a nearby defect, you should delete only the line of the bad centroid. If you are not sure if this is the case, go ahead and delete both lines, then follow the Frame Fixer prompts again to `add` a placeholder line for that star in that frame.

Saturated reference stars: these will not be flagged by the Check-Step above, but you should still remove them at this point in the reduction. Enter `add` at the prompt and enter the saturated stars and frames in which they are saturated. Frame Fixer will replace those stars' `*.centroid.*` lines with placeholder lines.

If you would rather fix the frames manually, quit `centroidframes.py` here by entering `q`. When you are ready, enter `centroidframes.py` at the command prompt and jump to Step 3 (safer option) or Step 5.

(Stage 2) Step 5 Generate final `*.sexout` and `*.hinfo` files.

This final step requires no user input. When complete, it reports that the `sexout` and `hinfo` files were compiled.

This step generates two files required for later stages of the pypline (such as `genpif` and `piplot.pro`). It organizes the centroid positions by concatenating the `*.centroid*` files into the `*.sexout` file. It also organizes the frame dates and exposure times by reading the `*.o.fits` headers (specified by `listfile`) to create the `*.hinfo` file. Both file names begin with the star's 7-character short name; for example, for GJ1005AB these files are `gj1005x.sexout` and `gj1005x.hinfo`.

It is critical here that your `*.centroid*` files follow the golden rule above. If any of them have missing or extra stars, `centroidframes.py` will not let you proceed with this step until they are corrected.

Stage 3 Process the measurements to prepare for parallax solving

Now that the centroid positions of the pi star and ref stars are measured and organized, we prepare the data to solve for parallax. To begin Stage 3, at the terminal enter: `parallaxprep.py`

The steps that follow must proceed in order. Unlike in previous stages, if you quit midway through you must restart again at the beginning. If you need to restart or redo this stage for any reason, you do not need to delete any files or do any other preparation.

As the program begins, it deletes any existing `coord.*` and `var.*` files and `coordbak` and `varbak` directories. It will alert you before it does this, and you will have the option to quit so you can save your existing files somewhere else. Files in any subdirectory will be safe from this process as long as that directory's name does not begin with `coord.` or `var.` or end with `.coord` or `.var` (note the location of the dot/period).

(Stage 3) Step 1 Confirm the settings for parallax reduction.

This step launches immediately upon starting `parallaxprep.py`. You are presented with the settings required to generate the parallax factors and DCR-corrected coordinates and magnitudes. Unless you are making substantial changes to the setup (i.e., changing the reference star set), you can press `enter` to confirm the default settings.

- Telescope (`0.9m` or `1.5m`): Early in the astrometry program we took data at the CTIO 1.5m, but now we exclusively use the 0.9m.
- DCR correction (`yes` or `no`): This applies the correction for differential color refraction (DCR), which requires photometry for this reference star field. Set this to `yes` whenever possible.
- Filter (`V` or `R` or `I`): All the data used for this reduction should be in one filter, which must be given here for accurate DCR. The "new" and "old" V filters are both considered `V` here.
- Number of stars (integer): This is the sum of the pi and reference stars. Change this value only if you changed the reference star set for this reduction.
- Number of frames (integer): Should match the length of `listfile` and `*.hinfo` — if not, use `ctrl-c` to quit now, find any missing/extra frames, and update `listfile` and `*.hinfo`. This frame count is taken after excluding those marked `X` in `listfile.all`.
- coord.* files (`yes` or `no`): Determines whether or not `coord.*` files are generated. You need these files to solve for proper motion and parallax.
- var.* files (`yes` or `no`): Determines whether or not `var.*` files are generated. You need these files to solve for variability.
- Include stars with peak counts <100? (`yes` or `no`): This allows you to include the very faint stars in your frames, which tend to have poorly determined centroids. Set this to `no` (default) to exclude any ref stars that are very faint, which may lower the overall number of ref stars in some frames. Set this to `yes` only if your field is sparse and needs these stars in order to have enough frames.
- pi star (integer): This is the star number of the pi star (target star). The pi star should always have the highest number in the set, unless this is a resolved binary system.

(Stage 3) Step 2 Run genpif to get pi factors.

This step requires no input from the user. It builds the input file `geninput` for the executable `genpif`, which it then runs to generate the parallax factors (`*.pifonly` file), `coord.*` files, and `var.*` files (if requested). As the user, all you need to do here is look for the "Completed genpif!" note after running it.

(Stage 3) Step 3 Reject bad frames.

Run this step to identify any additional frames that merit exclusion from the parallax reduction. In some cases it will ask you to confirm removal. You should always approve frame removal here unless (1) you have very few frames overall (< 40 frames) or (2) you have another compelling reason to keep the those frame(s).

Reasons for frame removal:

- **Hour Angle >120 minutes**: To minimize the effects of the atmosphere on the stellar PSFs, frames must have been taken close to the meridian (hour angle HA = 0 minutes). Our correction for DCR allows some deviation from meridian, but it is not valid past HA of 120 minutes. Do not keep these high HA frames unless necessary.
- **Fewer than 5 refs:** A valid fit to the astrometric model requires at least five reference stars in addition to the pi star. Do not keep frames with fewer than five refs unless excluding them would leave too few frames for a valid reduction.
- **Ellipticity or no pi star:** If the ellipticity of the pi star is greater than 20%, or if its PSF is otherwise poorly shaped or distorted, that centroid is not a faithful representation of the pi star's position. Frames with a poor pi star are always excluded.

As you reject bad frames, cross them off with an `X` in the grid on your paper log sheet. When you reach the end of this step, `parallaxprep.py` will pause to allow you time to make these notes. Press `enter` when you are ready to move on.

(Stage 3) Step 4 Select the trail plate.

This step presents you with a list of frames that are good candidates for the "trail plate." The trail plate is the final reference frame:. To get the motion of the pi star over time, the ref stars in all the other frames will be scaled to their positions in the trail plate and converted from image coordinates to RA and DEC.

Use the list to identify a trail plate with hour angle as close to zero as possible *and also* full width at half-max (FWHM) as low as possible (ideally <1.50). There is rarely a perfect frame. Input the *frame number* corresponding to your chosen plate (not the frame file name).

Remember that you can redo `parallaxprep.py` and choose a new trail plate if you later discover that your choice was no good — such as if the wcs fit (next step) fails, or you see unusual contamination in the image.

(Stage 3) Step 5 Match and rotate the trail plate to 2MASS positions.

This step proceeds automatically after the trail plate selection (step 4). Stars are automatically detected in the trail plate, and their pattern is matched against 2MASS using `imwcs`. The fitting process is recorded in the file `wcsfit.out`, and the results of that fit are reported in `angle.out` as well as at the user's terminal.

Next, the results in `angle.out` are used by the `pltrotate` script to apply a rotation matrix to the trail plate. Several details of that process are printed to the screen.

Finally, a summary is given before `parallaxprep.py` exits. Use the summary to fill out the paper log sheet by noting the trail plate name and number, rotation angle, scale, and frames used so far in the reduction.

: FIGURE: example summary card. FIGURE: selecting a trail plate.

Stage 4 Solve for parallax

This stage computes a least-squares fit of the astrometric model to the data to solve for parallax and proper motion of the pi star. You may need to run this stage several times if you discover ref stars or frames that should be excluded from the final result.

The steps in this stage must proceed in order, but several of them (2a–2c) are optional, and you can start over at any point. Begin by entering at the terminal: `pisolve.py`.

(Stage 4) Step 1 Initialize files (env, pi.par, etc.)

Run this step first to create the initial files that GaussFit will need to solve the astrometric model.

The first time you run this step, it will create a backup directory (`coordbak`) and save the unaltered `coord.*` files there. If you run this step again, the program asks if it can overwrite your `coord.*` files with those fresh backup versions. You should always say `yes`, unless you have a compelling reason not to.

If you have two pi stars (resolved binary): You must complete this stage for the first pi star only, then again for the second pi star. Before you begin your first run of `pysolve.py`:

(a) Edit `LONG_NAME` in `star.settings` to reflect the name of the first pi star. For example, if your system is LP771-095ABC (with "A" separated from "BC") and you are starting with A, change `LONG_NAME` to LP771-095A. Later, you will change it to LP771-095BC.

(b) When you change to the second pi star later, the program will ask for the 2MASS coordinates and JD epoch of that second star. You may wish to locate those values now, before beginning the process.

The rest of Step 1 requires no user input. **If this is your first time running `pisolve.py` for this system, skip straight to Step 3 when this step completes.**

(Stage 4) Step 2a Remove frames from the reduction

Run this step if you have identified frames that need to be removed from the parallax reduction. Often these are frames you would have chosen to remove during frame checking (Stage 1) if you were aware of their issues.

You must *always* run Step 1 (above) before running this step. **If this is your first time running `pisolve.py` for this system, skip to Step 3 now without running this step.**

This step will prompt you for the frame numbers that should be removed from the reduction. You can specify them as a space-separated or comma-separated list. "Removed" frames are not deleted, but renamed with their frame number and `coord.` label reversed; for example, if you remove frame 100, this program changes `coord.100` to `100.coord`. This means you can check for removed frames by searching for files named `*.coord`.

When Step 2a completes, it saves the new set of `coord.*` and `*.coord` files to the backup `coordbak` directory. To generate a fresh set of `coord.*` files with no frames removed, you must begin the reduction again at `parallaxprep.py`.

How do you decide if a frame should be removed?

A frame should be removed from the parallax reduction if the positions measured in it are not representative of the stars' true positions. This can happen if, for example, the flat-fielding procedure left the sky background very uneven, or if there is a defect on the PSF of one or more ref stars[1].

Bad frames often stand out as outlier points in the final residuals (`stats` plot). Use the `*.stats.pi.residual.out` file to identify the specific frame corresponding to a given outlier point in the `stats` plot. Then evaluate the potential bad frame by viewing it in DS9 (using `dis` in `tagframes` in Python or IRAF) and using contour and radial plots to check the PSFs of the pi star and ref stars.

(Stage 4) Step 2b Remove stars from the reduction

You must run Step 1 before this step, and if you choose to run Step 2a then you must do so before this step as well. **If this is your first time running `pisolve.py` for this system, skip to Step 3 now without running this step *unless* your system is a resolved binary with two pi stars (see below).**

Run this step if you have identified stars that should be removed from the parallax reduction. The program will ask for the numbers of the ref stars to remove (give it a comma- or space-separated list). You can remove as many refs as you want, as long as you keep at least five refs in the final set (the parallax solution is not possible with fewer). Keep in mind that the parallax result is usually improved by including more reference stars.

Removing ref stars this way is *not* permanent. If you run `pisolve.py` again, your previously removed ref stars will be restored until you run Step 2b again.

How do you decide if a star should be removed?

A reference star should be removed if it is not stationary in the sky, as in the astrometry reduction we assume the refs are not moving. Ref star apparent motion might happen if the ref star is:

- nearby, with distance of 300 pc or less (parallax 0.003 arcsec or more). Check suspicious ref stars' distances using other databases such as Gaia, which you can load in Aladin with 2MASS images to help star identification. You should also check the photometric distances of the ref stars listed in the `starname.refphot` file (where `starname` is the star's 7-character short name).
- part of a binary or multi-star system. Sometimes this information is noted in the star's SIMBAD page.

[1] If a ref star PSF has a defect in one frame or a few frames, you could either (1) remove the frame(s) or (2) restart the reduction at `centroidframes.py` (Stage 2) and replace that defective star in those frame(s) with placeholder lines.

- very faint relative to the rest of the stars in the setup. Faint ref stars are not necessarily moving in space, but their PSFs are less well-sampled on the CCD, which leads to poorly determined centroids.
- very near to the edge of the field in most images. Slight distortions in the shape of the CCD are more pronounced within ∼100 px of the edge of the field. These irregularities cause slight variations in the centroid positions of stars near the edges.
- overlapping with another star's PSF in the image. When the two PSFs overlap, frame-to-frame variations in seeing cause variations in their centroid positions.

Ref stars with significant motion are often, but not always, highlighted at the end of `piplot` ("Plot" Step below) as having high mean residuals.

As you remove ref stars, take care to leave the final setup as balanced as possible. Refs are more valuable when they are brighter and nearer (on the image) to the pi star, and a good final setup has refs evenly distributed in all directions. Remember that you must have at least five refs to run GaussFit (Step 3), and that more is usually better.

If you have two pi stars (resolved binary): Always remove the pi star that you are *not* reducing yet. For example, if your system is LP771-095ABC and you are finishing star A now, you must remove the star corresponding to BC. If you finished star A and intend to change the pi star to BC in the next step (2c), remove star A now.

(Stage 4) Step 2c Change to a different pi star

Only execute this step if you want to complete the reduction for a different star in the field, for example if you have two pi stars (resolved binary) and want to switch to the second pi star. You *must* remove the first pi star before beginning this step, otherwise it will be used as a ref star.

This step asks you for the name, 2MASS RA and DEC (in degrees), and 2MASS JD epoch for the new pi star. The program will overwrite these values in `star.settings`, so before you enter anything you may wish to open that file in another window and note the values of `INPUT_RA`, `INPUT_DEC`, and `JD`, in case you want to change back to the original pi star later.

If your system is not resolved in 2MASS (the two sources are a single star in 2MASS), just enter the same values already listed in `star.settings` for `INPUT_RA`, `INPUT_DEC`, and `JD`. Alternatively, you could skip this step, and instead edit `star.settings` to with the new star's `LONG_NAME` and star number for `PI_STAR`.

(Stage 4) Step 3 Run GaussFit to solve the parallax

You must always run Step 1 (above) before running this step. If you need to remove frames or stars (Steps 2a or 2b), you must also do that before running this step.

This process will run GaussFit to fit the astrometric model to the data in the `coord.*` files, yielding the proper motion and parallax of the pi star. The program will print substantial output to the terminal for a couple minutes (longer if there are more frames in the reduction). You will not need to provide any input. When it is complete, move on to the `piplot` step below.

(Stage 4) `piplot` Make the parallax plots

This step is not part of the Python suite: it requires running IDL to generate the final plots and data files for the astrometry reduction. You must complete Steps 1–3 of `pisolve.py` before running this step.

It is recommended to run this step in a second window, so you can keep IDL running if you need to re-run `pisolve.py`. In that case, when you open the new window remember to `cd` to the location of your current `pi.YYYY.MMDD` directory *before* launching IDL.

(a) Launch IDL at the command line: `idl83`

(b) In the IDL interface, create the parallax stats plot: `piplot`

(c) The `piplot` program will then ask you a short series of questions to classify the system and its reduction:

- Setup field: this will always be `1`, unless someone created a new reference star setup for this system. View the `*.stats.plot.ps` file from the previous reduction (located in that reduction's `pi.YYYY.MMDD` folder) to get the number of the setup you are using. If you changed the ref star setup for this reduction, augment that previous setup number by 1.
- Your initials: enter three characters here to indicate your name, so we have a record of who completed this data reduction.
- Parallax status code: locate this system in the astrometry observing list and note if it has been published previously. That determines the first character of this code:
  `f`: final parallax, for a system that RECONS has not published before.
  `u`: updated parallax, for a system that RECONS has published at least once.
  `r`: relative parallax, for a system with no ref star distance correction applied (in this case there will be only one line shown in upper left of the `.stats.plot.ps` plot).
  `e`: early parallax, for a system with less than two years of data or fewer than 30 morning and 30 evening frames (noted before the VRI columns in the observing list).

  If the residuals show a clear perturbation (PB), add `b` as a second character in the status code. If the residuals are extremely scattered (varying by > 0.02 arcsec from epoch to epoch), add `m` as a second character instead.

  For example, an updated parallax with a perturbation would get code `ub`. A result for a system with no previously published parallax (and no perturbation) would get code `f`.
- Sample: in the astrometry observing list, this is the six-letter all-caps code after the photometry and spectral type columns. For example, this could be `SURVEY`, `VARIAB`, `MASSIF`, etc.

If you make a mistake when entering this information, run `piplot` once more to try the prompts again. The files it generated previously will be overwritten.

(d) When `piplot` finishes, it will generate a plot, and also print some output to help you assess the quality of the reduction:

- Mean exposure time — note this on your log sheet. Longer exposure times generally improve the quality of the astrometry.
- Number of frames — note this on your log sheet as well, under "frames used for reduction."
- Reference star mean residuals — this table is a summary of how well the ref stars matched our assumption that they show no motion.

. talk about the piplot output (mean residuals)... write it down on the log sheet....

If you decide that some reference stars or some frames need to be removed from the reduction, restart `pisolve.py` at Step 1 and this time choose Steps 2a or 2b as needed. Removing ref stars is not permanent — they are all restored every time you begin `pisolve.py` again. This means you can run this stage of the reduction many times with different ref stars removed to see which combination gives the most precise or reliable result.

FIGURE: high mean residual ref stars

(e) When you are done with IDL, exit it with this command at the IDL prompt: `exit`

The final results of this stage are the `stats` plot and `nightly mean` plot: `YYYYMMDD.starname.stats.plot.ps` and `YYYYMMDD.starname.nightly.mean.ps`. Data files with the data shown in these plots are also generated as ....name these files..... Finally, the `YYYYMMDD.starname.parallax` file has been generated with a summary line for the full astrometry results file.

If you have two pi stars (resolved binary): When you are satisfied with your result for the first pi star, begin `pisolve.py` again with the focus on the second pi star. This means after you complete Step 1 you should remove the first pi star at Step 2b, and use Step 2c to change to the second pi star, before proceeding to Step 3.

(Stage 4) REF Refresh `coord.*` files

(Optional) This command copies fresh `coord.*` and `*.coord` files from the backup directory `coordbak`. Note that you will rarely need to do this step, as it happens automatically every time you start over at Step 1.

Stage 5 Solve for variability

The final stage of the pipeline determines how much the pi star is varying photometrically with respect to the reference stars. This procedure involves fitting the star magnitudes from all the frames (measured when we measured the centroids) and computing a least-squares fit of those data to a variability model. The fitting program used is called GaussFit. The fit minimizes frame-to-frame variability from non-astrophysical sources, such as the sky conditions and background, so we can assume the signal seen in the pi star's light curve is entirely from the star itself.

The stage proceeds very similarly to Stage 4 (`pisolve.py`). You will initialize files, remove any bad frames or bad stars, then fit the data and plot it. The procedure for creating the final plots will be different.

The steps in this stage must proceed in order, but a few of them (2a–2b) are optional, and you can start over at any point. Begin by entering at the terminal: `varsolve.py`.

(Stage 5) Step 1 Initialize files (env.var, etc.)

Run this step first to create the initial files GaussFit will need to solve the variability model.

The first time you run this step, it will create a backup directory (`varbak`) and save the unaltered `var.*` files there. If you run this step again, it will ask if it can overwrite your `var.*` files with those fresh backup versions. You should always say `yes`, unless you have a compelling reason not to.

The rest of this step requires no user input. If this is your first time running `varsolve.py` for this system, skip straight to Step 3 when this step completes.

(Stage 5) Step 2a Remove frames from var reduction.

Run this step if you have identified frames that need to be removed from the variability reduction. Often these are frames you would have chosen to remove during frame checking in Stage 1 if you were aware of their issues.

You must *always* run Step 1 (above) before running this step. **If this is your first time running `varsolve.py` for this system, skip to Step 3 now without running this step.**

This step functions identically to Step 2a for `pisolve.py`. It will prompt you for the frame numbers that should be removed from the reduction. You can specify them as a space-separated or comma-separated list. "Removed" frames are kept, but renamed with their frame number and `var.` label reversed; for example, if you remove frame 101, this program changes `var.101` to `101.var`. You can therefore check for removed frames by searching for files named `*.var`.

When this step completes, it saves the new set of `var.*` files and `*.var` files to the backup `vardbak` directory. To generate a fresh set of `var.*` files with no frames removed, you must begin the reduction again at `parallaxprep.py`.

How do you decide if a frame should be removed?

A frame should be removed from the variability reduction if its measured magnitudes are not representative of the stars. This could happen if there is a cosmic ray on one or more stars, or if the flat-fielding procedure left the sky background very uneven.

Bad frames usually stand out as outlier points in the final Δmag plots (`var` plot). Use the `???` file to identify the specific frame corresponding to a given outlier point in the `var` plot. Then evaluate the potential bad frame by viewing it in DS9 (using `dis` in `tagframes` in Python or IRAF).

(Stage 5) Step 2b Remove stars from var reduction.

You must run Step 1 before this step, and if you choose to run Step 2a then you must do so before this step as well. **If this is your first time running `varsolve.py` for this system, skip to Step 3 now without running this step.**

Run this step if you have identified stars that should be removed from the variability reduction. Just as in Step 2b of `pisolve.py`, the program will ask for the numbers of the ref stars to remove (give it a comma- or space-separated list). You may remove as many refs as you want, as long as you keep at least three?? refs in the final set.

Removing ref stars this way is *not* permanent. If you run `varsolve.py` again, your previously removed ref stars will be restored until you run Step 2b again.

How do you decide if a star should be removed?

A reference star should be removed if it is varying in brightness, especially if the variation is astrophysical. Perfect reference stars should have constant (flat) magnitude over time. Measured variation could occur if a ref star is an eclipsing binary, or intrinsically variable, or if its PSF overlaps with another on the sky.

Ref stars *are* expected to be more variable as they become more faint; the expected trend is shown in Figure **??**. Remove any star that is more variable than expected by this trend (i.e., standard deviation shown in Figure **??** much higher than other refs at similar magnitude).

You should also remove any refs that show a cyclical signal in their light curve, even if they do not have a high standard deviation. The measurements shown in each light curve are relative to the other ref stars, which means a signal in one star will be mirrored in the other ref stars. Take care to identify the truly varying refs; you may need to run `varsolve.py` several times with different ref star combinations.

FIGURE: good and bad variability pi stars

(Stage 5) Step 3 Run GaussFit to solve variability.

You must always run Step 1 before running this step. If you need to remove frames or stars (Steps 2a or 2b), you must also do that before running this step.

This process will run GaussFit to fit the variability model to the data in the `var.*` files, yielding the Δmag over time of each star in the current setup. The program will print substantial output to the terminal for a few seconds (longer if there are more frames in the reduction). You will not need to provide any input. When it is complete, move on to the `varplot` step below.

(Stage 5) `varplot` (all stars) Make the variability plots.

Similar to `piplot`, this step is not part of the Python suite: it requires running IDL to generate plots and data files to show the variability results. You must complete Steps 1–3 of `varsolve.py` before running this step.

(a) Launch IDL at the command line: `idl83`

(b) In the IDL interface, create the interactive variability plot: `varplot`

The `varplot` display will initially look like Figure **??**. Interact with it by left-clicking on a star to show its light curve, then left-clicking again on the light curve to revert to the first display.

(c) Identify any ref stars that you may want to exclude from the final variability results. These will be stars with unusually high standard deviation, or stars with light curves that show cycles or flares. See Step 2b (Stage 5) above for more detailed descriptions of these cases.

(d) Identify any outlier frames that you may need to remove (refer to (Stage 5) Step 2a).

(e) Finally, save a copy of the standard deviation vs. magnitude plot with all the stars (none removed). Do this by left-clicking on any star to get its light curve, then right-clicking on the light curve to get the prompts (at the IDL command line) to save it.

(f) Exit IDL with this command at the IDL prompt: `exit`

(g) Rename the plot you saved to make its purpose clearer. The default name will include the pi star's short name, followed by the number of the star you clicked on. It is recommended to replace the star's number with the word "ALL" and leave the rest as default.

For example, if on 31 May 2022 we clicked on star 3 in the GJ1005AB field, by default the plot will be `20220531.gj1005x-3.var.plot.ps` and we would rename it to `20220531.gj1005x-ALL.var.plot.ps`.

(Stage 5) `varplot` (ref stars only) Make the variability plots (again).

With the variability of the entire setup (all ref stars and pi star) saved, we need to complete the process again to generate the final variability results.

This time, we exclude the pi star and any variable ref stars, keeping only a set of high-quality refs that have flat variability. Completing the least-squares fit of these stars' variability will let us compare the pi star against all of them together, as if they were one very well-sampled flat star.

(a) Run `varsolve.py` again and select Step `1`. When prompted, answer `yes` to copying `var.*` files from the backup.

(b) If you noted any bad frames to remove, do so now with Step `2a`. Otherwise, skip this step.

(c) Run Step `2b` and remove the pi star (or both pi stars if this is a resolved binary). If you noticed any variable or bad ref stars, remove them now as well.

(d) Run Step `3` to complete GaussFit on just the remaining ref stars.

(e) Open IDL again with `idl83` and launch `varplot`. The interface should now show only the flat reference stars you left in the set.

If you notice any ref stars that should be excluded after all, exit now (right click outside any stars) and begin `varsolve.py` again at Step 1. When you reach Step 2b again, remember to remove all the variable refs as well as the pi star(s).

If you are satisfied with your ref star set, left-click on the brightest one (furthest left) and right-click its light curve to save it.

(f) Exit IDL with `exit`. Then you may wish to rename the bright star's `var` plot by replacing its number with a term like "BrightRef". For example, if on 31 May 2022 we saved ref 1 as the brightest ref star for GJ1005AB, the file would have saved as `20220531.gj1005x-1.var.plot.ps` and we would rename it to `20220531.gj1005x-BrightRef.var.plot.ps`.

(Stage 5) DAT Generate a data file on one star.

With the variability established for the flat ref stars, we are ready to compare the pi star to that set.

Do not complete this step until you have completed every step that precedes it, and are satisfied that you have only non-variable refs in your set.

(a) At the terminal, run `varsolve.py`, and at its initial prompt enter the command `dat`.

(b) The program will then ask you for the pi star's number. If you are reducing a resolved binary system (two pi stars), choose one of the two pi stars for now. The program will generate a data file for the star you specified.

(c) Launch IDL with `idl83` and at its prompt enter `problemstar`.

A light curve for the star you selected will appear, and the program will prompt you for the star's name. The preferred format is the pi star's 7-character short name, followed by a hyphen and the word "pi". For example, for GJ1005AB (with short name gj1005x) we would enter `gj1005x-pi`.

If this is one component of a resolved binary, replace "pi" with the letter of the component ("a" or "b") or some other clear indicator of which component of the system is being plotted.

The name you specify in this step is printed on the light curve as well as the file name of its saved file, which is formatted with today's date and `.var.plot.ps` as in the `var` plots in the previous step. If you type the wrong name at the prompt, make sure to delete the bad `.var.plot.ps` file before re-running `problemstar`.

Finally, if this system is a resolved binary, complete steps (a), (b), and (c) again for the second component.

FIGURE: good and bad variability pi stars

Final Stage Saving results and printing

The data reduction is complete! It is time to clean up and catalog the results.

1. Verify that you are currently located in your `pi.YYYY.MMDD` reduction directory (*not* the system's main data folder) by entering `pwd` and checking that the path ends in `pi.YYYY.MMDD` (where `YYYY.MMDD` indicate today's date). Then delete the `*.o.fits` files in your `pi.YYYY.MMDD` folder.

2. Print the files needed to update the paper packet for this system. With `LONGNAME` = the pi star's full name, and `shortname` = the pi star's 7-character short name, and `YYYYMMDD` = today's date, those files are:

   `YYYYMMDD.LONGNAME.stats.plot.ps`

   `YYYYMMDD.LONGNAME.nightly.mean.ps`

   `YYYYMMDD.shortname-ALL.var.plot.ps` (page 2 only, showing the std dev vs. mag plot)

   `YYYYMMDD.shortname-BrightRef.var.plot.ps` (page 1 only, showing the brightest ref's light curve)

   `YYYYMMDD.shortname-pi.var.plot.ps`

3. On the `var` plot showing std dev vs. mag of all the stars, mark the pi star, the brightest ref, and any ref stars you excluded from the variability reduction. An example of this procedure is shown in page ??.

4. Verify that your log sheet is completely filled out. Then arrange your new packet pages in this order (top to bottom):
   (a) `YYYYMMDD.LONGNAME.stats.plot.ps`
   (b) `YYYYMMDD.LONGNAME.nightly.mean.ps`
   (c) `YYYYMMDD.shortname-pi.var.plot.ps`
   (d) `YYYYMMDD.shortname-BrightRef.var.plot.ps` (page 1 only)
   (e) `YYYYMMDD.shortname-ALL.var.plot.ps` (page 2 only)
   (f) your log sheet

5. Open the packet and get rid of any old `var` plot pages (old versions of (c), (d), and (e) above). Keep any astrometry pages, log sheets, and extra pages not described here. Three-hole-punch your new pages such that they align with the rest of the packet, then staple them to the front.

6. Open the file that ends in `.parallax`. The single line in this file summarizes this data reduction.
   (a) Add the pi star's variability to the field marked with dashes (starting at character 131), and replace the three dashes following it with your initials. If this system shows an astrometric perturbation (PB), add the code "PB?" or "PB!" to the end of the line (depending on how clear the PB is).
   (b) Open the latest parallax results file under
   `/nfs/morgan/users/thenry/public_html/CTIOPI/protected.dir/`
   and add your updated line. Note that the systems in the file are organized by RA.
   (c) Before closing the results file, add the K magnitude of this system to the line you just added. The K field follows the VRI filter.
   When you are done, the columns of your new line should align with the existing columns. Save and close the results file.

Appendix

Lookup table between old and new pipeline:

New pypline function	Old pipeline function
`setup.reduction`	`folder.processor`
	`star.processor`
`centroidframes.py` steps 1–3	`sss.var` options 0–3
`centroidframes.py` step 4	`sss.var` option 6
`centroidframes.py` step 5	`sss.var` option 4
	`headmet.cl`
`20220604.LHS3739BC.stats.residual.out`	`20220604.LHS3739BC.refs.stats` or `refs.highestresiduals`
`20220604.LHS3739BC.stats.pi.residual.out`	`20220604.LHS3739BC.stats.dat`
`nightly.mean.dat`	`20220604.LHS3739BC.nightly.mean.dat`

Index of potential issues

Housekeeping:

- make sure xpans is in your path
- get the redpi package installed to use iraf
- (Stage 0) Few `*.o.fits` files in the data folder: If very few frames have been taken, this system is not yet ready for an astrometry reduction. Use the observing list to verify that it has at least 30 morning and 30 evening frames.
- (Stage 0) If `setup.reduction` alerts you that images are missing from the astrometry catalog (see Figure 4), you will need to add those file names manually to `listfile.all` and `listfile.newframes` (described in § Stage 1 below).
- `tagframes.py`: DS9 does not launch, or images do not load in DS9, because there is already a "pyplineds9" window open elsewhere (such as in another remote session).
  Solution: The code will always connect to the first pyplineds9 window that was opened, so you need to close the DS9 window from the other session. At the Linux terminal, display the running processes from `top`, filtered to include only DS9 windows: `top -p 'pgrep ds9'` (note the grave accents). Within the `top` interface, note the process ID corresponding to the extra DS9 window, and kill it with the `k` command. For example, if the process ID is 434077, you would enter `k 434077`.

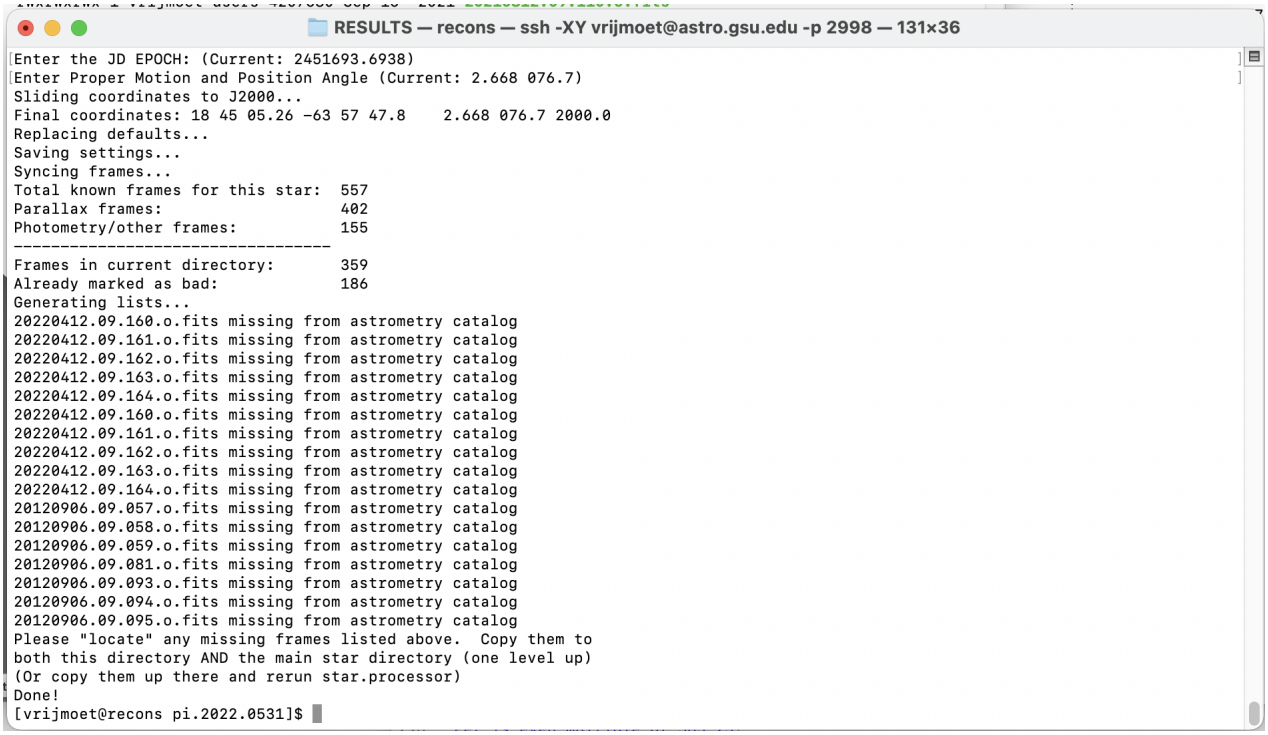

Figure 4: Example of output from `setup.reduction` when the astrometry catalog is incomplete. These frames are not missing from the data folder, but their file names will need to be added manually to `listfile.all` and `listfile.newframes` (see §Stage 1).

- `tagframes.py`: DS9 gives the error "The regions file temptags.reg cannot be loaded," and the code crashes.

  Solution: close DS9, then relaunch it by entering `ds9 -title pyplineds9 &` at the command line — or, if you are using Python (not IRAF), you can enter `tagframes.py`, select `dis` again, and let Python launch DS9 automatically.

- `centroidframes.py`: In the centroiding check output (step 3), one star is missing in a large set of consecutive frames (usually a pi star).

  Explanation: This happens because that star has moved more than its expected proper motion, so `autotag` placed the position too far away from the PSF. The frames need to be re-tagged, but doing so many one by one will be tedious.

  Solution: We can run autotag again, but adjust the taglist to focus on only these frames to be re-tagged. Choose a reference frame in the middle of the bad frame range — doing this will accommodate for the imperfect proper motion, so the issue does not occur again. Run `tagframes.py` and choose Step 1, then at the prompt specify your new chosen ref frame. Tag the stars again. Then when the code offers to continue to Step 2, quit the program with `q`. Open the file `taglist` and edit it to include only the badly tagged frames. Then run `tagframes.py` again and start at Step 3 (`autotag`). Finally, run `centroidframes.py` again from the top, and see if your re-tagging improved the centroiding.