Tom's Image Reduction & Photometry Guide

Introduction:

So you've decided you want to reduce and analyse your data from Greenhill Observatory? How exciting! But now you are wondering: 'how the hell do I achieve this'? That is where this guide comes in. I will go through the steps needed to go from 'hot of the telescope' data to a light curve. In this document, I will be specifically focusing on using my pipeline and PySIS, though it should be noted another pipeline written by a visitor exists (ask JP/Andrew for more info on this). My pipeline can be found pre-installed on Keck Machine in the 'TomUtils' folder or on Github here:

https://github.com/tjplunkett/TomUtils

There is further documentation in this folder about how each script works, which I recommend reading. This document is the hands-on guide and will gloss over some details already listed.

Before beginning, go into the terminal and activate the reduction environment ('red' on Keck Machine) and change into the TomUtils directory!

Part 1 – Image Reduction/Calibration:

The first phase of any good analysis is correcting your images for sources of noise. This typically involves 3 main steps: bias-subtraction, dark current subtraction and flat-fielding. I will not go into full details of what these are, instead you can read Howell (2011) for background. Here I present your 2 choices for reduction using my pipeline, both using Prose (Garcia et al. 2021).

Automatic Reduction:

If you are lazy, then you are in luck! I am a kind person and have already written an automated script that reduces all new data taken from the Harlingten 50cm each day via a Cron job. This achieved through the scripts 'autoReduce.py' and 'ReductionBot.py', which live in the 'TomUtils' folder.

All you need to do is copy the night's folder (i.e 20231008) containing your data from the Planewave computer to the **'/home/obs/Work/Data/[Year]**' folder via WinSCP. Then, the script will go through the images and search the **'/home/obs/Work/Calibration**' directory to find the closest flats by date (before the images) and darks for the maximum exposure time image in the night's folder. It will also build a bad pixel map for the CCD from this set of darks and flats. Finally, the images are reduced by scaling the darks down to match each individual image exposure time. These reduced images will be outputted to object folders, with sub folders for filter in **'/home/obs/Work/Reduced/[object]'**.

Please note, most of the time this is best for a quick look at data, but more thorough reduction may be necessary (especially if there is unusual noise/systematics in your data or the calibration frames aren't great).

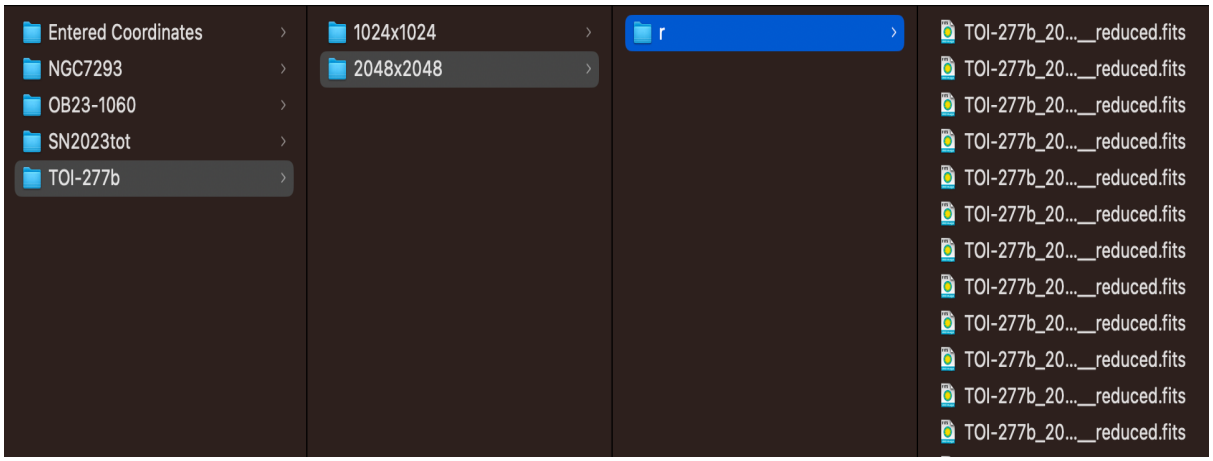


Figure 1. Example of the structure of the 'Reduced' folder.

<u>Manual Reduction:</u>

If you want more control over the calibration frames used to reduce your data, you can use the script 'reduce_H50.py'. First, copy your raw science frames to a new folder. Then copy in the desired calibration frames into this folder. Finally, type:

python reduce_h50.py [path2fits] [depth]

This will perform the reduction and save the new files into new sub-folders (labelled by image dimension) in the original directory.

```
(red) a4-cf-99-84-4d-5c-dyn:TomUtils-main tp22$ python reduce_H50.py /Users/tp22/Desktop/TPHE 0
RUN Parsing FITS: 100%|██████████| 103/103 [00:00<00:00, 178.30images/s]
RUN Parsing FITS: 100%|██████████| 103/103 [00:00<00:00, 2109.79images/s]
          date       telescope filter   type     target  width  height  files
id
9   2023-04-12  PlaneWave 50cm      i   bias  HIP 39961   2048    2048     10
5   2023-04-20  PlaneWave 50cm      i   dark          A   2048    2048     10
6   2023-09-13  PlaneWave 50cm      B   flat  HIP 99570   2048    2048      4
8   2023-09-13  PlaneWave 50cm      V   flat  HIP 99570   2048    2048      6
7   2023-09-13  PlaneWave 50cm      g   flat  HIP 99570   2048    2048      6
10  2023-09-13  PlaneWave 50cm      r   flat  HIP 99570   2048    2048      7
4   2023-10-01  PlaneWave 50cm      B  light     TPHE-A   2048    2048     15
2   2023-10-01  PlaneWave 50cm      V  light     TPHE-A   2048    2048     15
3   2023-10-01  PlaneWave 50cm      g  light     TPHE-A   2048    2048     15
1   2023-10-01  PlaneWave 50cm      r  light     TPHE-A   2048    2048     15
RUN Parsing FITS: 100%|██████████| 43/43 [00:00<00:00, 136.51images/s]
INFO buidling bad pixels map
RUN 100%|██████████| 15/15 [00:06<00:00,  2.19images/s]
INFO buidling bad pixels map
RUN 100%|██████████| 15/15 [00:02<00:00,  7.49images/s]
INFO buidling bad pixels map
RUN 100%|██████████| 15/15 [00:02<00:00,  6.84images/s]
```

Figure 2. Example command line output using reduce_H50.py

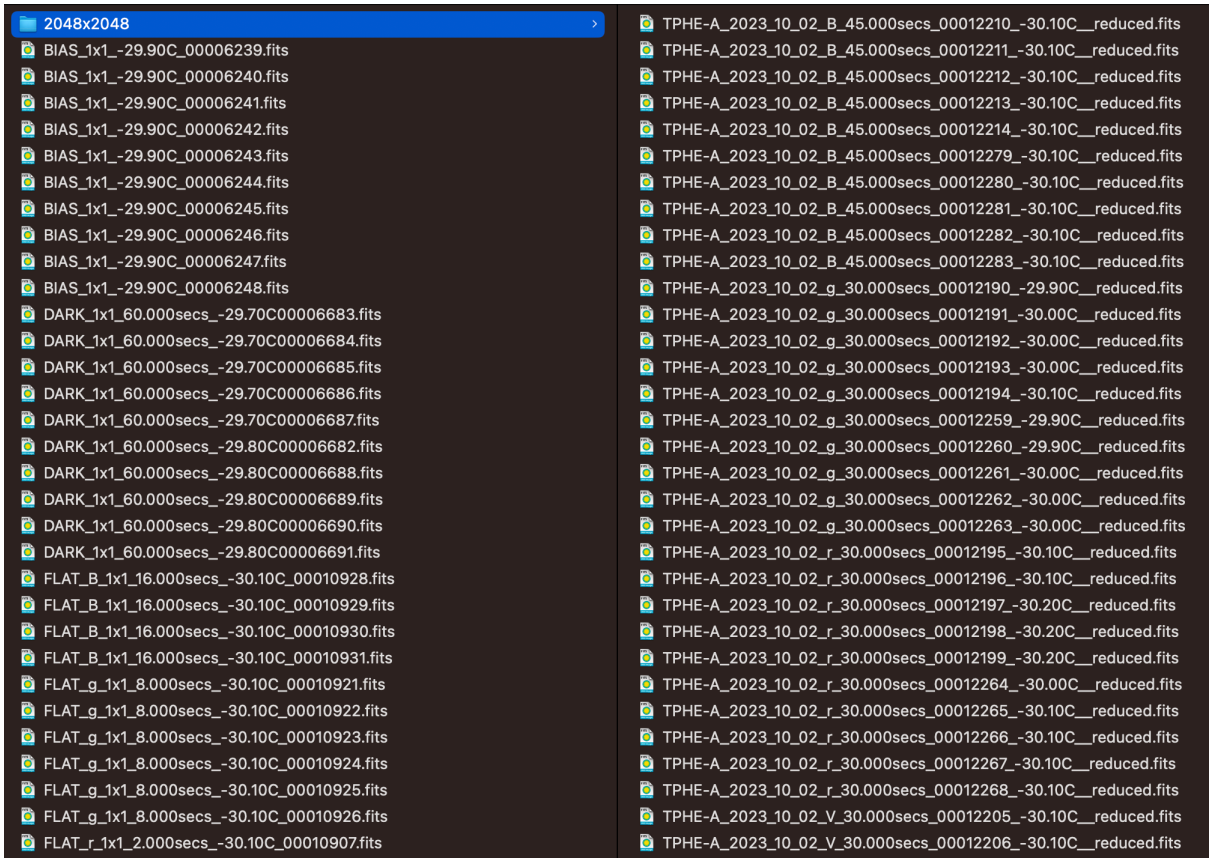


Figure 3. Example of the input and output folder structures for reduce_H50.py

Part 2 – Photometry:

Now that you have reduced your data, you are ready to perform photometry to get your light curves. But first, you must ask yourself: what type of photometry do I need? There are 3 types available to you currently: simple aperture, differential aperture and difference imaging.

Simple Aperture Photometry (SAP):

SAP is the most basic photometry method and most prone to error. It involves defining a circular region (the 'aperture') around a target and then summing up the count values, scaling to electrons and subtracting the local background noise to find the flux/magnitude. I therefore urge caution when using it, as you will need to use external catalogues such as GAIA, SDSS, etc. to determine zero points for each observing session (or even each image for best accuracy).

To perform SAP, you can use the 'forced_phot.py' script. Type:

python forced_phot.py [path2folder] [depth] [astr] [vis]

If you select 'y' for [vis], this will display a sample image for you to visually choose the centroid of your target. This is done by double clicking on the image. If you select 'n', then be prepared to type the (x,y) coordinates of your target into the terminal.

The outputted file will be in .csv format, containing the JD, aperture flux, instrumental magnitude, errors and an approximate calibrated magnitude.

Summary:

1. Run the forced photometry script (forced_phot.py) on reduced images folder.
2. Select target position either by clicking on image or manually entering coordinates.
3. Profit???

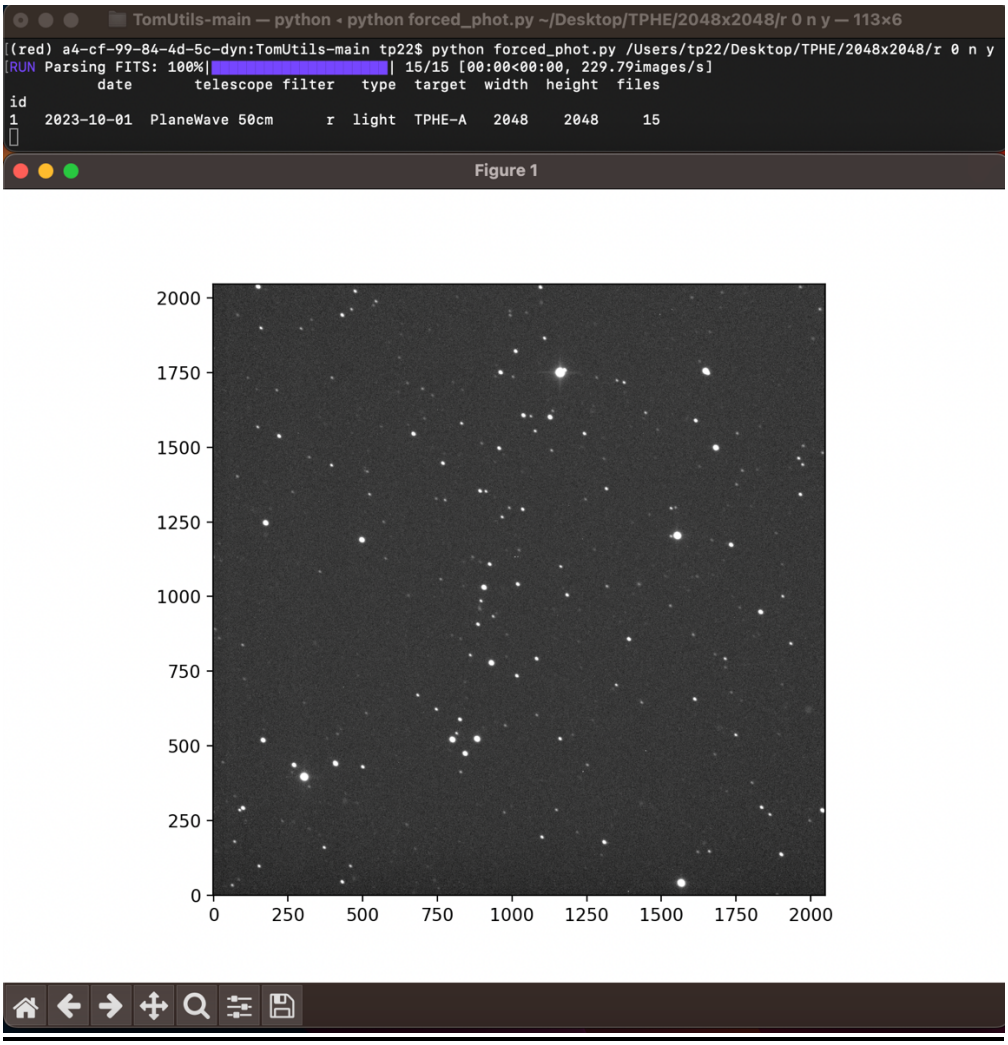


Figure 4. Example of the visual target selection from forced_phot.py

TPHE-A_2023_10_02_r_30.000secs_00012195_-30.10C__reduced_forcedphot

xcenter	ycenter	aperture_sum	JD	Phot_Aper	Inst_Mag	SNR	Airmass	Mag_Aper	Mag_Er
497.5536046440710	1189.7670321913300	241121.46198425000	2460219.9154099100	267808.51160411700	-9.876757802769430	497.86984039745200	1.35352770360976	11.96314248761690	0.07014643709270820
497.02267770112800	1190.6332519115100	241293.34320124100	2460219.915860710	267863.4483385570	-9.876980501757700	497.79457808490500	1.35110760665204	11.96333120511150	0.07014602588515330
499.74268927458100	1190.4686436915700	243889.755414601	2460219.916311510	271326.97489738200	-9.890929295141860	501.53999429042500	1.34881379195802	11.94977236022530	0.07014520055762910
500.0266426957760	1191.2746358182000	240436.23082311200	2460219.916762140	266525.0175935290	-9.871541814927	496.18930836562800	1.34653226917502	11.969547699313200	0.0701454205034832
500.8397522087250	1190.9847503898900	243642.4445035320	2460219.9172128400	270756.33223657800	-9.888643419087320	500.73912399272100	1.34426435229601	11.95283164102240	0.07014450527323450
487.6848215504400	1190.0416896753500	239883.74638454000	2460219.9965162000	267112.266121089	-9.873931442730010	497.8549131509320	1.08046002332995	12.012390353303900	0.07010378797386790
488.17730762681600	1190.325176159560	235324.35767645200	2460219.9969671600	260819.67519556200	-9.848047737546880	490.8132582618850	1.07955258452768	12.038428323083400	0.07010449375399400
487.79085452186200	1189.9499383342600	237326.0228707710	2460219.997417910	262831.5567119560	-9.856390631507540	492.50903687518400	1.07871556198583	12.030227722954900	0.07010417426310300
488.0406457320670	1190.4561479012500	233880.534928015	2460219.997868850	257741.08330620300	-9.835155987101400	486.514722767944	1.07786304854041	12.051607294646700	0.07010478526436700
488.6328004481910	1190.412007902450	233444.0294174550	2460219.9983197700	256517.654435419	-9.829990013526430	484.6578503140980	1.07701463920507	12.056917497808700	0.07010489924502350
491.8950622871250	1182.7578976416900	261572.19465604700	2460220.046778480	244815.7811470360	-9.779295387123350	436.9713852296390	1.01508222026929	12.118140635430900	0.07010357098642060
493.00372082041500	1182.348071155010	267039.6672181140	2460220.0472291700	251245.90958686200	-9.807444363070070	443.8339177002230	1.01474023237108	12.090049797426900	0.07010238025150560
493.1870997966080	1182.6361249483900	274543.53469753400	2460220.0476802300	260683.1499071980	-9.847479263342360	454.1682043320820	1.01441889893729	12.050069523838300	0.07010070937769420
493.05749413465000	1182.3760789736800	266560.12921548100	2460220.048131270	249661.52237790600	-9.8005758992294	441.43157686170900	1.0141020744731	12.097026748110200	0.07010269230449110
493.23002104707600	1182.8873203149300	262374.05533352400	2460220.0485822700	242951.3942373420	-9.770995352485060	432.98049113300400	1.01378905386826	12.126660508357300	0.07010409432480310

Figure 5. Example of the output .csv file from the forced_phot.py script

Differential Aperture Photometry (DAP):

This type of photometry is best used in cases where you have an isolated target and absolute calibration is not needed (i.e you just need to know how the light curve changed relative to some starting value). Hence, this is commonly used for transit photometry when searching for exoplanets.

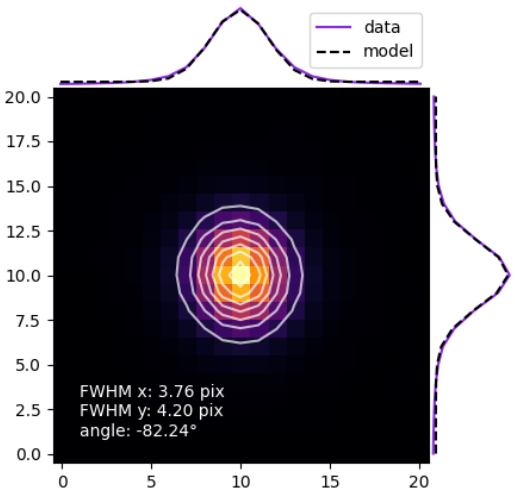
To perform DAP, type:
python diff_phot.py [path2calibratedfiles] [depth] [gaia_flag]

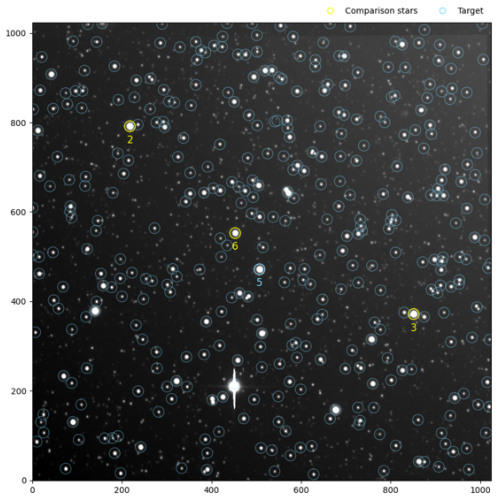
The gaia_flag argument informs the code on how you want to perform the photometry. In the case of ‘n’, this will use the algorithm defined in Broeg et al. 2005 to find the ‘optimal’ comparison stars for the target. If you choose [gaia_flag] = ‘y’, then GAIA DR2 is queried to find the 5 closest stars by colour and magnitude for comparison stars (in the case where differential refraction causes systematics).

This code will output files to a new subfolder called ‘phot_outputs’, containing a .csv file with your light curve (and other systematics), along with a basic summary plot of the observations, image showing the chosen comparison stars and a plot of the PSF model.

Summary:

1. Run differential photometry script (diffphot_pipe.py)
2. Select your target via number listed beneath star. Zoom in if field is crowded.
3. Profit?





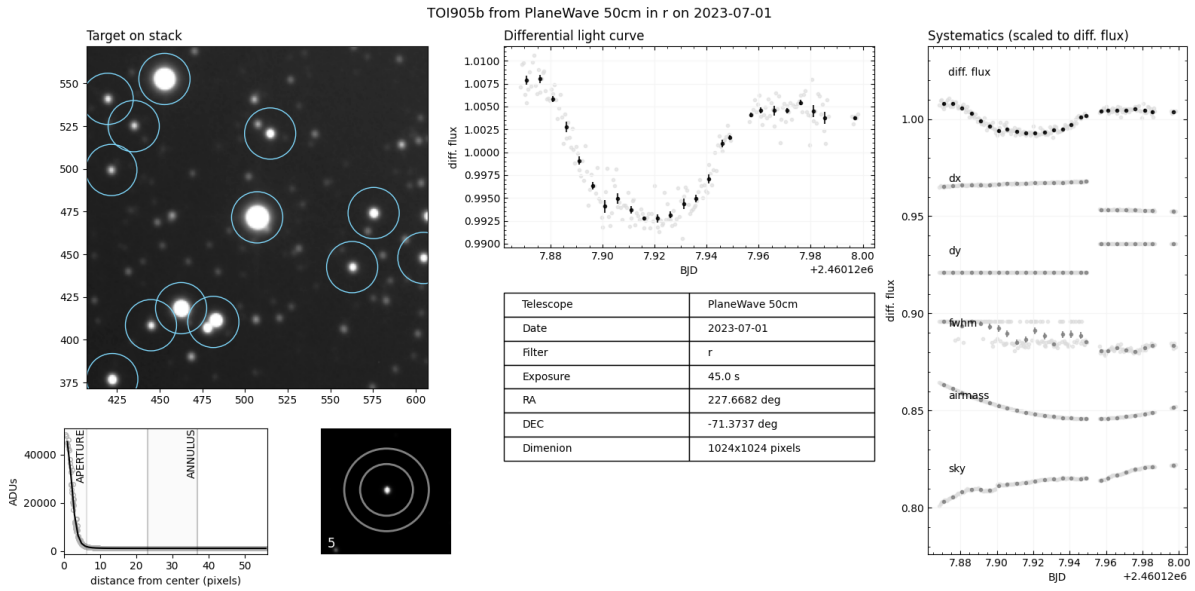


Figure 6. Output images from the DAP script (diffphot_pipe.py).

Difference Imaging Analysis (DIA) Photometry:

The final form of photometry is best used in situations where your target is located in a very crowded field (such as the Galactic Bulge) or close to another object. Again, you will only get a relative light curve (calibration is possible with some effort). Note that this method is also the most intensive and sensitive to user choices.

To start, we need to do some formatting and manipulation of our files to be compliant with the PySIS pipeline. Copy your reduced images to the **'/home/obs/Pysis_3.1'** folder into a new sub-folder (maybe named with your target ID). Then, type:

python prep4pysis.py [path2images] [prefix] [verbose]

The prefix is a 9-character string for renaming files to be PySIS compliant. This will separate the files into 2 folders called ‘Left’ and ‘Right’ to indicate the side of meridian they are on (arbitrary). Then, the script will flip one set of images to match the other and copy the remaining files to a final folder ‘All’. You can then proceed to run PySIS from this ‘All’ folder. To do this, type:

cd [path2Allfolder]

../../bin/reduce4 -e [Prefix+LeadingZeros] -s Squid -I ds9

This will commence the PySIS photometry process. The first step in the photometry is creating a reference image, which is usually a stack of the best seeing and low background images (of similar exposure time). However, the automated choice of the best frames seems to fail more often than not, due to light cloud making the image seem to have better seeing than it really does. I therefore recommend opening the seeing file and choosing manually. You will then enter the selection images as space separated list.

Then, the images will be registered (i.e shifted and transformed to have the same x,y coordinate between all frames). You will then choose the ‘lens’ position, which is the position of your target on the frame (this comes from microlensing terminology). You will need to open one of the ‘_interp.fits’ images via DS9 to get the coordinates.

Now you can sit back as the magic happens. The pipeline will now perform image subtraction, create a PSF model from your images and then perform aperture and PSF photometry on the difference images to get relative photometry of your target. The photometry is outputted to the files '[object_id].report' and '[object_id].pysis'. The difference images are labelled 'conv_[object_id].fits'. Ask JP for more information about the other outputs within the folder.

Summary:

1. Copy reduced images into a new subfolder in the **'/home/obs/Pysis_3.1'** directory.

2. Run 'prep4pysis.py' on this subfolder.

3. Now change directory to the newly made 'All' subfolder inside the original.

4. Start PySIS pipeline.

5. Use DS9 to choose images for the reference.

6. Use DS9 to find the x,y coordinates of the lens on an 'interp' image.

7. Profit???