UTGO Image Reduction & Photometry Guide

Introduction:

The UTGO pipeline can be found pre-installed on Keck Machine in the 'UTGO_Pipeline' folder or on Github here:

https://github.com/tjplunkett/UTGO_Pipeline

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 UTGO_Pipeline 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 (2006) for background. Here I present your 2 choices for reduction using my pipeline, both using Prose (Garcia et al. 2021).

Automatic Reduction:

There is an automated script that reduces all new data taken from the Harlingten 50cm each day via a Cron job. This achieved through the scripts ‘update_calibs.py’, ‘autoReduce.py’ and ‘ReductionBot.py’, which live in the ‘UTGO_Pipeline’ folder.

To use:

- 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.

- Each day at 7 am, the reduction begins on any new data.

- These reduced images will be outputted to object folders, with sub folders for filter in **‘/home/obs/Work/Reduced/[object]’**.

Disclaimer: more thorough reduction may be necessary (especially if there is unusual noise/systematics in your data or the calibration frames aren’t great).

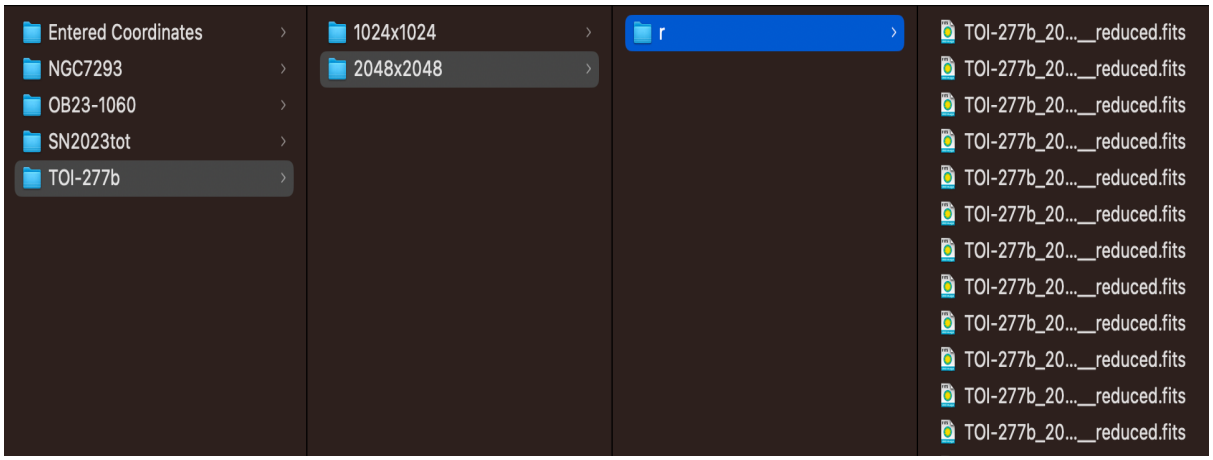

Figure 1. Example of the structure of the ‘Reduced’ folder.

Manual Reduction:

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

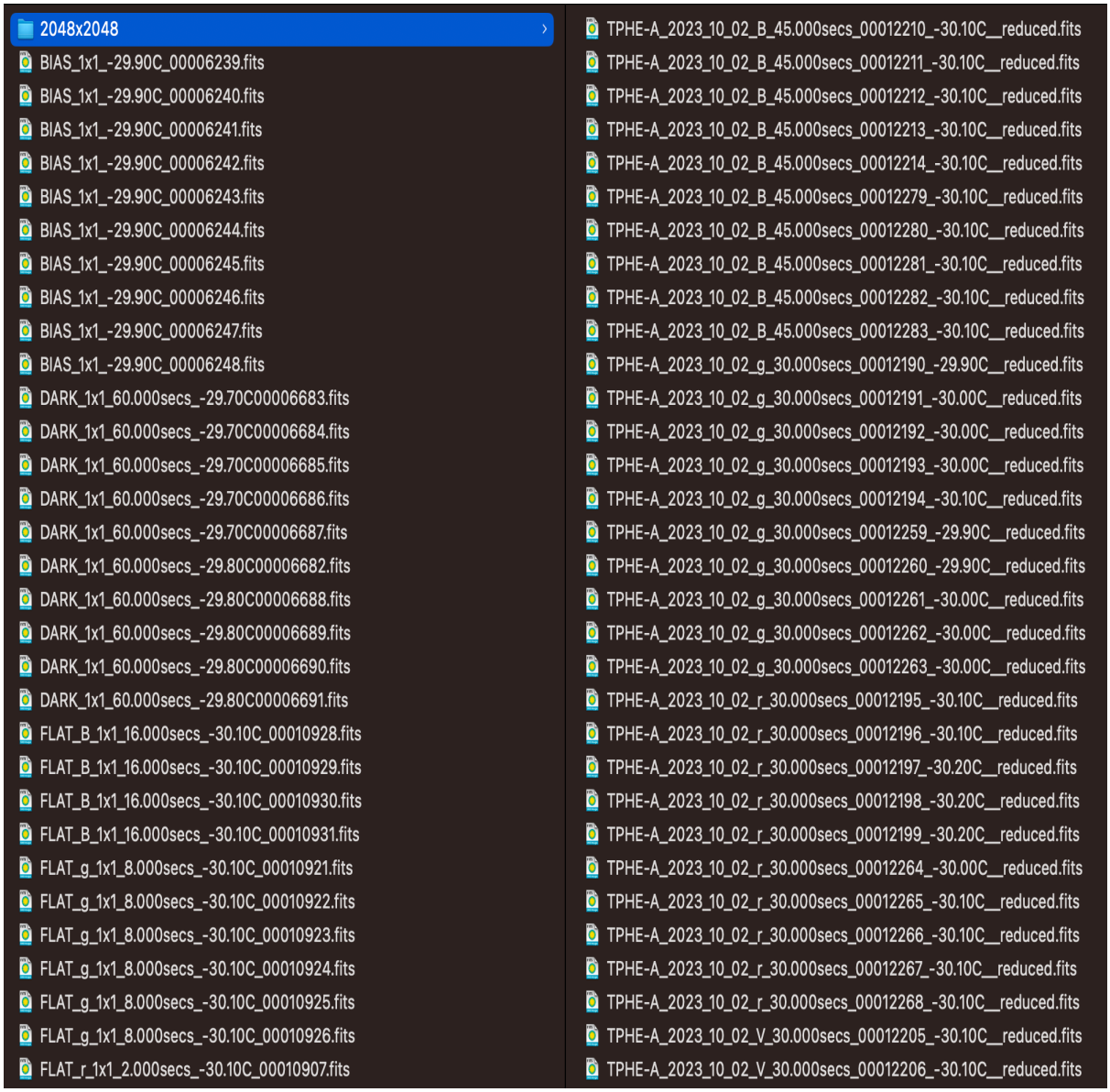

Figure 3. Example of the input and output folder structures for reduce_H50.py

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. To start, change into the **'UTGO_Pipeline/Photometry'** directory.

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.

To perform SAP, you can use either the 'aper_phot.py' or 'forced_phot.py' script. The 'aper_phot.py' can be used when your target has good SNR across all frames, which might not be the case for some transients. To use:

cd UTGO_Pipeline/Photometry

python aper_phot.py [path2images] [ID] [autoAperture] [verbose]

This will output a calibrated light curve to the 'Phot_Outputs' subfolder, using SExtractor or PhotUtils to extract a catalogue and find zeropoints for the image (using the same aperture). Note, the auto-aperture will be the one that maximises the SNR in the first frame.

The ‘forced_phot.py’ script can be used to perform forced photometry anywhere on your image. Note, it does not centroid your aperture, so be very careful with placement. To use, 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 calibrated magnitude.

Summary:

1. Run the aperture (aper_phot.py) or 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???

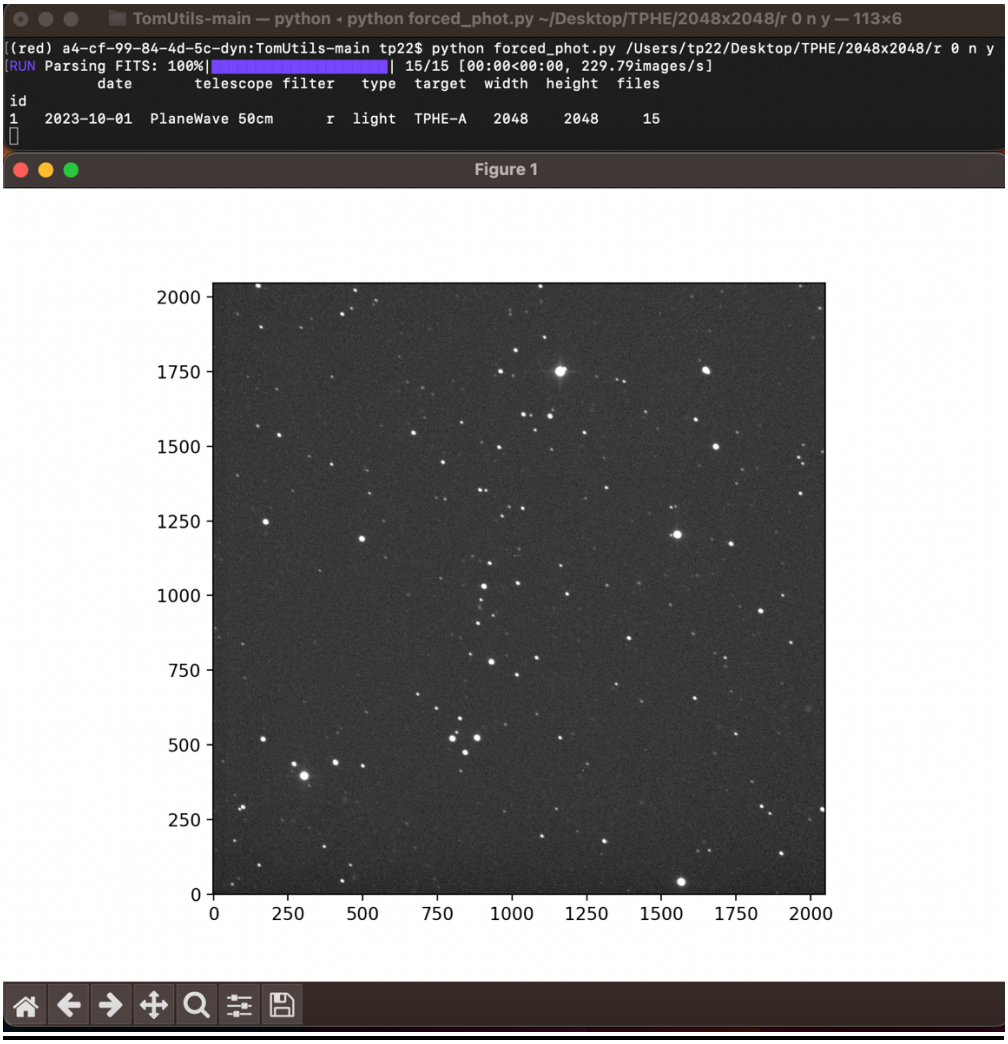

Figure 4. Example of the visual target selection from forced_phot.py

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	12.01239035303900	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] [auto] [threshold] [obs_date]

The [auto] argument informs the code on how you want to perform the photometry. In the case of 'y', this will present two options. The first will use the algorithm defined in Broeg et al. 2005 to find the 'optimal' comparison stars for the target. Otherwise, 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). If you choose 'n', this will give you manual choice of your comp. stars by entering their IDs (shown in a pop-up reference frame window) in a comma separated list.

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 (diff_phot.py)
2. Select your target via number listed beneath star. Zoom in if field is crowded.
3. Profit?

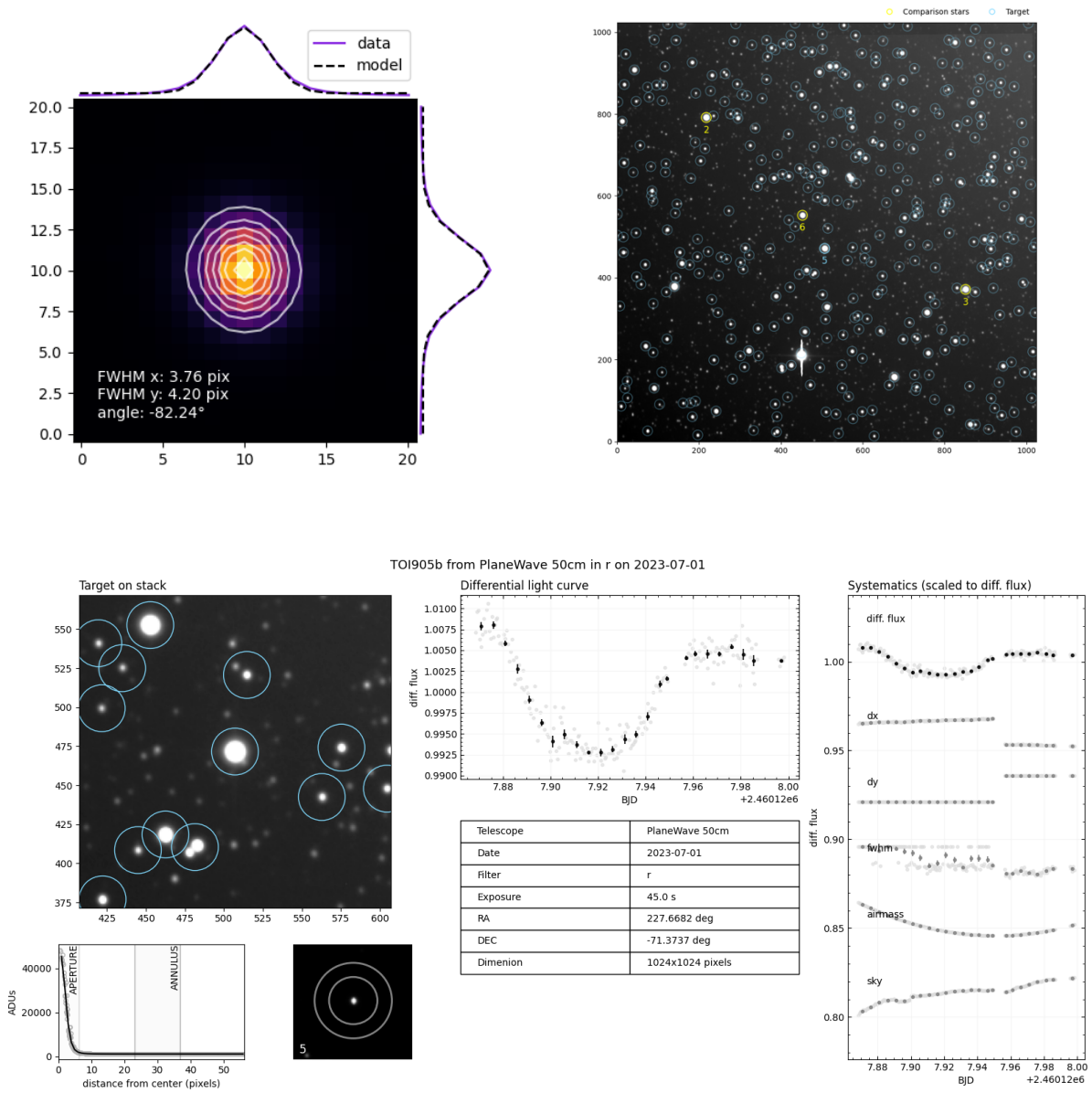

Figure 6. Output images from the DAP script (diffphot_pipe.py).

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. To do this, we have two choices. If you want an industry-standard, tried and tested code that is optimised for microlensing, then you should use the pySIS code. However, the process of using this code with our data is lengthy and takes some experimentation. A guide is included for this in a separate file.

If you want an easy/quick (but possibly less thorough/clean) option, you can use the 'dia_phot.py' script instead. This has the benefit of being fully integrated into the existing pipeline and consistent with other photometry scripts. To perform DIA photometry, type:
python dia_phot.py [Path] [Visual] [AutoRef] [Stack] [Redo] [Verbose]

Here, the path flag is to directory with all the reduced files. If you would like the code to find the best frames (i.e lowest seeing and background) and stack them, you would use [AutoRef] = y and [Stack] = y. Otherwise, you can choose an individual file that you prefer for the reference by using [AutoRef] = n. The [Redo] flag tells the code if you want to redo subtraction (so should be set to 'y' if doing from scratch), otherwise only the photometry steps will be performed. The subtraction is performed in parallel, but can take a reasonable amount of time if the seeing is bad (i.e a larger kernel size is used).

The code will output the reference and registered frames in a new ‘DIA’ sub-directory, along with the photometry catalogue on the reference frame (in ‘ref_phot.csv’), a light curve for the target calibrated to GAIA DR2 (in ‘target_dia_phot.csv’) and a diagnostic plot showing examples of differencing for each night (in ‘diff_check.png’).

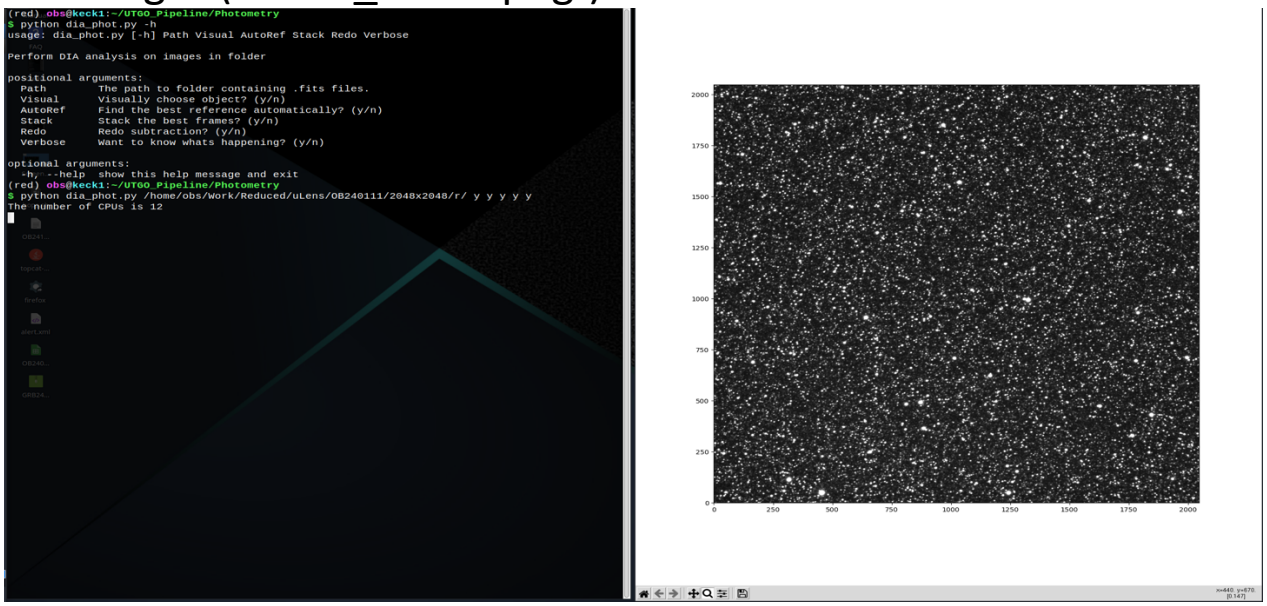

Figure 7. The visual selection option for dia_phot.py

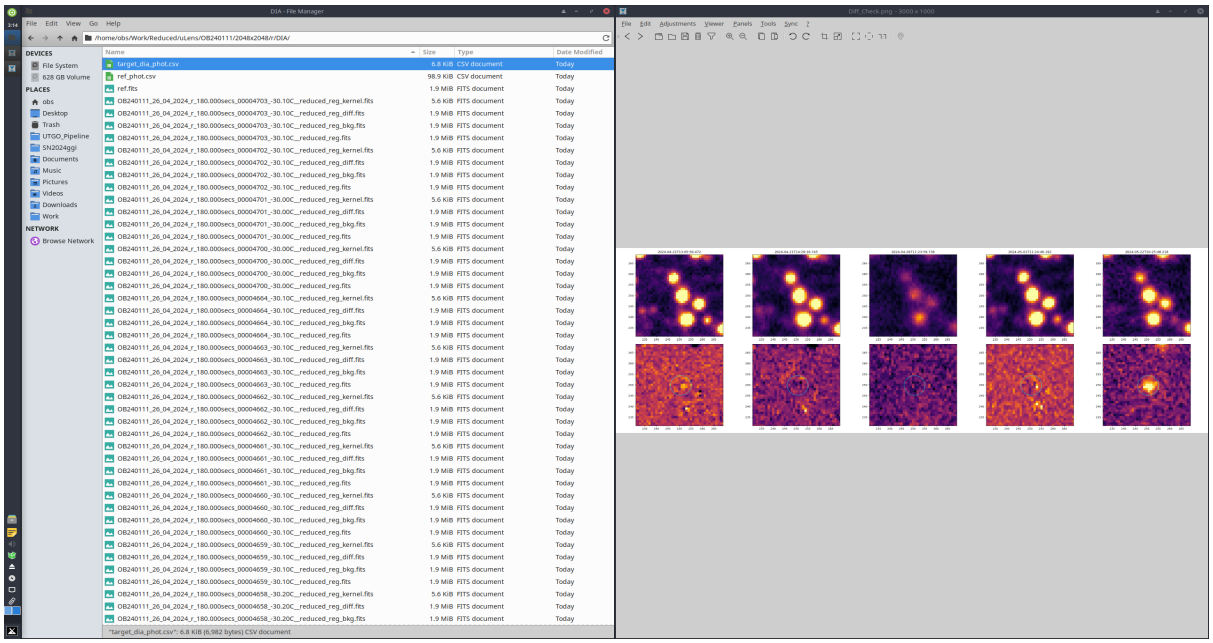

Figure 8. The output of dia_phot.py, including the ‘diff_check.png’ diagnostic plot showing the differencing for each unique night of observation.

Part 3 – Stacking:

If you have a very faint target with poor SNR, you may need to co-add multiple frames to get good measurements. To do this, type:

cd UTGO_Pipeline/Utils

python stacker.py [path2folder] [depth] [method] [period]

The method will either be ‘swarp’ or default to median stack. The period tells it how many images to stack per co-add, i.e if you pass 5 it will stack every 5 images or if you type ‘D’ it will stack all the frames from a night.