




VALIDATION OF THE PIPELINE FOR THE 1.4m MILANKOVIĆ TELESCOPE

A. Lalović , O. Vince , M. Stojanović  and S. Marčeta Mandić 

Astronomical Observatory, Volgina 7, 11060 Belgrade, Serbia

E-mail: ana@aob.rs

(Received: September 9, 2024; Accepted: November 11, 2024)

SUMMARY: In this paper, we present a pipeline written in Python v3.9 that performs reduction of CCD images, astrometric calibration, and photometric measurements for observations made with the 1.4m Milanković telescope, installed at the Astronomical Station Vidojevica in Serbia. The main driver for writing such a multipurpose pipeline lies in its intended use once the dedicated observations of transient objects with the Milanković telescope start as the follow-up observations contributing to the Vera Rubin Observatory scientific goals, but not limited to this specific need, since other follow-up observations had been conducted and some are still ongoing. The pipeline is run via the command line using different positional and optional arguments depending on the required operations (reduction, astrometry, photometry). There are also command-line arguments that allow the user to choose how to perform a certain operation (mode), or whether to use more PC processors to execute them. By default, only bias, dark, and flat-field corrections are performed, while other operations and modes are optional. In the paper, we will describe in detail each of the above mentioned operations, as well as the use of positional/optional arguments. We will also describe in detail the output FITS files that can be used to analyze the pipeline performance, as well as the final output tables with photometric measurements. Finally, we compare the pipeline performance with IRAF package procedures to test and validate the results.

Key words. Methods: observational – Methods: data analysis – Techniques: image processing

1. INTRODUCTION

The Milanković telescope is a 1.4m telescope constructed by the Austrian company ASA through the BELISSIMA project¹ (no. 256772) funded by the European Commission with the support of the Ministry of Education, Science and Technological Development of the Republic of Serbia. It is installed at the

Astronomical Station Vidojevica (ASV)², which is established by the Astronomical Observatory of Belgrade (AOB)³, at the elevation of 1150 m. The telescope is an Alt-Az mount with two Nasmyth ports, which are equipped with de-rotators for correcting the field of view for rotation due to tracking. One of the Nasmyth ports is dedicated to photometry, astrometry and imaging and is equipped with IkonL CCD camera⁴ with standard Johnson's-Cousin BVRI broadband filters, luminance L filter, and H α , H α continuum, and SII narrow-band filters.

© 2024 The Author(s). Published by Astronomical Observatory of Belgrade and Faculty of Mathematics, University of Belgrade. This open access article is distributed under CC BY-NC-ND 4.0 International licence.

¹BELISSIMA is abbreviation for BELgrade Initiative for Space and Science Instrumentation and Modelling in Astrophysics.

²<https://vidojevica.aob.rs>

³<https://www.aob.rs/en/>

⁴<https://andor.oxinst.com/products/ikon-xl-and-ikon-large-ccd-series/ikon-l-936>

Since the beginning of observation activities at ASV in 2010, the AOB has started numerous international follow-up observation projects such as DWARF, WEBT, Gaia (e.g. Vince *et al.* 2018, Vince 2021). This effort was intensified with the start of operation of the 1.4m Milanković telescope in 2016. In parallel with that, the need for a pipeline that will automatically process observations and perform photometric calibration grew. A specific pipeline has been made suitable for dithering techniques aimed at low-surface brightness feature detection (Müller *et al.* 2019). In 2020, two projects from Serbia were evaluated and accepted by the Vera C. Rubin Observatory⁵ project Legacy Survey of Space and Time (LSST)⁶ International In-kind Contribution, out of which one program was proposed by the Serbian AGN Team (program code: SER-SAG)⁷. SER-SAG proposed two contributions: (1) SER-SAG-S1: directable software development "Science Pipeline Development for analysis of variability of celestial sources in the LSST AGN and TVS Science Collaboration" (the contribution lead Andjelka Kovačević) and (2) SER-SAG-S2: an active follow-up program "Optical follow-up of bright LSST transients with Milanković 1.4 Telescope" (the contribution lead Maša Lakićević). One of the scientific results of the Rubin Observatory is the real-time difference image analysis processing as a world-public streams of alerts containing data about transient, variable, and moving sources (Jurić *et al.* et al. 2023). The Rubin Observatory is expected to produce around 10 million of these alerts per night, which will be distributed through seven official Full-Stream Alert Brokers, that primarily built their alert distribution system based on the results of the Zwicky Transient Facility (ZTF, Patterson *et al.* (2019)). ZTF is a wide-field sky survey acting as a sort of a smaller-scale prototype of Rubin Observatory's LSST. In order to characterize and supplement the forthcoming Rubin observatory's discoveries and also the science goals in general, follow-up observations will be performed by many telescopes around the world. The follow-up observations will be coordinated through the Astronomical Event Observatory Network (AEON) that is specially made to meet the demands of science. The SER-SAG-S2 team aims to integrate the Milanković telescope into AEON and to dominantly perform follow-up observations of requested transient sources within LSST, but other observations are also possible through the AEON's regular call for proposals. To fulfill the standards proposed by AEON, which was the motivation for us to write the pipeline, but also to have a functional pipeline for anyone who is using the Milanković telescope, we will describe in this paper the pipeline that is capable of obtaining the best results from ob-

servations performed and in timely manner. At latest, raw data should be accessible within a day upon the observing night, while calibrated data and photometry measurements should be available within the next 48 hours (72 hours in total). The SER-SAG-S2 in-kind program set the list of general basic requirements of the pipeline to: (1) perform a semi-automatic basic reduction (bias, dark, flat-field by default) and, in addition, the sky-background and cosmetic defects corrections, (2) provide astrometric solution, and (3) provide aperture photometry of point-sources. If an additional ASCII file with celestial coordinates of comparison stars is provided by the user, than the differential aperture photometry is the end result.

The output of the pipeline, apart from the fully reduced CCD images, includes ASCII tables (one for each image) of all/preselected light sources listing their celestial position and flux/magnitude measurements. In addition, a single ASCII table with differential photometry is created for each target observed during the night. The pipeline is optimized to work in parallel mode using multiple processors detected automatically to reduce the processing time.

In the following sections, we will describe the pipeline structure (Section 2) and in particular parts of the pipeline that do the reduction (Section 3), astrometric calibration (Section 4), and aperture photometry (Section 5). In Section 6, we test the pipeline by comparing the output parameters with the same measurements made in the IRAF **I**mage **R**eduction and **A**nalysis **F**acility⁸ package. The last chapter is devoted to the summary.

2. PIPELINE STRUCTURE

A pipeline is a standalone python code (>v3.9) incorporating several python packages: astropy (v5.3.4), numpy (v1.26.0), ccdproc (>v2.4.1), scipy (>v1.11.3), pandas (>v2.1.3), photutils (>v1.10.0), and twirl (v0.4.2). We recommend using the Conda or Anaconda environment to install the python programming language and specified packages to successfully run the pipeline.

The pipeline is designed to process observations made with the 1.4m Milanković telescope, and it depends heavily on the keywords that are written in the FITS header. For example, the IMAGETYP keyword is used to collect and distinguish between the calibration and science FITS files, OBJCTRA and OBJCTDEC are used to transform pixels into the WCS system, and so on. At the Astronomical Station Vidojevica (ASV) the MaxImDL⁹ software is used to control the CCD camera and to create FITS headers. We emphasize this to users who want to use the pipeline to process observations from other telescopes, which

⁵<https://rubinobservatory.org>

⁶<https://www.lsst.org>

⁷<https://www.lsst.org/scientists/in-kind-program/programs>

⁸<https://iraf-community.github.io/>

⁹<https://diffractionlimited.com/product/maxim-dl/>

might be using a different software, writing different FITS header keywords. In that case, the source code will have to be slightly modified. For this reason, in the following text, we will always describe the keywords that are used for the specific operations.

The pipeline is run via command-line commands that are entered at the Terminal/Konsole. The `argparse` python package¹⁰ enables interface with the pipeline. In particular, it allows one to specify various positional and optional arguments that determine which of the operations should be executed and in which manner.

In essence, the pipeline performs three operations:

- **reduction (correction for bias, dark, flat, etc., Section 3)**
- **astrometric calibration (Section 4)**
- **aperture photometry (Section 5)**

Except for reduction, astrometric calibration and photometric measurements are initiated by specifying the appropriate optional arguments on the command line. However, the astrometric calibration only works with reduced CCD images and photometry only works with astrometrically calibrated images, so the operations can be run separately providing this condition is satisfied.

The reduction part of the pipeline will always perform the basic calibration, i.e. correction for bias, dark, and flat-field, and all other types of correction are optional. There are various other optional arguments that control the execution modes of this part of the pipeline and they will be introduced in the Section 3. We note, that for a successful reduction, the calibration frames (bias, dark, flat) must have the same resolution as the scientific frames; for example, if we observe in the 2x2 binned mode then the calibration frames must be binned as well.

The astrometric calibration is triggered by specifying the appropriate optional argument on the command line. A prerequisite for this part of the pipeline to work is that the science frames have been reduced. The astrometric calibration can be done in two ways - using the python package `twirl` (Garcia et al. 2022), and by the independent software `astrometry.net`¹¹ that must be installed on the computer where the pipeline is run. Both packages have their advantages and disadvantages and they are described in the Section 4. The desired package can be selected using an optional argument on the command line.

Photometry is also performed provided the appropriate optional argument is given on the command line. This part of the program is only executed if the science FITS images are astrometrically calibrated. The pipeline is designed to do the aperture photometry of point sources. The optional keys, output FITS files, and tables will be described in Section 5.

Fig. 1 is a flow-chart showing the execution of the pipeline. On the left side of the diagram, the operations for reduction and astrometry are shown, while those on the right side, are for photometry. The steps in the process (symbols) are documented in detail and, in that sense, the diagram is a good summary of what the pipeline performs. We note here that all execution steps are recorded in the file `logfile` along with the exact time when the execution started and ended. Upon each new run of the pipeline, the information is appended, but the blocks of information can be distinguished by the date and time that is being recorded when different operations started.

A good starting point to familiarize the user with positional and optional arguments is to run the pipeline in the following way:

```
$ python reduce.py -h
```

where `-h` is an optional argument that prints the help message on the screen including the list of all available arguments. In the list, one can note that there is only one positional argument that needs to be specified when running the pipeline, i.e. the absolute/relative path to the folder containing the raw calibration and science images (see for example command 3.1). Only if the raw FITS images are stored in the current working directory along with the pipeline, this positional argument can be omitted. Output files of the pipeline are saved in the current working directory and subfolders `calibration`, `astrometry`, and `photometry`, where FITS images and tables associated with corresponding operations are stored.

The current version of the pipeline can be found at Github¹². Apart from the source code, there is a README file containing: a list of various arguments with examples, example headers of output tables, etc. There is also a Jupyter notebook `example.ipynb` that provides guidelines to use the pipeline with a specific observational date and additional files which requires cloning a repository to the local computer.

3. DATA REDUCTION

The working directory (`/home/data/`) should contain the calibration files (bias, dark and flat files), and science frames. The pipeline can handle multiple targets observed during a single night. In the reduction part of the pipeline, the raw CCD images (science frames) are corrected for bias, dark, flat-field (basic corrections), cosmetic defects (hot/dead pixels, cosmic rays)¹², and sky background (fringing, sky gradient). When starting the pipeline, the basic corrections are always performed. Correction for cosmetic defects and sky background is optional and included in the process by specifying the appropriate keys. In the following subsections, we will describe each of the mentioned corrections in more detail.

¹⁰<https://docs.python.org/3/library/argparse.html>

¹¹<https://astrometry.net/>

¹²<https://github.com/anavudragovic/pipeline/tree/main>

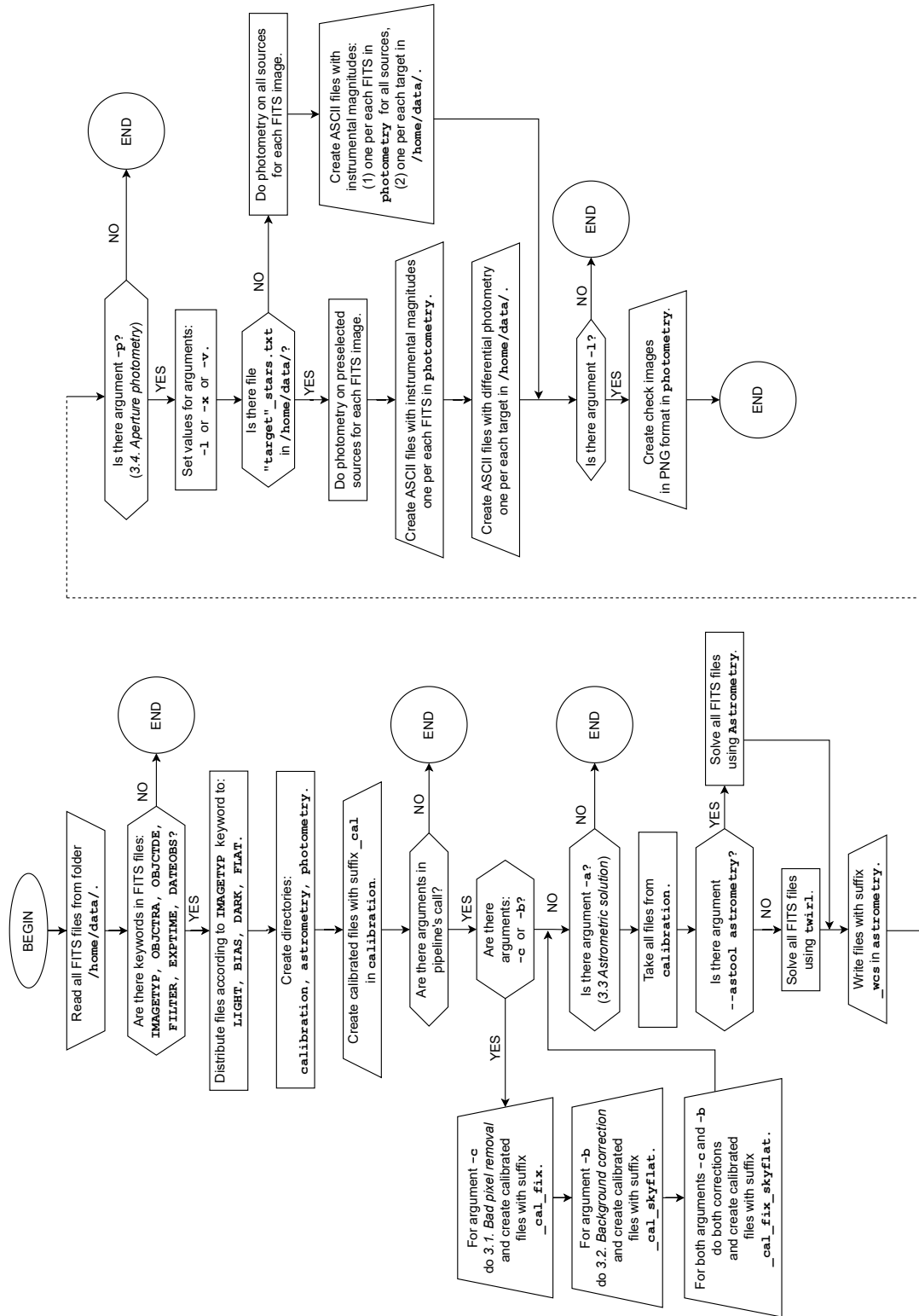


Fig. 1: Flow chart of the pipeline. The main branch (operations executed by default) always descends vertically downwards, while the optional branches (procedures) branch out sideways.

3.1. Basic correction

The bias, dark, and flat corrections are done in the usual way: (1) the master bias, dark, and flat frames are created using appropriate calibration images, and (2) the scientific images are corrected using master frames. The master bias frame is created by median combining all bias frames. The pipeline uses `IMAGETYP=BIAS|ZERO` (i.e. `BIAS` or `ZERO`) to recognize bias calibration images. The output master bias frame is called `mBias.fits`. The master dark frames are also obtained by median combination of master bias subtracted dark frames, with the same exposure time. The pipeline uses `IMAGETYP=DARK` and `EXPTIME` from the header to create master dark frames. Outputs are for example `mDark_5sec.fits` or `mDark_200sec.fits` where the exposure times of dark frames can be discerned from the FITS names. If there is a master dark frame with the same exposure as the flat-field, and science frames being calibrated, the pipeline will use them for calibration. Otherwise, it will use the master dark frame with the highest exposure time and apply appropriate scaling for dark calibration, provided the exposure time is greater than the exposure time of the frames being calibrated. If none of these conditions are met, only the master bias frame correction will be performed. The master flat frames in different filters are bias and dark subtracted, median combined, and normalized to the median value of the master frame for the particular filter. The pipeline uses `IMAGETYP=FLAT` and `FILTER` from the header to recognize flat-field images and the filter in which the flats were created. Outputs are for example `flat_V.fits` or `flat_I.fits` where the names identify the filter used to create the flat-field images. All the master calibration frames are saved in the subfolder `calibration`. As we mentioned above, the bias, dark, and flat corrections are always performed (by default) when the pipeline is run:

```
$ python reduce.py /home/data/
```

Calibration of scientific images is done in the usual way, i.e. by subtracting master bias frame, subtracting the corresponding (scaled) master dark frame and dividing by the corresponding normalized master flat frame. The pipeline uses `IMAGETYP=OBJECT|LIGHT` and `FILTER` from the header to identify science images and the filter in which they were taken. Calibrated science frames are also stored in the `calibration` subfolder, and the suffix `_cal` is added to each calibrated FITS file. Among other things, the pipeline uses this suffix so that it does not repeat the calibration if the pipeline is restarted. This part of the program is accelerated by using `multiprocessing` python package, which enables calibration to run in parallel mode on multiple processors.

3.2. Correction for bad pixels

Bad pixel correction includes correction for hot/dead pixels and cosmic rays. Cosmic rays are detected using the `cosmicray_lacosmic` module of the `ccdproc` python package. `Cosmicray_lacosmic` is a complex python program with many input parameters and the effectiveness of cosmic ray detection depends on their settings. Based on previous processing of the CCD images from the 1.4m Milanković telescope, we have listed the recommended set of parameters, which may not be optimal for all type of observations. We have increased the `sigclip` parameter to 7 (4 by default) to lower the cosmic rays detection threshold since too many pixels had been flagged with the default, lower sigma clipping threshold. Further, we have increased the parameter `objlim` from 5 to 20 to exclude saturated stars from the detected cosmic rays. Finally, we have set the saturation level (`satlevel` parameter) to 55 000 ADU to correspond to the saturation level in images. To check the effectiveness of cosmic ray detection, one can use the FITS files with the extension `CRmask`. These check files are created when the pipeline is started with the optional argument `-s` (save), which allows them to be saved. In the case that detection of cosmic rays is not satisfactory, the input parameters for the `cosmicray_lacosmic` module will have to be changed in the source code itself.

To detect the hot/dead pixels, we use the master dark frame with the highest exposure time. First, we make a difference between the master dark image and its smoothed version. Then, the hot/dead pixels are identified as pixels whose standard deviation is above a given threshold. Similar to the case of `cosmicray_lacosmic`, we set the threshold to 3 based on our experience working with the pipeline, but this threshold can be changed in the source code itself (search for the `hotPixThresh` parameter in the `fixpix` function).

The final mask, which contains the hot/dead pixels and cosmic rays, is a combination of these two masks, and pixel correction is performed by assigning interpolated values of 'healthy' pixels in the vicinity of 'bad' pixels in the mask.

From our experience, correction of science images for bad pixels is time consuming, so we put this module in the pipeline to be optional. Correction of bad pixels will be done with the `-c` (correct) option when starting the pipeline:

```
$ python reduce.py /home/data/ -c
```

The output corrected FITS files are stored in the directory `calibration` and the suffix `_fix` is added to each corrected FITS file.

3.3. Background correction

For the correction of large-scale variations in the background (for example fringing in the I filter or sky gradient in general) we use all calibrated science

images in the appropriate filters. First, each science image is divided by the median value of the image on which the stars were previously masked. Then, we median combine the masked images in all filters to get the corresponding super-sky flat. Next, super-sky flats are multiplied with the corresponding median value of each science frame and subtracted from it. Finally, the median value of each frame is added as a constant to preserve the background signal in individual images.

To make the background correction, we start the pipeline with the option `-b` (background) as follows:

```
$ python reduce.py /home/data/ -b
```

Calibrated images are recorded in the `calibration` folder, and the suffix `_skyflat` is added to the corrected FITS images. Running the pipeline with the option `-s` allows saving various FITS images for checking the efficiency of creating a super-sky flat. The output files named `forSkyFlat*` are FITS files used to calculate the super-sky flat images. The super-sky flat images are saved as `mSkyFlat[FILTER].fit`.

Similar to the correction for bad pixels, we have also made the correction for the background optional because the effectiveness of the correction depends on many factors. In the first place, it is worth noting that the dithering technique is mandatory if a single object is imaged and its efficiency increases with the larger number of images taken in each filter. The probability of stars overlapping with themselves should be reduced as much as possible with appropriate offset, and the accompanying noise is reduced by combining more images. The dithering offset should be larger than the number of pixels the largest object in the field occupies to ensure this object will not overlap with itself between individual exposures.

Fig. 2 shows the result of correction of a raw science image in I filter with all possible optional arguments offered by the pipeline. The top left image is a raw science image. The upper middle image is the master flat in I filter, where one can see the presence of dust on the optical elements near the focus in the form of donuts, as well as the asymmetric brightness of the field of view due to the poor centering of the CCD camera relative to the optical axis. The upper right image has been corrected for master bias, dark, and flat images. The lower left image shows the calibrated image with masked stars in the field of view. The bottom middle image is the result of making a super-sky flat image. Finally, the bottom right image is a fully calibrated image

For new users, we suggest running the pipeline with the `-c -b -s` options first. In this way, all possible corrections offered by the pipeline will be performed and intermediate FITS files will be saved to check if the calibration was successful. With further testing, the user will become more familiar with the pipeline efficient usage as well as with its limitations.

4. ASTROMETRIC CALIBRATION

After proper calibration, the astrometry is done using either the external code `Astrometry` or using the `twirl` python package (Garcia et al. 2022). Both methods are optimized to run as multiprocessor tasks. Astrometric solution is also an optional feature and needs to be invoked by the following command:

```
$ python reduce.py /home/data/ -a
```

Please note that the above command will perform astrometry on the `*cal.fit` files, that is, science FITS images corrected for bias, dark, and flat. If we want to do astrometry on FITS files that have been corrected for bad pixels (with option `-c` in the data reduction part), we should apply the following command:

```
python reduce.py /home/data/ -c -a
```

Likewise, we use the optional arguments `-c -b -a` if we want to perform astrometry on background-corrected CCD images that have been corrected for cosmetic defects.

There are several prerequisites for `Astrometry` to work. The first one is that it must be installed on the local computer where the pipeline is run (a detailed description of the `Astrometry`, as well as the installation process, is given on their official website¹¹). The second one is that the appropriate index files are downloaded. Index files are used to identify our field of view based on different catalogs. Basically, all index files can be downloaded if there is enough space on the computer where we run the pipeline, but for a successful astrometric calibration, it is enough to download the index files that correspond to the pixel scale of the Milanković telescope.

On the other hand, the `twirl` package (Garcia et al. 2022, Lang et al. 2010) is based on the same algorithm as `Astrometry` and needs an internet connection to work, because a World Coordinate System is computed by cross-matching objects in each frame with the GAIA DR2 catalog of reference stars fed by the celestial coordinates of the image center and its field-of-view. Both functions use the celestial equatorial coordinates for transformation into the WCS system, and require the following keywords from the header: `OBJCTRA` and `OBJCTDEC`. These two parameters with an estimated field-of-view are sufficient for the fast and efficient operation. The `twirl` package is used by default and `Astrometry` can be used by specifying the appropriate optional key:

```
$ python reduce.py -a --astool astrometry
```

In the case of basic data reduction, the output files stored in the directory `astrometry` will have the suffix `cal_wcs`. If, however, both additional corrections were applied then the output files will have the suffix `cal_fix_skyflat_wcs`.

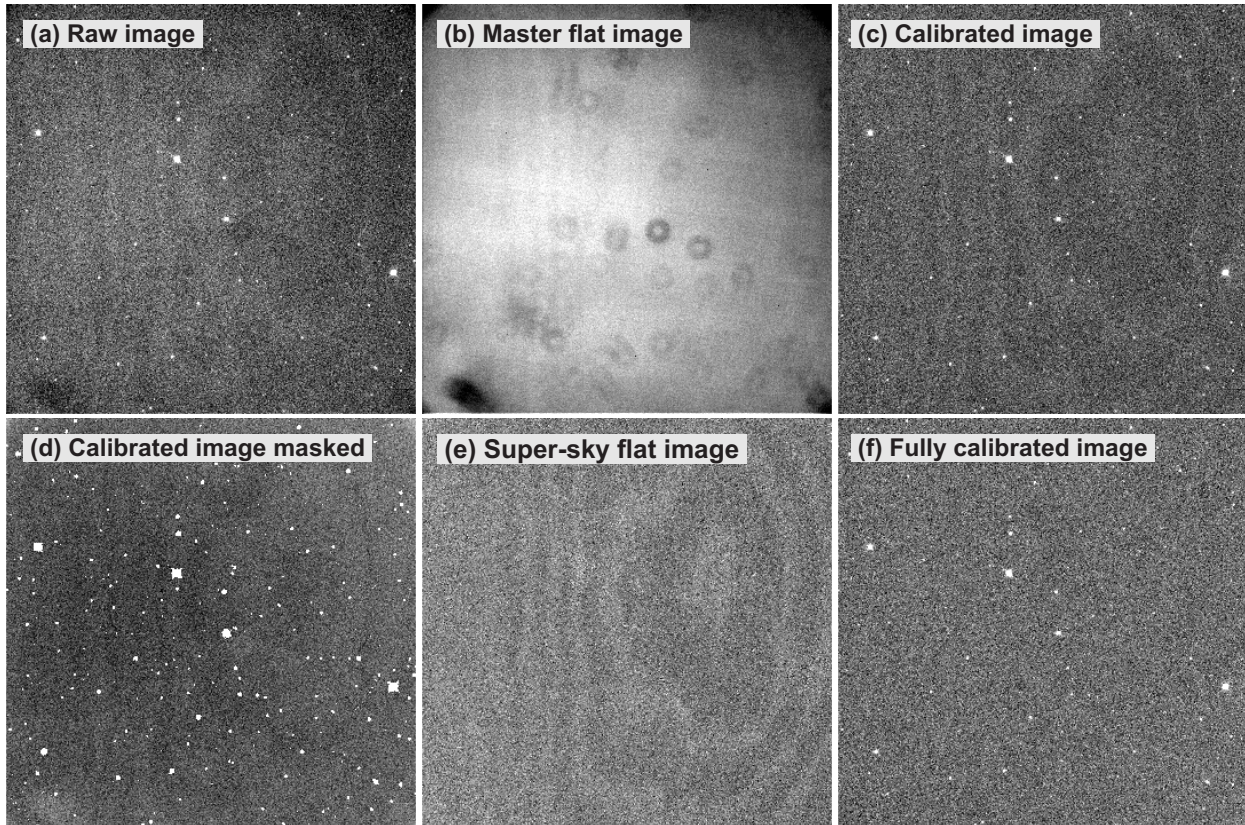


Fig. 2: One of the images taken with the I filter shown through different steps of calibration process from the raw image (a) corrected to the flat-field (b) shown in the panel (c). In this image objects are masked (d) for the purpose of making the super-sky flat (e) and finally, the calibrated image shown in the panel (c) is corrected to the super-sky flat (f).

5. APERTURE PHOTOMETRY

Aperture photometry is based on the `photutils` and `astropy` modules. Astrometric solution is a prerequisite step if the user wants to proceed with aperture photometry, since the target and/or comparison stars have to be identified in each image frame by their celestial coordinates. In this case, in addition to photometry (`-p`), the keyword for astrometry (`-a`) must be included:

```
$ python reduce.py /home/data/ -a -p -l
```

where the optional keyword `-l` is added to provide check images in the png format saved in the `photometry` directory with circles drawn around selected objects.

An important note: whatever additional keys were added in the process of calibration they should be kept until the end. For example, if the user was satisfied with the super-sky flat correction, a `-b` key should be added in a call:

```
$ python reduce.py /home/data/ -b -a -p -l
```

so that magnitudes are measured in calibrated frames corrected for sky gradient.

Photometry is done in the following way: all light frames are being inspected to differentiate between multiple observing targets if present. For each target observed during the single night aperture photometry is performed separately. There are two options: (1) either a list of preselected sources is provided through an external ascii file (a usual procedure for differential photometry), or (2) sources brighter than some predefined threshold are being detected. The external file consists of one target and its corresponding comparison stars, not of all stars in the image. There can be multiple external files, each for a single target. If provided, the external file must be named after the target followed by the `_stars.txt`, since the pipeline expects such a file, with celestial coordinates of the target followed by comparison stars. In addition, we recommend that the user provides the catalog with celestial coordinates of all targets that were observed during the particular night, so that the targets can be properly identified based on their celestial coordinates. This is a file named `catalog.cat` by default at the Astronomical Station Vidojevica, and is usually provided even before the observing run. If such a catalog is not provided, the pipeline uses the nominal

celestial coordinates read from the image header to identify the observed targets.

The preselected sources are re-centered by modelling their light distribution with 2D Gaussian function if the external file of comparison stars is provided. Otherwise, the sources are detected above some predefined threshold (three standard deviation above the image median). By default, the full width at half maximum (FWHM, hereafter) is fixed to 3 pixels. The fixed value can be changed supplying the keyword `--fwhm-fixed` (or `-x` for short), followed by the user defined value. For example

```
$ python reduce.py /home/data/ -a -p -x 4
```

assigns value of 4 pixels to the FWHM keeping it fixed.

If the user wants to enable variable FWHM the keyword `--if-fwhm-variable` (`-v` for short) should be set to True by simply adding `-v` option:

```
$ python reduce.py /home/data/ -v
```

In this case, the pipeline will take an average measured FWHM of all the sources as an optimal value throughout further measurements: apertures are defined as 3 times the optimal FWHM, and sky annulus is 10 pixels wide and placed 10 pixels away from the aperture.

Output tables are produced for each science frame and they include: pixel positions (`xpix`, `ypix`), source ID (`ID`), equatorial coordinates (`RAJ2000`, `DECJ2000`), fluxes (`flux`, `flux_err`) and magnitudes (`mag`, `mag_err`) with their corresponding errors, and additional information read from the image header: `AIRMASS`, `FWHM`, `FILTER`, `MJD-HELIO`, `DATE-OBS`, `flag`. The last column `flag` is created by the pipeline to indicate if the particular source is saturated. If an external input table listing preselected sources is given (the target underscore `_stars.txt` file), than an additional table is created with differential photometry. The content of this table depends on the number of comparison stars, and includes the following: `FILENAME`, `MJD-HELIO`, `FILTER`, `DATE-OBS`, `TmCi`, `TmCi_err` (where index `i` takes values from 1 to the number of comparison stars; label `T` stands for target, while `C` stands for the comparison star, so that these columns represent the difference between the target and comparison star magnitudes); `CimCj`, `CimCj_err` (where `i` and `j` are different indices referring to the magnitude difference of comparison stars). Even if this input table is missing, the single output table with aperture photometry of the target is created, consisting of a target photometry solely with each line corresponding to the single frame, and consisting of the following columns: `FILENAME`, `MJD-HELIO`, `FILTER`, `DATE-OBS`, `RAJ2000`, `DECJ2000`, `xpix`, `ypix`, `flag`, `flag_p`, `flux_peak`, `flux`, `mag`, `mag_err`. Apart from the header keywords and flux and magnitude measurements, additional flag (`flag_p`) is present indicating if the object in the image is found within 5 arcsec from the celestial coordinates provided either through the

input file (`katalog.cat`) or read from the image header. Also, there is a measure of maximum flux for the target object (`flux_peak`).

Output tables are named after the target followed by the underscore "photometry.txt" and are placed in the directory with raw data (the working directory). Example of all these output tables can be found on the GitHub.

6. OBSERVATIONS AND TESTS

Observations were carried out during three nights: 12/09/2023, 09/10/2023 and 13/10/2023, targeting Sayfert 1 galaxy Mrk335 ($RAJ2000 = 00^h : 06^m : 19^s.537$, $DECJ2000 = 20^\circ : 12' : 10''62$) that is known to be variable over a longer time period (Hyung et al. 2000). The PI of the observations was Maša Lakićević. Since this is a variable object, the comparison stars were imaged in the same field-of-view. The list of comparison stars was taken over from Table 4 of Doroshenko et al. (2005), where the finding chart is given along in their Fig. 3a. Each night the target Mrk 335 was observed using the following filters: B, V, R, I, $H\alpha$, SII, and $H\alpha$ -continuum (c, hereafter). On the first night (12/09/2023), 140 frames were taken (20 per each band), followed by 35 frames (5 in each band) on 09/10/2023, and finally 112 frames (16 in each band) were taken on the last night (13/10/2023). Dithering of 40 pixels corresponding to ~ 16 arcsec was used. Seeing was 2 arcsec on average. Raw data were calibrated with the pipeline and the flatness of the images was inspected, measured, and compared with the corresponding IRAF procedures. Aperture photometry included into the pipeline was also done separately with the IRAF `phot` procedure for comparison.

6.1. Sky flatness test

Sky flatness is being inspected using the noise variations along the image comparing local to global noise. We have placed 50 boxes $10 \text{ arcsec} \times 10 \text{ arcsec}$ size randomly in each image measuring the noise inside each box as a standard error of the mean, i.e. standard deviation (σ) of the mean sky value divided by the square-root of the number of pixels (N_{pix}) inside the box. In Fig. 3, the standard deviation (σ) of the background signal in the single box is illustrated. For the entire image, the local noise is simply the average value of the local noise of all the selected boxes, i.e. $\langle \sigma \rangle / \sqrt{N_{pix}}$.

The objective is to check if the local noise defined above is smaller than or equal to the global noise that is measured by the standard deviation of these individual local noise estimates (σ_σ). The sky is considered to be flat if this condition is satisfied since there are no significant gradients left over after the data reduction is done. To conclude, if the sky is flat, the ratio of these two measures should not exceed unity, i.e. $\sigma_\sigma / (\langle \sigma \rangle / \sqrt{N_{pix}}) \leq 1$. The vast

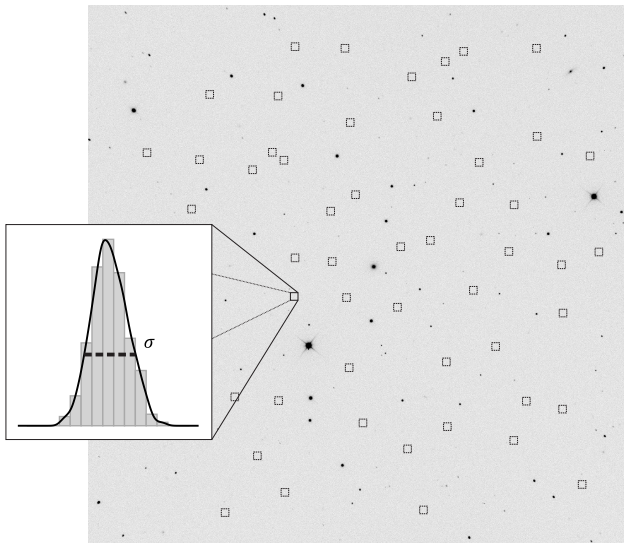


Fig. 3: Small boxes (50 in total) were placed in empty patches of calibrated images for background statistics. Inside each box, the 'local' standard deviation of the sky background (σ) in ADU units is measured as illustrated in a single zoomed-in box with tick dashed line.

majority of points representing the aforementioned ratio lie below unity in Fig. 4. Both IRAF and the pipeline provide good results and agree well. Several measurements above unity that fail to satisfy the imposed criterion are considered to arise from the images that are not flat either way - calibrated with the pipeline, or IRAF. These results are based on the basic data reduction without correcting for the sky gradient.

6.2. Aperture photometry test

Aperture photometry is incorporated into the pipeline via `photutils`, an Astropy package for photometry, using circular apertures with the radii of 3 times the FWHM of the sources. Sigma clipping is done to cancel out any emission left in the sky annulus that is placed 10 pixels away from the aperture and is 10 pixels wide. The magnitude error is estimated using the standard formula: $1.0857 * F_{err} / F$, where flux (F) is measured inside the aperture, while the flux error (F_{err}) is estimated as the square-root of the sum of the flux itself (F), the noise in the aperture (i.e. standard deviation σ per pixel), and in the annulus (S) scaled to the size of the aperture (A):

$$F_{err} = \sqrt{(F + A * \sigma^2 + A^2 / S * \sigma^2)}$$

In addition, the peak flux value of objects is stored and used to warn the user if the particular source is saturated by adding the flag column (`flagP`) to the output table: if the object is saturated, the flag is set to unity, otherwise it is set to zero.

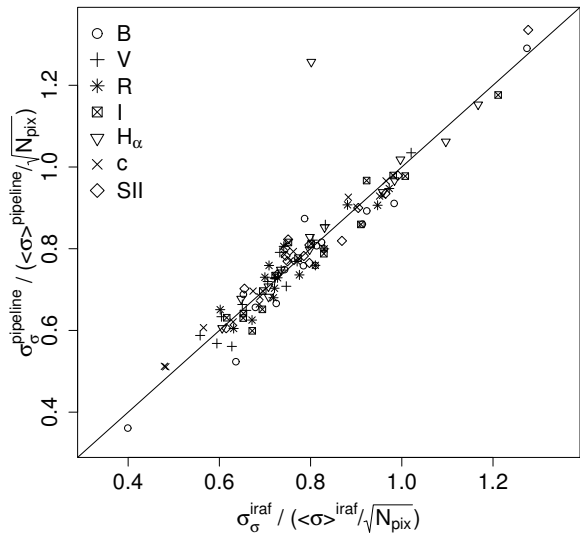


Fig. 4: The ratio between the so-called global and local noise variation for each image is plotted using different point shapes corresponding to the particular filter listed in the legend for the observing night 13/10/2023: the sky flatness measured in the images calibrated with the IRAF standard reduction procedures (x-axis) is compared with the output from the pipeline (y-axis).

Aperture photometry is done for objects selected either from an external file with comparison stars providing their celestial coordinates (equatorial J2000), or all the objects detected in images. The external file should consist of a list of target and comparison stars (in that order) with their equatorial coordinates (J2000) followed by their names (optionally) - each line of the file should correspond to a single object. There is a requirement for the file name - it has to be specific: the target name followed by `_stars.txt`. If this file is provided, differential photometry will be provided as the end result. If such an input file is not provided, then all the sources that have more than 5 connected pixels above the 3σ threshold are detected in the image. Since FWHM of the sources can be of importance, there are two choices: (1) to keep it fixed or (2) to measure it (if `if_fwhm_variable` global variable can be set to either True or False). In the later case, all the sources and modelled with a 2D Gaussian function whose mean value is taken to be an optimal FWHM. Either way, the output table is produced listing celestial and pixel coordinates, measured fluxes and magnitudes with useful keywords (modified Julian date, airmass, filter etc.) - one for each science frame. In addition, a master photometry ASCII file is created in the working directory ending with "photometry.txt". If the list of comparison stars is provided than this master file will list the magnitude difference between the target and all comparison stars, followed by the magnitude difference of all comparison stars themselves (differ-

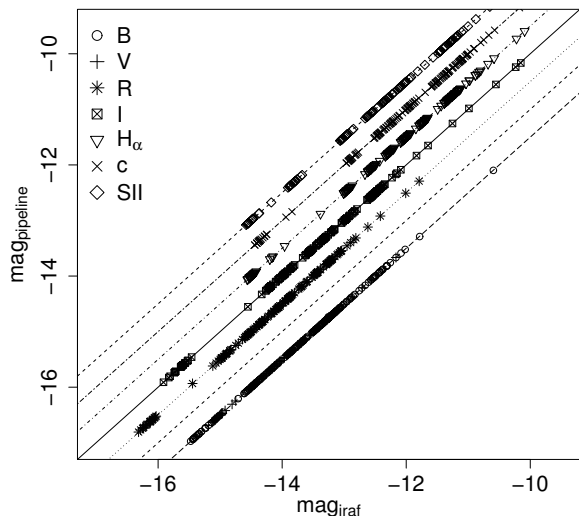


Fig. 5: Instrumental magnitudes of comparison stars measured using the pipeline $\text{mag}_{\text{pipeline}}$ and IRAF `phot` procedure mag_{iraf} . Different filters are given by various point shapes as indicated in the legend. Individual points are offset from one-to-one line that goes through I filter for clarity.

ential photometry). Otherwise, this file will simply contain the instrumental magnitude of the observed target in a single line for a single image.

To test the performance of aperture photometry done with the Astropy package `photutil`, we have measured fluxes and magnitudes with the IRAF procedure `phot` as a standard tool for aperture photometry. The comparison is given in general for all the comparison stars and all observational dates differentiating only by the filter used in Fig. 5. Errors are omitted since they are smaller than the point markers. Magnitudes in all other bands except for the I-band (thick line) are offset from one-to-one line for clarity. Since the standard deviation of the magnitude difference is smaller than or equal to the mean error in magnitudes the agreement is very good (Table 1).

7. SUMMARY

In this paper, we described the pipeline for reduction of CCD images taken with the 1.4m Milanković telescope, astrometric calibration of target images, and photometric measurements of point-like sources in the field of view based on their celestial coordinates. By default, the reduction performs bias, dark and flat corrections, but optionally the corrections for cosmetic defects and/or sky background can be applied. The astrometric part of the pipeline works under the condition that the previously mentioned reduction is done. This calibration can be performed with the python based code `twirl` or with the exter-

Table 1: For each filter listed in the first column, the mean difference between magnitudes measured with the pipeline and IRAF `phot` procedures ($\Delta\text{mag} = \text{mag}_{\text{pipeline}} - \text{mag}_{\text{iraf}}$) and the standard deviation of this difference ($\sigma(\Delta\text{mag})$) are given, followed by the mean error of both magnitudes measured by summing up individual errors in quadrature ($\Delta\text{mag}_{\text{err}}$).

Filter	Δmag	$\sigma(\Delta\text{mag})$	$\Delta\text{mag}_{\text{err}}$
B	0.002	0.003	0.004
V	0.001	0.016	0.016
R	1.e-6	0.003	0.003
I	0.001	0.001	0.005
H α	0.001	0.001	0.008
c	0.001	0.001	0.008
SII	0.002	0.002	0.007

nal code `astrometry`. The photometric part of the pipeline is performed under the condition that the reduction and astrometric calibration have been performed. The pipeline performs photometry of only point objects in the field of view.

The pipeline is tested and compared to the IRAF procedures both in terms of data reduction and aperture photometry and has proven to be very successful. The pipeline uses multiple processors and is highly efficient and fast. The whole reduction is done automatically for multiple observed objects. Optionally, the correction of fringing and large scale sky gradients can be invoked if a sufficient number of frames is available and if dithering was properly used during observations. The output ASCII files listing aperture photometry of all detected objects are created, and, in addition, separate photometry files consisting of photometric measurements of targets solely are also made. If a list of comparison stars is provided, these separate photometry files contain differential photometry.

Further improvements of the pipeline will include photometric calibration using existing all-sky surveys since instrumental magnitudes cannot be directly used without calibration unless the differential photometry is applied.

Acknowledgements – In-kind contribution of the SER-SAG team to the Vera C. Rubin Observatory’s Legacy Survey of Space and Time (LSST) is twofold: SER-SAG1 (software contribution) and SER-SAG2 (telescope contribution). This work made use of IRAF which listed in the Astronomical Source Code Library as ascl:9911.002. The DOI is 10.5281/zenodo.5816743 (Tody 1986, 1993). This work was supported by the Astronomical Station Vidojevica and by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia through contract no. 451-03-66/2024-

03/200002 made with Astronomical Observatory of Belgrade. This work made use of Anaconda Distribution of Python; Astropy, a community-developed core Python package and various tools and resources for astronomy (Astropy Collaboration et al. 2013, 2018, 2022), particularly Photutils (Bradley et al. 2023), and in addition Python package SciPy (Virtanen et al. 2020).

REFERENCES

- Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, *A&A*, **558**, A33
- Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, *AJ*, **156**, 123
- Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, *ApJ*, **935**, 167
- Bradley, L., Sipőcz, B., Robitaille, T., et al. 2023, *astropy/photutils*: 1.8.0
- Doroshenko, V. T., Sergeev, S. G., Merkulova, N. I., et al. 2005, *Astrophysics*, **48**, 156
- Garcia, L. J., Timmermans, M., Pozuelos, F. J., et al. 2022, *MNRAS*, **509**, 4817
- Hyung, S., Kim, H., Lee, W. B., et al. 2000, *Journal of Korean Astronomical Society*, **33**, 81
- Jurić, M., Axelrod, T., and Becker, A. C. et al. 2023, in <https://lse-163.lsst.io>, pp. 69
- Lang, D., Hogg, D. W., Mierle, K., Blanton, M., and Roweis, S. 2010, *AJ*, **139**, 1782
- Müller, O., Vudragović, A., and Bílek, M. 2019, *A&A*, **632**, L13
- Patterson, M. T., Bellm, E. C., Rusholme, B., et al. 2019, *PASP*, **131**, 018001
- Tody, D. 1986, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 627, *Instrumentation in astronomy VI*, ed. D. L. Crawford, 733
- Tody, D. 1993, in Astronomical Society of the Pacific Conference Series, Vol. 52, *Astronomical Data Analysis Software and Systems II*, ed. R. J. Hanisch, R. J. V. Brissenden, and J. Barnes, 173
- Vince, O. 2021, in XIX Serbian Astronomical Conference, ed. A. Kovačević, J. Kovačević Dojčinović, D. Marčeta, and D. Onić, Vol. 100, 161–168
- Vince, O., Samurovic, S., Pavlovic, R., Cvetkovic, Z., and Djurasevic, G. 2018, *Publications de l’Observatoire Astronomique de Beograd*, **98**, 233
- Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, *Nature Methods*, **17**, 261

ВАЛИДАЦИЈА ПРОГРАМА ЗА ОБРАДУ И АНАЛИЗУ ПОДАТАКА ДОБИЈЕНИХ 1.4_m ТЕЛЕСКОПОМ МИЛАНКОВИЋ

А. Лаловић , О. Винце , М. Стојановић  и С. Марчета Мандић 

Астрономска опсерваторија, Волгина 7, 11060 Београд 38, Србија

E-mail: *ana@aob.rs*

УДК 520.823 : 520.8

Стручни чланак

У овом раду представљамо програм за обраду података написан у језику Python v3.9 који врши обраду CCD слика, астрометријску калибрацију и фотометријска мерења посматрања извршених 1.4-метарским телескопом Миланковић, који се налази на Астрономској станици Видојевица у Србији. Главни подстицај за писање вишенаменског програма налази се у његовој будућој употреби када одговарајућа посматрања пролазних објеката почну као пратећа посматрања која доприносе научним циљевима Вера Рубин опсерваторије, али без ограничења на ове специфичне задатке, с обзиром на то да се већ дуже време на Астрономској станици Видојевица врше и друга пратећа посматрања. Програм се покреће путем командне линије користећи различите позиционе и опционе аргументе зависно

од жељених операција (редукција, астрометрија, фотометрија). Постоје и аргументи задати из командне линије који омогућавају кориснику избор начина вршења одређене операције (мода), као и избора вршења кода на више процесора. Базично, само се врше основне корекције bias, dark и flat-field, док су друге операције и моде опционе. У овом раду описаћемо детаљно сваку од поменутих операција, као и коришћење позиционих/опционих аргумената. Такође ћемо детаљно описати излазне FITS датотеке, помоћу којих се може анализирати рад програма, као и крајње излазне датотеке са фотометријским мерењима. Коначно, упоредићемо резултате програма са одговарајућим процедурама програмског језика IRAF са циљем тестирања и валидације резултата.