Astronomical Data Analysis Software and Systems XXVI
ASP Conference Series, Vol. 521
Marco Molinaro, Keith Shortridge, and Fabio Pasian, eds.

SimCADO - a Python Package for Simulating Detector Output for MICADO at the E-ELT

Kieran Leschinski,[1] Oliver Czoske,[2,1] Rainer Köhler,[2,1] Michael Mach,[1]
Werner Zeilinger,[1] Gijs Verdoes Kleijn,[3] Wolfgang Kausch,[1,2] Norbert Przybilla,[2]
João Alves,[1] and Richard Davies[4]

[1]*Department of Astrophysics, University of Vienna, Austria;*
kieran.leschinski@univie.ac.at

[2]*Institute for Astro- and Particle Physics, University of Innsbruck, Austria*

[3]*Kapteyn Astronomical Institute, University of Groningen, Netherlands*

[4]*Max Planck Institute for Extraterrestrial Physics, Garching, Germany*

Abstract.
SimCADO is the instrument data simulation software for the E-ELT's near-infrared wide-field imaging camera - MICADO. Written in Python, SimCADO allows the user to simulate possible future observations of astronomical objects with the 39m European Extremely Large Telescope (E-ELT). In these proceedings we present a brief introduction into how to use SimCADO.

1. Introduction

Within the next 10 years a new generation of extremely large telescopes will become available to the astronomical community. These telescopes and the instruments attached to them will allow observations of the faintest and most distant phenomena in the Universe. The first-generation near-infrared wide-field imaging camera for the European Extremely Large Telescope (Gilmozzi & Spyromilio 2007) will be MICADO (Davies 2016). As part of its development we have created a Python package – SimCADO – which simulates the output of the MICADO detector array (Leschinski 2016).

With the help of the SCAO and MCAO observing modes, MICADO will have the ability to observe at the diffraction limit in the 0.7–2.5μm wavelength range. It will offer a wide field (4mas/pixel) and a zoom (1.5mas/pixel) imaging mode, as well as a long-slit spectrographic mode. For each of these modes the SimCADO package combines the effects that elements along the optical path have on photons traveling from their point of origin, through the E-ELT+MICADO system and onto the detector chips. The output of a SimCADO simulation run is therefore akin to the raw output expected from the MICADO detector array.

SimCADO is, and will be, used by various teams within the MICADO consortium. Consortium-internal uses of SimCADO include: providing a single platform for the science team to solidify the science drivers for MICADO; acting as a stand-in for the MICADO instrument to aid in creating the specifications for the data flow infrastructure; as a preliminary tool for design trade-off studies; generating updated data tables for the ESO exposure time calculator; etc. SimCADO will be a useful tool for simulating a broad variety of astrophysical objects and processes in order to efficiently prepare observing programs for MICADO at the E-ELT. In these proceedings we provide an overview of the internal working of the SimCADO package and describe an example of a use-case simulation.

2. The inner workings of SimCADO

The SimCADO package contains four main classes, each of which represents an independent part of the simulation process. A detailed description of these classes and the methods contained within can be found in Leschinski (2016).

`Source`	holds all the spatial and spectral information pertaining to the celestial object of interest.
`OpticalTrain`	holds information on the spatial, spectral, and (in the future) temporal effects that each element in the optical train has on the incoming light.
`Detector`	contains information on the individual chips in the focal plane array and how they convert incoming "photons" into pixel counts in the final FITS image. This conversion uses the code developed by Rauscher (2015) to model detector noise.
`UserCommands`	contain a keyword-value-pair style dictionary with all the parameters required to successfully run a simulation.

If the casual user is only interested in simulating future MICADO imagery, they may use the convenience function `simcado.run()` which uses the default parameters based on the current design of both the E-ELT and MICADO. Thus the casual user will only need to be familiar with the first of the four classes, i.e. "Source".

The `Source` object contains spectral (SEDs, weights) and spatial (x,y) information about the object of interest. In order to save memory, and thus computing time, this information is split and stored separately. Two arrays (`<Source>.x`, `<Source>.y`) hold the coordinates of each "pixel" containing photon emission above a certain threshold (default is > 0 ph/s). The array `<Source>.spectra` contains all the unique spectral energy distributions (SEDs) in the field of view (FoV). `<Source>.ref` links each set of coordinates with the corresponding spectrum. As many coordinates may reference the same unique spectrum (e.g. all G2V stars in the FoV), each coordinate also contains a spectrum scaling factor (`<Source>.weight`)

3. Working with 'Source' objects in SimCADO

`Source` objects can be created in many ways, although in all cases the user must provide some sort of spectrum and the spatial coordinates of the emission regions (where the distance units are in arcseconds from the centre of the FoV). For convenience, SimCADO provides stellar spectra from the Pickles (1998) library. The following lines of code create a `Source` object for a single K=20 G2V star in the centre of the FoV:

```
>>> import simcado.source as src

>>> lam, spec = src.SED("G2V", "K", 20.)
>>> my_star = src.Source(lam=lam, spec=spec, x=[0], y=[0], ref=[0])
```

In case the object is an extended source, SimCADO can also create a `Source` object from a 2D `numpy` array (e.g. read in from the data extension of a FITS file). In this case the user must also specify, at the very least, the plate scale (`pix_res`) of the image. SimCADO represents the continuous emission regions as a series of point sources on an over sampled grid. In order to maintain the illusion of continuity, the spacing between these point sources is well under the Nyquist sampling size of the MICADO detector chips. Additionally, the energy units of the image and a zero-flux threshold value are optional but highly recommended.

```
>>> ## br_gamma is the emission spectrum for the nebula at 2.16um
>>> m42 = astropy.io.fits.getdata("orion.fits")
>>> my_nebula = src.source_from_image(m42, lam, br_gamma, pix_res=0.004)
```

Science fields very rarely contain only single objects. Thus SimCADO allows the user to combine `Source` objects using the "+"-operator, in order to build up a more realistic picture of the simulated observations. For example, the G2V star referenced above may be added to the centre of the Orion nebula by simply adding the two `Source` objects. In this way many individual objects may be combined to create `Source` objects which contain all the components of a real observational field.

```
>>> obs_field = my_star + my_nebula
```

4. Running a SimCADO simulation

SimCADO offers several levels of control over how a simulation is run. At the highest level, SimCADO assumes default parameters for everything and only requires a `Source` object to create mock observations. This level of control is suitable for testing the feasibility of science cases. By creating and modifying a `UserCommands` dictionary, the user has access to approximately 100 parameters which dictate how SimCADO applies the effects of the optical train to the incoming light. By modifying one or more parameters the user can simulate different optical path configurations and therefore investigate the effects that these changes will have on the quality of the MICADO observations. At the lowest level, the user can over-ride the inbuilt data for the optical train, e.g. provide different adaptive optics PSF images, change the detector layout, use different mirror reflectivity curves, etc. This level of control allows the user to study the effects of different materials and/or designs for the optical train. It must be stated though, that SimCADO is not designed to replace detailed trade-off studies and that any results with SimCADO should solely be used to "test the water".

A simple simulation using all the default values can be achieved using the command:

```
>>> simcado.simulation.run(source [, filename, cmds])
```

If a filename is specified, the SimCADO output is written to disk in the form of a FITS file, otherwise SimCADO returns to the user (e.g. console, ipython notebook, etc.) an object of type `astropy.io.fits.HDUList` with the readout images from the detector chips.

Alternatively, the user can control each step of the simulation and extract intermediate data (i.e. noiseless images, detector noise frames, etc.). An example of a step-wise simulation could be:

```
>>> my_source = simcado.source.source_1E4_Msun_cluster()
>>> my_commands = simcado.UserCommands()
>>>
>>> my_optics = simcado.OpticalTrain(my_commands)
>>> my_detector = simcado.Detector(my_commands)
>>>
>>> my_source.apply_optical_train(my_optics, my_detector)
>>> my_detector.readout(filename=''my_obs.fits'')
```

For more detailed information, please see the online documentation at:
`http://www.univie.ac.at/simcado/`

5. Conclusion and future development plans

The core structure of SimCADO, as well as a first implementation of the imaging modi for MICADO have been implemented. It is now possible to simulate mock output imagery for the MICADO focal plane array using the current instrument and telescope design parameters. The

current version of the package is available upon request. Over the coming months we plan to implement MICADO's spectroscopic mode, as far as the current design will allow. We also plan to create a version of SimCADO for the currently operating HAWK-I instrument at the VLT, so that we can rigorously test the photometric accuracy of SimCADO.

Acknowledgments. This publication is supported by the Austrian Ministry of Science, Research and Economy (bmwfw) through project IS538003 (Hochschulraumstrukturmittel). GVK acknowledges support by the Netherlands Research School for Astronomy (NOVA) and Target. Target is supported by Samenwerkingsverband Noord Nederland, European fund for regional development, Dutch Ministry of economic affairs, Pieken in de Delta, Provinces of Groningen and Drenthe.

References

Davies, R. e. a. 2016, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 9908 of Proc. SPIE, 99081Z. 1607.01954, URL http://dx.doi.org/10.1117/12.2233047

Gilmozzi, R., & Spyromilio, J. 2007, The Messenger, 127, 11

Leschinski, K. e. a. 2016, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 9911 of Proc. SPIE, 991124. 1609.01480, URL http://dx.doi.org/10.1117/12.2232483

Pickles, A. J. 1998, PASP, 110, 863

Rauscher, B. J. 2015, PASP, 127, 1144. 1509.06264