DRAGONS Tutorial - GSAOI Data Reduction

Release 3.0.2

Bruno Quint

July 2022

Table of Contents

Document ID

PIPE-USER-118_GSAOIImg-DRTutorial

CHAPTER 1

Introduction

This tutorial covers the basics of reducing GSAOI data using DRAGONS.

The next two sections explain what are the required software and the data set that we use throughout the tutorial. *Chapter 2: Data Reduction* contains a quick example on how to reduce data using the DRAGONS command line tools. *Chapter 3: Reduction with API* shows how we can reduce the data using DRAGONS packages from within Python.

1.1 Software Requirements

Before you start, make sure you have DRAGONS properly installed and configured on your machine. You can test that by typing the following commands:

```
$ conda activate dragons
$ python -c "import astrodata"
```

Where `dragons` is the name of the conda environment where DRAGONS should be installed. If you have an error message, make sure:

- Anaconda or MiniConda is properly installed;
- A Conda Virtual Environment is properly created and is active;
- AstroConda (STScI) is properly installed within the Virtual Environment;
- DRAGONS was successfully installed within the Conda Virtual Environment;

1.2 Downloading the tutorial datasets

All the data needed to run this tutorial are found in the tutorial's data package:

http://www.gemini.edu/sciops/data/software/datapkgs/gsaoiimg_tutorial_datapkg-v1.tar

Download it and unpack it somewhere convenient.

```
cd <somewhere convenient>
tar xvf gsaoiimg_tutorial_datapkg-v1.tar
bunzip2 gsaoiimg_tutorial/playdata/*.bz2
```

The datasets are found in the subdirectory `gsaoiimg_tutorial/playdata`, and we will work in the subdirectory named `gsaoiimg_tutorial/playground`.

Note: All the raw data can also be downloaded from the Gemini Observatory Archive. Using the tutorial data package is probably more convenient but if you really want to learn how to search for and retrieve the data yourself, see the step-by-step instructions in the appendix, *Downloading from the Gemini Observatory Archive*.

1.3 About the dataset

The table below contains a summary of the dataset downloaded in the previous section. Note that for GSAOI, the dark current is low enough that there is no need to correct for it.

Science	S20170505S0095-110	Kshort-band, on target, 60 s
Flats	S20170505S0030-044 S20170505S0060-074	Lamp on, Kshort, for science Lamp off, Kshort, for science
Standard star	S20170504S0114-117	Kshort, standard star, 30 s

CHAPTER 2

Data Reduction with “reduce”

This chapter will guide you on reducing **GSAOI data** using command line tools. In this example we reduce a GSAOI observation of the resolved outskirt of a nearby galaxy. The observation is a dither-on-target with offset-to-sky sequence. Just open a terminal to get started.

While the example cannot possibly cover all situations, it will help you get acquainted with the reduction of GSAOI data with DRAGONS. We encourage you to look at the *Tips and Tricks* and *Issues and Limitations* chapters to learn more about GSAOI data reduction.

DRAGONS installation comes with a set of scripts that are used to reduce astronomical data. The most important script is called “”, which is extensively explained in the . It is through that command that a DRAGONS reduction is launched.

For this tutorial, we will be also using the following supplemental tools: “”, “”, “”, and “”.

2.1 The dataset

If you have not already, download and unpack the tutorial’s data package. Refer to *Downloading the tutorial datasets* for the links and simple instructions.

The dataset specific to this example is described in:

About the dataset.

Here is a copy of the table for quick reference.

Science	S20170505S0095-110	Kshort-band, on target, 60 s
Flats	S20170505S0030-044 S20170505S0060-074	Lamp on, Kshort, for science Lamp off, Kshort, for science
Standard star	S20170504S0114-117	Kshort, standard star, 30 s

Note: A master dark is not needed for GSAOI. The dark current is very low.

2.2 Set up the Local Calibration Manager

DRAGONS comes with a local calibration manager that uses the same calibration association rules as the Gemini Observatory Archive. This allows `reduce` to make requests to a local light-weight database for matching **processed** calibrations when needed to reduce a dataset.

Let's set up the local calibration manager for this session.

In `~/.geminidr/`, create or edit the configuration file `rsys.cfg` as follow:

```
[calibs]
standalone = True
database_dir = ${path_to_my_data}/gsaoiimg_tutorial/playground
```

This simply tells the system where to put the calibration database, the database that will keep track of the processed calibrations we are going to send to it.

Note: The tilde (~) in the path above refers to your home directory. Also, mind the dot in `.geminidr`.

Then initialize the calibration database:

```
caldb init
```

That's it! It is ready to use!

You can add processed calibrations with `caldb add <filename>` (we will later), list the database content with `caldb list`, and `caldb remove <filename>` to remove a file **only** from the database (it will **not** remove the file on disk). For more the details, check the Recipe System Local Calibration Manager documentation, .

2.3 Check files

For this example, all the raw files we need are in the same directory called `../playdata/`. Let us learn a bit about the data we have.

Ensure that you are in the `playground` directory and that the `conda` environment that includes DRAGONS has been activated.

Let us call the command tool "":

```
$ typewalk -d ../playdata/

directory:  /data/workspace/gsaoiimg_tutorial/playdata
     S20170504S0114.fits ............... (GEMINI) (GSAOI) (IMAGE) (RAW) (SIDEREAL)␣
↪(SOUTH) (UNPREPARED)
     ...
     S20170505S0030.fits ............... (AZEL_TARGET) (CAL) (DOMEFLAT) (FLAT)␣
↪(GEMINI) (GSAOI) (IMAGE) (LAMPON) (NON_SIDEREAL) (RAW) (SOUTH) (UNPREPARED)
     ...
     S20170505S0060.fits ............... (AZEL_TARGET) (CAL) (DOMEFLAT) (FLAT)␣
↪(GEMINI) (GSAOI) (IMAGE) (LAMPOFF) (NON_SIDEREAL) (RAW) (SOUTH) (UNPREPARED)
     ...
     S20170505S0095.fits ............... (GEMINI) (GSAOI) (IMAGE) (RAW) (SIDEREAL)␣
↪(SOUTH) (UNPREPARED)
     ...
     S20170505S0110.fits ............... (GEMINI) (GSAOI) (IMAGE) (RAW) (SIDEREAL)␣
↪(SOUTH) (UNPREPARED)
Done DataSpider.typewalk(..)
```

This command will open every FITS file within the folder passed after the `-d` flag (recursively) and will print an unsorted table with the file names and the associated tags. For example, calibration files will always have the `CAL` tag. Flat images will always have the `FLAT` tag. This means that we can start getting to know a bit more about our data set just by looking at the tags. The output above was trimmed for presentation.

2.4 Create File lists

This data set contains science and calibration frames. For some program, it could have different observed targets and different exposure times depending on how you like to organize your raw data.

The DRAGONS data reduction pipeline does not organize the data for you. You have to do it. DRAGONS provides tools to help you with that.

The first step is to create lists that will be used in the data reduction process. For that, we use "". Please, refer to the "" documentation for details regarding its usage.

2.4.1 A list for the flats

Let us create the list containing the domeflats:

```
$ dataselect --tags FLAT ../playdata/*.fits -o flats_Kshort.list
```

We know that our dataset has only one filter (Kshort). If our dataset contained data with more filters, we would have had to use the `--expr` option to select the appropriate filter as follow:

```
$ dataselect --tags FLAT --expr "filter_name=='Kshort'" ../playdata/*.fits -o flats_
↪Kshort.list
```

Note: To see the name of the filter, use "" (show descriptor):

```
$ showd ../playdata/*.fits -d filter_name
------------------------------------------------
filename                                filter_name
------------------------------------------------
../playdata/S20170504S0114.fits   Kshort_G1105&Clear
...
...
```

2.4.2 A list for the standard star

In this case we have only one standard star. Indeed, we can confirm that by selecting on partner calibrations and showing the object name:

```
$ dataselect --expr 'observation_class=="partnerCal"' ../playdata/*.fits | showd -d␣
↪object
---------------------------------------
filename                          object
---------------------------------------
../playdata/S20170504S0114.fits     9132
../playdata/S20170504S0115.fits     9132
../playdata/S20170504S0116.fits     9132
../playdata/S20170504S0117.fits     9132
```

If we had more than one object, a list for each standard star is created by using the `object` descriptor as a selection criterium in "":

```
$ dataselect --expr 'object=="9132"' ../playdata/*.fits -o std_9132.list
```

2.4.3 A list for the science observations

The rest is the data with your science target. Before we create a new list, let us check that indeed we have only one science target and a unique exposure time:

```
$ dataselect --expr 'observation_class=="science"' ../playdata/*.fits | showd -d␣
↪object,exposure_time
------------------------------------------------------------
filename                          object   exposure_time
------------------------------------------------------------
../playdata/S20170505S0095.fits   NGC5128            60.0
../playdata/S20170505S0096.fits   NGC5128            60.0
...
../playdata/S20170505S0109.fits   NGC5128            60.0
../playdata/S20170505S0110.fits   NGC5128            60.0
```

Just to demonstrate how expression are built, let us consider that we need to select only the files for which `object` is NGC5128 and `exposure_time` is 60 seconds. We also want to pass the output to a new list:

```
$ dataselect --expr '(observation_class=="science" and exposure_time==60.)' ../
↪playdata/*.fits -o science.list
```

2.5 Create a Master Flat Field

The GSAOI Kshort master flat is created from a series of lamp-on and lamp-off dome exposures. They should all have the same exposure time. Each flavor is stacked (averaged), then the lamp-off stack is subtracted from the lamp-on stack and the result normalized.

We create the master flat field and add it to the calibration manager as follows:

```
$ reduce @flats_Kshort.list
$ caldb add S20170505S0030_flat.fits
```

The master flat file is found in two places: inside the same folder where you ran `reduce` and inside the `calibrations/processed_flats/` folder, for safekeeping. Here is an example of a master flat:

Note that this figure shows the masked pixels in white color but not all the detector features are masked. For example, the "Christmas Tree" on detector 2 can be easily noticed but was not masked.

2.6 Reduce Standard Star

The standard star is reduced essentially the same way as the science target (next section). The processed flat field that we added earlier to the local calibration database will be fetched automatically. Also, in this case the standard star was obtained using ROIs (Regions-of-Interest) which do not match the flat field. The software will recognize that the flat field is still valid and will crop it to match the ROIs.

```
$ reduce @std_9132.list
```

To stack, the tool `disco_stu` is needed for GSAOI. It is discussed later in this chapter.

```
$ disco `dataselect *_skyCorrected.fits --expr='observation_class=="partnerCal"'`
```

2.7 Reduce the Science Images

This is an observation of a galaxy with offset to sky. We need to turn off the additive offsetting of the sky because the target fills the field of view and does not represent a reasonable sky background. If the offsetting is not turned off *in this particular case*, it results in an over-subtraction of the sky frame.

Note: Unlike the other near-IR instruments, the additive `offset_sky` parameter is used by default to adjust the sky frame background for GSAOI instead of the multiplicative `scale_sky` parameter. It was found to work better when the sky background per pixel is very low, which is common due to the short exposure time needed to avoid saturating stars and the small pixel scale. The reader is encourage to experiment with `scale_sky` if `offset_sky` does not seem to lead to an optimal sky subtraction.

(Remember that when the source is extended, both parameters normally need to be turned off.)

The sky frame comes from off-target sky observations. We feed the pipeline all the on-target and off-target frames. The software will split the on-target and the off-target appropriately using information in the headers.

Once we have our calibration files processed and added to the database, ready for retrieval, we can run `reduce` on our science data.

```
$ reduce @science.list -p skyCorrect:offset_sky=False
```

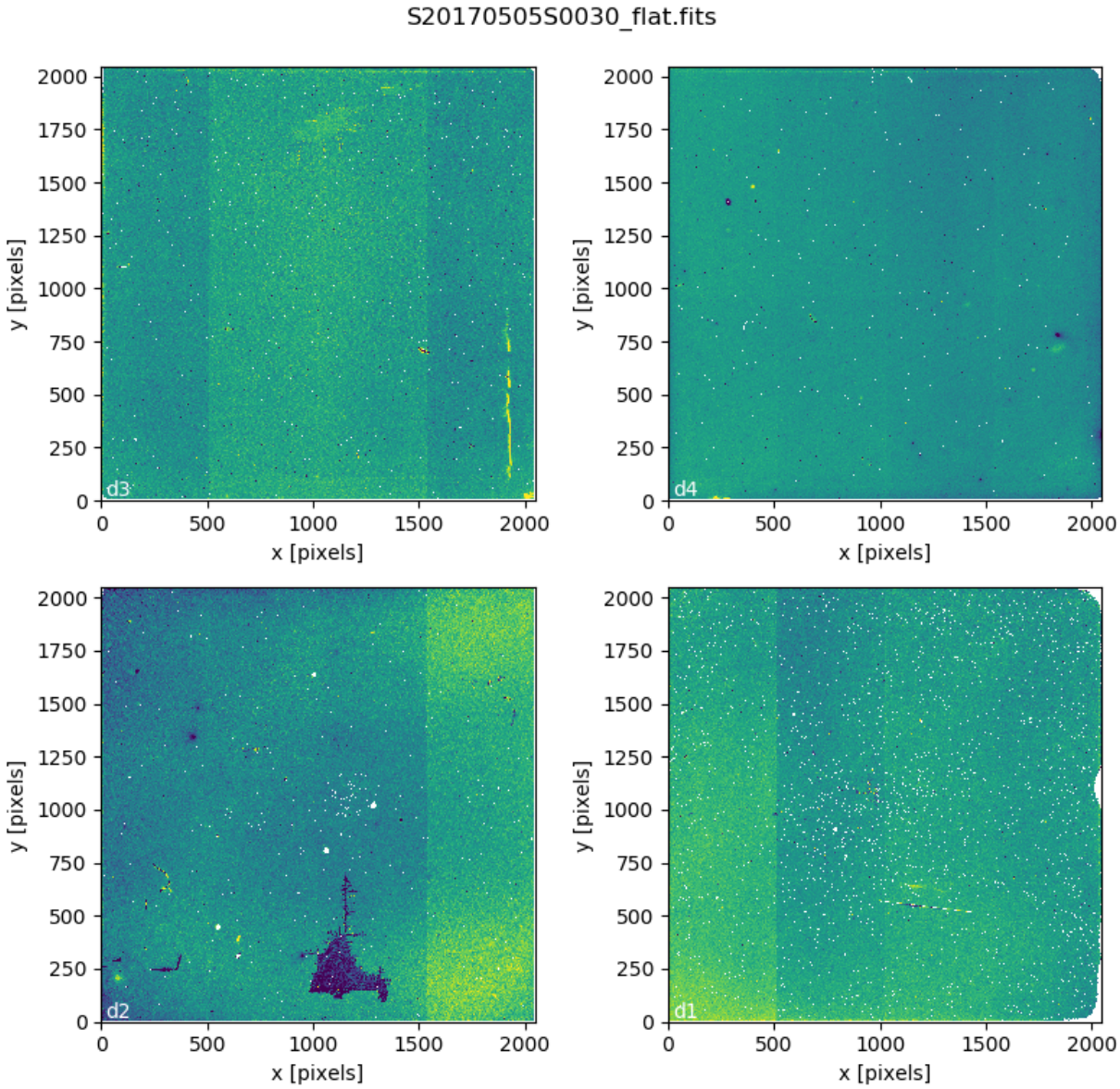

Fig. 1: Master Flat - K-Short Band

This command will generate flat corrected and sky subtracted files but will not stack them. You can find which file is which by its suffix (`_flatCorrected` or `_skyCorrected`). The on-target files are the ones that have been sky subtracted (`_skyCorrected`). There should be nine of them.

The frames are not stacked because of the high level of distortion in the GSAOI images that requires special software to correct and properly stack. The tool `disco_stu` (next section) must be used to stack GSAOI science data.

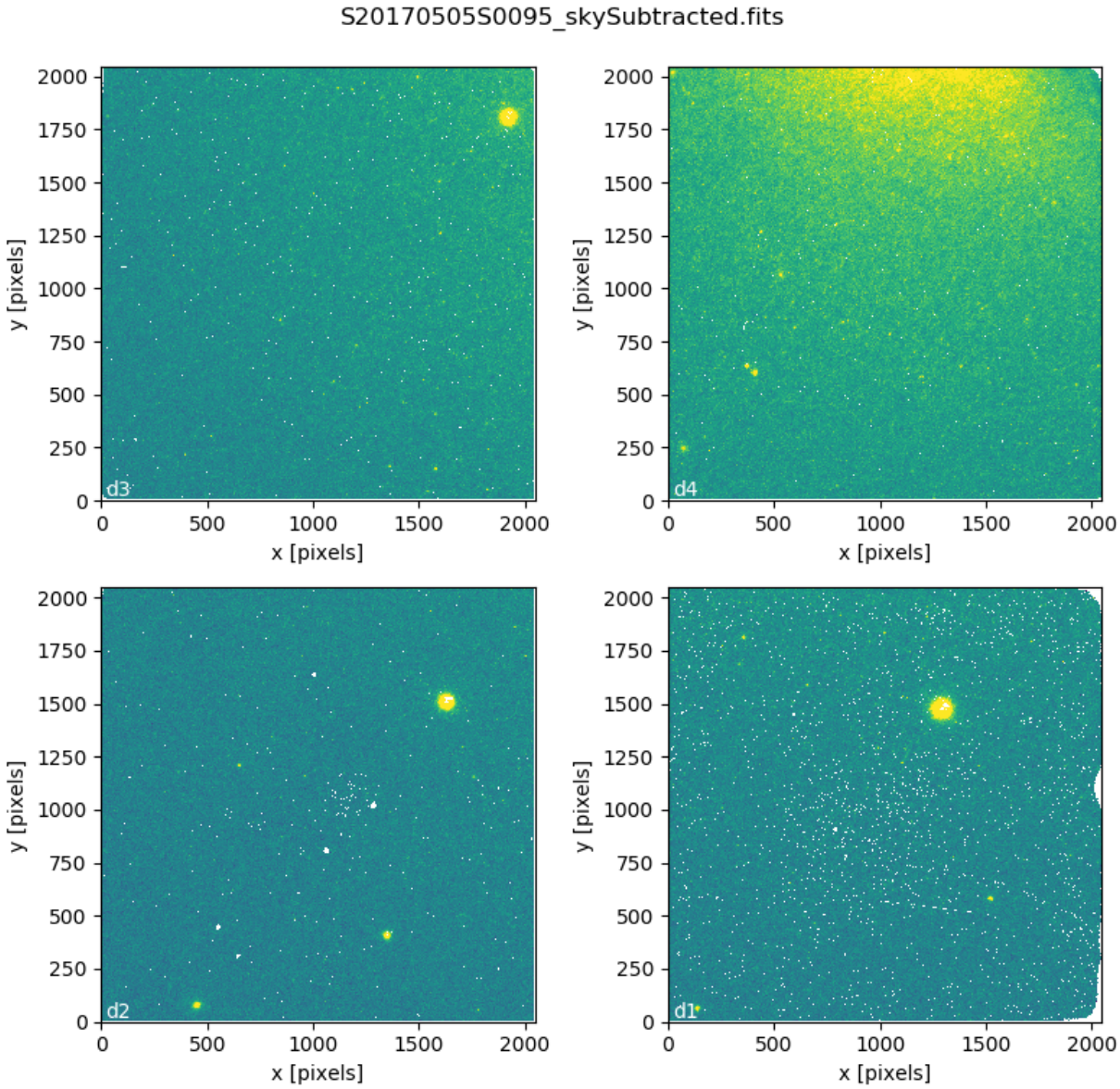

Fig. 2: S20170505S0095 - Flat corrected and sky subtracted

The figure above shows an example of the sky-subtracted frames. The masked pixels are represented in white color.

2.8 Stack Sky-Subtracted Science Images

The final step is to stack the images. For that, you must be aware that GSAOI images are highly distorted and that this distortion must be corrected before stacking. The tool for distortion correction and image stacking is `disco_stu`.

Note: `disco_stu` is installed with conda when the standard Gemini software installation instructions are followed. To install after the fact:

```
conda install disco_stu
```

Note: The `disco_stu` manual can be found at http://www.gemini.edu/sciops/data/software/disco_stu.pdf

---

The simplest use of `disco_stu` is to run the command `disco` on the files to be stacked.

```
$ disco `dataselect *_skyCorrected.fits --expr 'observation_class=="science"'` -o my_
↪Kshort_stack.fits
```

By default, `disco` will write the output file as `disco_stack.fits`, the `-o` flag allows us to override that and choose the name of the output stack.

For absolute distortion correction and astrometry, `disco_stu` can use a reference catalog provided by the user. Without a reference catalog, like above, only the relative distortion between the frames is accounted for. For more information about `disco_stu` see the `disco_stu.pdf` manual in `$CONDA_PREFIX/share/disco_stu`.

The output stack units are in electrons (header keyword BUNIT=electrons). The output stack is stored in a multi-extension FITS (MEF) file. The science signal is in the "SCI" extension, the variance is in the "VAR" extension, and the data quality plane (mask) is in the "DQ" extension.

The final image is shown below.

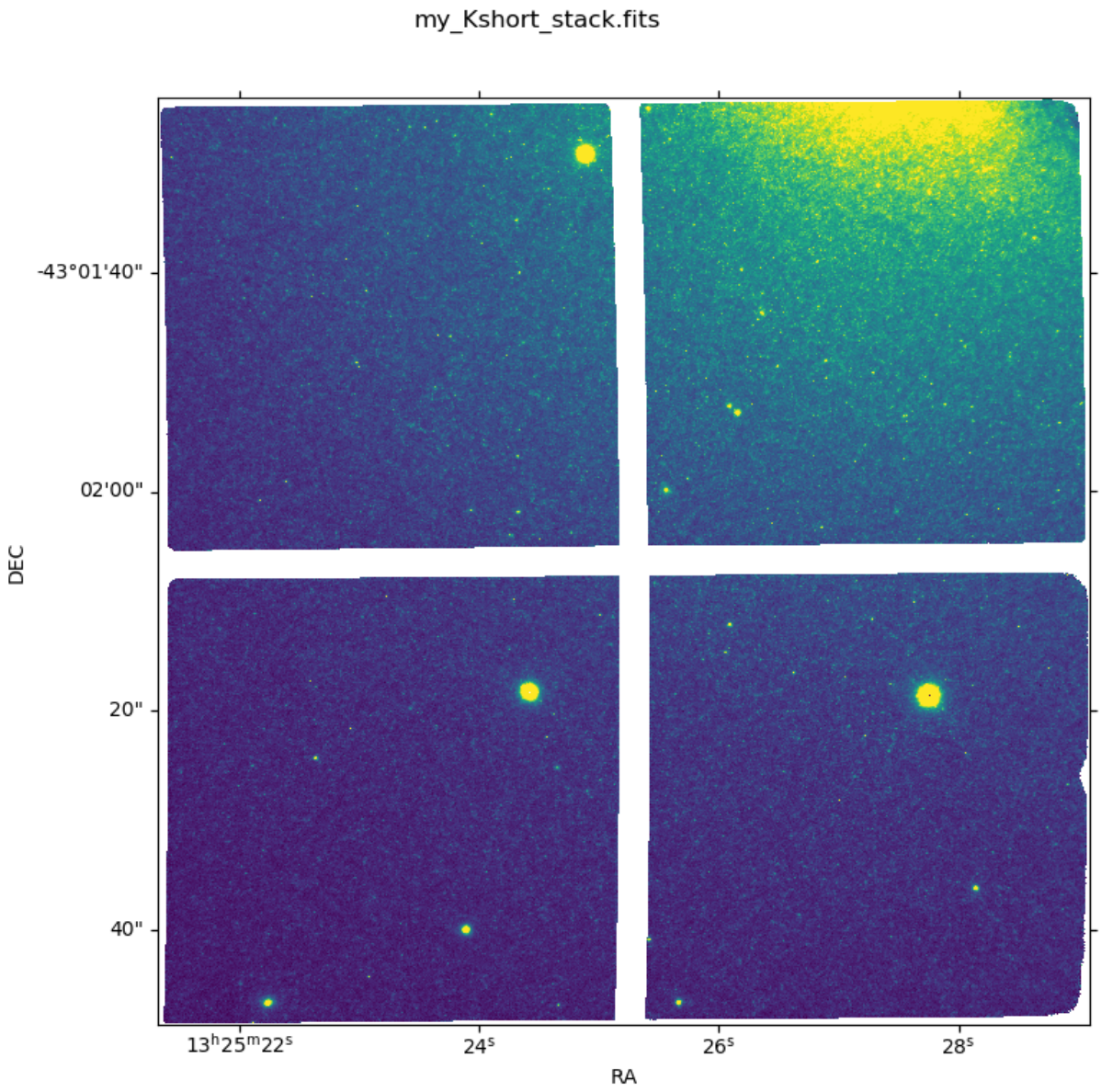

Fig. 3: Sky Subtracted and Stacked Final Image

CHAPTER 3

Reduction using API

There may be cases where you might be interested in accessing the DRAGONS' Application Program Interface (API) directly instead of using the command line wrappers to reduce your data. In this case, you will need to access DRAGONS' tools by importing the appropriate modules and packages.

3.1 The dataset

If you have not already, download and unpack the tutorial's data package. Refer to *Downloading the tutorial datasets* for the links and simple instructions.

The dataset specific to this example is described in:

> *About the dataset*.

Here is a copy of the table for quick reference.

Science	S20170505S0095-110	Kshort-band, on target, 60 s
Flats	S20170505S0030-044 S20170505S0060-074	Lamp on, Kshort, for science Lamp off, Kshort, for science
Standard star	S20170504S0114-117	Kshort, standard star, 30 s

Note: A master dark is not needed for GSAOI. The dark current is very low.

3.2 Setting up

3.2.1 Importing Libraries

We first import the necessary modules and classes:

```
import glob

from gempy.adlibrary import dataselect
from recipe_system import cal_service
from recipe_system.reduction.coreReduce import Reduce
```

`glob` is a Python built-in package. It will be used to return a `list` with the input file names.

`dataselect` will be used to create file lists for the darks, the flats and the science observations. The `cal_service` package is our interface with the local calibration database. Finally, the `Reduce` class is used to set up and run the data reduction.

3.2.2 Setting up the logger

We recommend using the DRAGONS logger. (See also *Double messaging issue*.)

```
from gempy.utils import logutils
logutils.config(file_name='gsaoi_data_reduction.log')
```

3.2.3 Setting up the Calibration Service

Before we continue, let's be sure we have properly setup our calibration database and the calibration association service.

First, check that you have already a `rsys.cfg` file inside the `~/.geminidr/`. It should contain:

```
[calibs]
standalone = True
database_dir = ${path_to_my_data}/gsaoiimg_tutorial/playground
```

This tells the system where to put the calibration database. This database will keep track of the processed calibrations as we add them to it.

Note: The tilde (~) in the path above refers to your home directory. Also, mind the dot in `.geminidr`.

The calibration database is initialized and the calibration service is configured as follow:

```
caldb = cal_service.CalibrationService()
caldb.config()
caldb.init()

cal_service.set_calservice()
```

The calibration service is now ready to use. If you need more details, check the section in the .

3.3 Create list of files

Next step is to create lists of files that will be used as input to each of the data reduction steps. Let us start by creating a `list` of all the FITS files in the directory `../playdata/`.

```
all_files = glob.glob('../playdata/*.fits')
all_files.sort()
```

Before you carry on, you might want to do `print(all_files)` to check if they were properly read.

Now we can use the `all_files list` as an input to `select_data()`. The `dataselect.select_data()` function signature is:

```
select_data(inputs, tags=[], xtags=[], expression='True')
```

3.3.1 A list for the flats

Now you must create a list of FLAT images for each filter. The expression specifying the filter name is needed only if you have data from multiple filters. It is not really needed in this case.

```
list_of_flats_Ks = dataselect.select_data(
     all_files,
     ['FLAT'],
     [],
     dataselect.expr_parser('filter_name=="Kshort"')
)
```

3.3.2 A list for the standard star

For the standard star selection, we use:

```
list_of_std_stars = dataselect.select_data(
    all_files,
    [],
    [],
    dataselect.expr_parser('observation_class=="partnerCal"')
)
```

Here, we are passing empty lists to the second and the third argument since we do not need to use the Tags for selection nor for exclusion.

3.3.3 A list for the science data

Finally, the science data can be selected using:

```
list_of_science_images = dataselect.select_data(
    all_files,
    [],
    [],
    dataselect.expr_parser('(observation_class=="science" and exposure_time==60.)')
)
```

The exposure time is not really needed in this case since there are only 60-second frames, but it shows how you could have two selection criteria in the expression.

3.4 Create a Master Flat Field

As explained on the calibration webpage for GSAOI, *dark subtraction is not necessary* since the dark noise level is very low. Therefore, we can go ahead and start with the master flat.

A GSAOI K-short master flat is created from a series of lamp-on and lamp-off exposures. Each flavor is stacked, then the lamp-off stack is subtracted from the lamp-on stack and the result normalized.

We create the master flat field and add it to the calibration manager as follow:

```
35 reduce_flats = Reduce()
36 reduce_flats.files.extend(list_of_flats_Ks)
37 reduce_flats.runr()
38
39 caldb.add_cal(reduce_flats.output_filenames[0])
```

Once `runr()` is finished, we add the master flat to the calibration manager (line 38).

3.5 Reduce Standard Star

The standard star is reduced essentially the same way as the science target (next section). The processed flat field that we added above to the local calibration database will be fetched automatically.

```
40 reduce_std = Reduce()
41 reduce_std.files.extend(list_of_std_stars)
42 reduce_std.runr()
```

For stacking the sky-subtracted standard star images, the easiest way is probably to use `disco_stu`'s command line interface as follow:

```
$ disco `dataselect *_skyCorrected.fits --expr='observation_class=="partnerCal"'`
```

If you really want or need to run `disco_stu`'s API, see the example later in this chapter where we do just that for the science frames.

3.6 Reduce the Science Images

The science observation uses a dither-on-target with offset-to-sky pattern. The sky frames from the offset-to-sky position will be automatically detected and used for the sky subtraction.

The master flat will be retrieved automatically from the local calibration database.

We use similar commands as before to initiate a new reduction to reduce the science data:

```
43 reduce_target = Reduce()
44 reduce_target.files.extend(list_of_science_images)
45 reduce_target.uparms.append(('skyCorrect:offset_sky', False))
46 reduce_target.runr()
```

3.7 Stack Sky-subtracted Science Images

The final step is to stack the images. For that, you must be aware that GSAOI images are highly distorted and that this distortion must be corrected before stacking. The tool for distortion correction and image stacking is `disco_stu`.

Note: `disco_stu` is installed with conda when the standard Gemini software installation instructions are followed. To install after the fact:

```
conda install disco_stu
```

Note: The `disco_stu` manual can be found at http://www.gemini.edu/sciops/data/software/disco_stu.pdf

---

This package was created to be accessed via command line (See the *Stack Sky-Subtracted Science Images* command line section). Because of that, the API is not the most polished, and using it requires a fair number of steps. **If you can use the command line interface, it is recommended that you do so.** If not, then let's get to work.

First, let's import some libraries:

```
from collections import namedtuple

from disco_stu import disco
from disco_stu.lookups import general_parameters as disco_pars
```

Then we need to create a special class using `namedtuple()`. This object will hold information about matching the objects between files:

```
MatchInfo = namedtuple(
    'MatchInfo', [
        'offset_radius',
        'match_radius',
        'min_matches',
        'degree'
        ])
```

We now create objects of `MatchInfo` class:

```
object_match_info = MatchInfo(
    disco_pars.OBJCAT_ALIGN_RADIUS[0],
    disco_pars.OBJCAT_ALIGN_RADIUS[1],
    None,
    disco_pars.OBJCAT_POLY_DEGREE
)

reference_match_info = MatchInfo(
    disco_pars.REFCAT_ALIGN_RADIUS[0],
    disco_pars.REFCAT_ALIGN_RADIUS[1],
    disco_pars.REFCAT_MIN_MATCHES,
    disco_pars.REFCAT_POLY_DEGREE
)
```

Finally, we call the `disco()` function and pass the arguments.

```
disco.disco(
    infiles=reduce_target.output_filenames,
    output_identifier="my_Kshort_stack",
    objmatch_info=object_match_info,
    refmatch_info=reference_match_info,
    pixel_scale=disco_pars.PIXEL_SCALE,
    skysub=False,
)
```

This function has many other parameters that can be used to customize this step but further details are out of the scope of this tutorial.

CHAPTER 4

Tips and Tricks

This is a collection of tips and tricks that can be useful for reducing different data, or to do it slightly differently from what is presented in the example.

4.1 Sky Subtraction

For sky subtraction, there are two input parameters to `skyCorrect` that users should be aware of: `scale_sky` and `offset_sky`. Both serve to match the sky frames to the target frame before the subtraction. The first, `scale_sky` is multiplicative and is turned off by default for GSAOI, while the second, `offset_sky` is additive and is turned **on** by default for GSAOI.

The reason why `offset_sky` is favored for GSAOI is that often the flux in individual pixels can be very low and that is observed to make the multiplicative scale less accurate. In any case, from experience, it was found that `offset_sky==True` was more successful, more often, with GSAOI data, which is why it was set as the default.

Depending on the data and the science objectives, those two input parameters might have to be experimented with. The only combination we would not recommend is setting both of them on. (The software will not let you either.)

When there are offset to sky, it is likely to be because the target fills the field of view and there is no usable sky. In those cases, all sky scaling and offsetting should be turned off (`skyCorrect:scale_sky=False` and `skyCorrect:offset_sky=False`). There is no sky to measure in the target frame, any attempts at scaling or offsetting will result in an over subtraction of the sky.

CHAPTER 5

Issues and Limitations

5.1 Memory Issues

Some primitives use a lot of RAM memory and they can cause a crash. Memory management in Python is notoriously difficult. The DRAGONS's team is constantly trying to improve memory management within `astrodata` and the DRAGONS recipes and primitives. If an "Out of memory" crash happens to you, if possible for your observation sequence, try to run the pipeline on fewer images at the time, like for each dither pattern sequence separately.

Then to align and stack the pieces, run the `alignAndStack` recipe:

```
$ reduce @list_of_stacks -r alignAndStack
```

5.2 Double messaging issue

If you run the `Reduce` API without setting up a logger, you will notice that the output messages appear twice. To prevent this behaviour set up a logger. This will send one of the output stream to a file, keeping the other on the screen. We recommend using the DRAGONS logger located in the `gempy.utils.logutils` module and its `config()` function:

```
from gempy.utils import logutils
logutils.config(file_name='gsaoi_data_reduction.log')
```

CHAPTER 6

Downloading from the Gemini Observatory Archive

For this tutorial we provide a pre-made package with all the necessary data. Here we show how one can search and download the data directly from the archive, like one would have to do for their own program.

If you are just interested in trying out the tutorial, we recommend that you download the pre-made package (*Downloading the tutorial datasets*) instead of getting everything manually.

6.1 Step by step instructions

For this tutorial we selected data observed for for the GS-2017A-Q-29 program on the night starting on May 04, 2017.

6.1.1 Science data

Access the GOA webpage.

In the search form, enter the following information:

- **Program ID**: GS-2017A-Q-29
- **UTC Date**: 20170504-20170505
- **Obs. Class**: science

The search will return 16 files. Download them all by pressing the "Download all 16 files" button at the bottom.

6.1.2 Calibrations

Matching calibration files can be obtained by clicking on the *Load Associated Calibrations* tab. For this data, domeflats (lamp on and off) and a standard star observation.

The first four files are the standard star sequence. The other files are the lamp on and lamp off domeflats.

The table returned by the automatic calibration association has all that we need. Download everything by pressing the "Download all 34 files" button at the bottom.

6.1.3 Unpacking

Now, copy all the `.tar` files to the same place in your computer. Then use `tar` and `bunzip2` commands to decompress them. For example:

```
$ cd ${path_to_my_data}/
$ tar -xf gemini_data.tar
$ bunzip2 *.fits.bz2
```

(The tar files names may differ slightly depending on how you selected and downloaded the data from the Gemini Archive.)

Note: If you are using the manually selected data to run the tutorial, please remember to put all the data in a directory called `playdata`, and create a parallel directory of running the tutorial called `playground`. The tutorial makes assumption as to where everything is located.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search