Goodman HTS Pipeline User Manual Documentation

Release 1.3.2

Simon Torres R.

Oct 09, 2020

USER MANUAL

This is the User Manual for the *Goodman Spectroscopic Data Reduction Pipeline*. It provides an overview of the pipeline's main features, instructions on its use and how to run it on our dedicated *Data Reduction Server*, and installation instructions for those who wish to run it on their own computers.

CHAPTER

ONE

OVERVIEW

The Goodman Spectroscopic Data Reduction Pipeline - *The Goodman Pipeline* - is a Python-based package for producing science-ready, wavelength-calibrated, 1-D spectra. The goal of *The Goodman Pipeline* is to provide SOAR users with an easy to use, very well documented software for reducing spectra obtained with the Goodman High Troughput Spectrograph. Though the current implementation assumes offline data reduction, our aim is to provide the capability to run it in real time, so 1-D wavelength calibrated spectra can be produced shortly after the shutter closes.

The pipeline is primarily intended to be run on a data reduction dedicated computer though it is available for local installation. The *Goodman Spectroscopic Pipeline* project is hosted at GitHub at it's GitHub Repository.

Instructions for running the software are provided in the *Usage* section of this guide. How to access the the data reduction server is on *Setup for Remote Use* or if you prefer to install a local version instructions are in *Install*

Currently the pipeline is separated into two main components. The initial processing is done by `redccd`, which does the following processess.

- Identifies calibrations and science frames.
- Create master bias.
- Creates master flats and normalizes it.
- Apply overscan correction.
- Trims the image.
- For spectroscopic data find slit edges and trims again.
- Applies bias correction.
- Applies flat correction.
- Applies cosmic ray removal.

The spectroscopic processing is done by `redspec` and carries out the following steps:

- Identifies point-source targets.
- Traces the spectra.
- Extracts the spectra.
- Estimates and subtract background.
- Saves extracted (1D) spectra, without wavelength calibration.
- Finds the wavelength solution.
- Linearizes data (resample)
- Writes the wavelength solution to FITS header
- Creates a new file for the wavelength-calibrated 1D spectrum

CHAPTER

TWO

USAGE

The *Goodman Spectroscopic Pipeline* is designed to be simple to use, however simple does not always is the best case for everyone, thus *The Goodman Pipeline* is also flexible.

Getting Help. This manual is intended to be the prefered method to get help. However the quickest option is using `-h` or `--help`

`redccd --help`

Will print the list of arguments along with a quick explanation and default values.

It is the same for `redspec`

`redspec --help`

2.1 Observing Guidelines

In order to be able to process your data with the *Goodman Spectroscopic Pipeline* you need to follow some guidelines, we do not intend to tell you how to do your science. Here are some basic hints.

- Make sure you have a good observing plan as well as a good backup plan too.
- Put special attention to the calibration files that are needed for the data that you are planning to obtain, for instance, you can process your spectroscopic data without bias because using overscan will give you a good enough approximation, but Imaging does not have overscan therefore you MUST obtain bias frames.
- Keep a detailed log of things that happened while you were observing, mistakes that you made, exposures repeated, etc. An observing log is not an extraction of header information. Well, it can be, but it will be useless.
- If you are unsure about the required steps to achieve your science goals ask your PI, not the support scientist, Her/His job is to assist you on how to get good quality data not what data you need in order to achieve your scientific goals.

For using the pipeline you don't need to use any special file naming convention, in fact all the information is obtained from the headers. As of version `1.2.0` you need to use a reference lamp naming convention though. Not the file but the field that goes into OBJECT. It is actually very simple:

Table 1: Convention names for comparison lamps

Lamp name	Convention
Argon	Ar
Neon	Ne
Copper	CuHeAr
Iron	FeHeAr
Mercury Argon	HgAr
Mercury Argon Neon	HgArNe

This is to ensure the pipeline is able to recognize them. This will no be the case in future versions but for now this is how it works.

2.2 Observing for Radial Velocity

Radial velocity measurements are possible with the Goodman High Throughput Spectrograph but you have to be careful. A very detailed description of the procedures and what you can expect was prepared and is available HERE.

Please read it carefully so you don't find any surprises when trying to reduce your data.

2.3 Prepare Data for Reduction

If you did a good job preparing and doing the observation this should be an easy step, either way, keep in mind the following steps.

- Remove all *focus* sequence.
- Remove all *target acquisition* or *test* frames.
- Using your observation's log remove all unwanted files.
- Make sure all data has the same gain (`GAIN`) and readout noise (`RDNOISE`)
- Make sure all data has the same Region Of Interest or ROI (`ROI`).

The pipeline does not modify the original files unless there are problems with fits compliance, is never a bad idea to keep copies of your original data in a safe place.

2.3.1 Updating Keywords

Since version *1.3.0* if your data is older than **August 6, 2019**, you will need to change the following keywords.

- `SLIT`: Replace whitespaces with underscore, remove " and all letters are uppercase. For instance `0.45" long slit` becomes `0.45_LONG_SLIT`.
- `GRATING`: Grating's *lines/mm* goes first and then the manufacturer. For instance. `SYZY_400` becomes `400_SYZY`.
- `WAVMODE`: Replace whitespace with underscore and all letters are capitalized. For instance. `400 m1` becomes `400_M1`.

Note: General rules are: Underscore is the only accepted separator. All letter must be upper case. Remove any character that need escaping.

2.4 Processing your 2D images

It is the first step in the reduction process, the main tasks are listed below.

- Create master bias
- Create master flats
- Apply Corrections:
  - Overscan
  - Trim image
  - Detect slit and trim out non-illuminated areas
  - Bias correction
  - Normalized flat field correction
  - Cosmic ray rejection

Note: Some older Goodman HTS data has headers that are not FITS compliant, In such cases the headers are fixed and that is the only modification done to raw data.

The 2D images are initially reduced using `redccd`. You can simply move to the directory where your raw data is located and do:

```
redccd
```

Though you can modify the behavior in several ways.

Running `redccd` will create a directory called `RED` where it will put your reduced data. If you want to run it again it will prevent you from accidentally removing your already reduced data unless you use `--auto-clean` this will tell the pipeline to delete the `RED` directory and start over.

```
redccd --auto-clean
```

A summary of the most important *command line arguments* are presented below.

- `--cosmic <method>` Let you select the method to do *Cosmic Ray Removal*.
- `--debug` Show extended messages and plots of intermediate steps.
- `--flat-normalize <method>` Let you select the method to do *Flat Normalization*.
- `--flat-norm-order <order>` Set order for the model used to do *Flat Normalization*. Default 15.
- `--ignore-bias` Ignores the existence or lack of `BIAS` data.
- `--ignore-flats` Ignores the existence or lack of `FLAT` data.
- `--raw-path <path>` Set the directory where the raw data is located, can be relative.
- `--red-path <path>` Set the directory where the reduced data will be stored. Default `RED`.
- `--saturation <saturation>` Set the saturation threshold in percentage. There is a table with all the readout modes and their values at which saturation is reached, then all the pixels exceeding that value are counted. If the percentage is larger that the threshold defined with this argument the flat is marked as saturated. The default value is 1 percent.

This is intended to work with *spectroscopic* and *imaging* data, that it is why the process is split in two.

2.5 Extracting the spectra

After you are done *Processing your 2D images* it is time to extract the spectrum into a wavelength-calibrated 1D file.

The script is called `redspec`. The tasks performed are the following:

- Classifies data and creates the match of `OBJECT` and `COMP` if it exists.
- Identifies targets
- Extracts targets
- Saves extracted targets to 1D spectrum
- Finds wavelength solution automatically
- Linearizes data
- Saves wavelength calibrated file

First you have to move into the `RED` directory, this is a precautionary method to avoid unintended deletion of your raw data. Then you can simply do:

```
redspec
```

And the pipeline should work its magic, though this might not be the desired behavior for every user or science case, we have implemented a set of *command line arguments* which are listed below.

- `--data-path <path>` Folder were data to be processed is located. Default is *current working directory*.
- `--proc-path <path>` Folder were processed data will be stored. Default is *current working directory*.
- `--search-pattern <pattern>` Prefix for picking up files. Default `cfzsto`. See *File Prefixes*.
- `--extraction <method>` Select the *Extraction Methods*. The only one implemented at the moment is `fractional`.
- `--reference-files <path>` Folder where to find the reference lamps.
- `--debug` Shows extended and more messages.
- `--debug-plot` Shows plots of intermediate steps.
- `--max-targets <value>` Maximum number of targets to detect in a single image. Default is 3.
- `--save-plots` Save plots.
- `--plot-results` Show plots during execution.

The mathematical model used to define the wavelength solution is recorded in the header even though the data has been linearized for record purpose.

2.6 Description of custom keywords

The pipeline adds several keywords to keep track of the process and in general for keeping important information available. The following table gives a description of all the keywords added by *The Goodman Pipeline*, though not all of them are added to all the images.

2.6.1 General Purpose Keywords

These keywords are used for record purpose, except for `GSP_FNAM` which is used to keep track of the file name.

Table 2: General purpose keywords, added to all images at the moment of the first read.

Keyword	Purpose
GSP_VERS	Pipeline version.
GSP_ONAM	Original file name, first read.
GSP_PNAM	Parent file name.
GSP_FNAM	Current file name.
GSP_PATH	Path from where the file was read.
GSP_TECH	Observing technique. Imaging or Spectroscopy.
GSP_DATE	Date of processing.
GSP_OVER	Overscan region.
GSP_TRIM	Trim section.
GSP_SLIT	Slit trim section. From slit-illuminated area.
GSP_BIAS	Master bias file used.
GSP_FLAT	Master flat file used.
GSP_SCTR	Science target file name (for lamps only)
GSP_LAMP	Reference lamp used to obtain wavelength solution
GSP_NORM	Master flat normalization method.
GSP_COSM	Cosmic ray rejection method.
GSP_TERR	RMS error of target trace
GSP_EXTR	Extraction window at first column
GSP_BKG1	First background extraction zone
GSP_BKG2	Second background extraction zone
GSP_WRMS	Wavelength solution RMS Error.
GSP_WPOI	Number of points used to calculate RMS Error.
GSP_WREJ	Number of points rejected from RMS Error Calculation.
GSP_DCRR	Reference paper for DCR software (cosmic ray rejection).

2.6.2 Target Trace Model

Table 3: Keywords used to describe the model used to fit the target's trace.

Keyword	Purpose
GSP_TMOD	Name of mathematical model from astropy's `modeling`
GSP_TORD	Order of the model used.
GSP_TC00	Value of parameter `c0`.
GSP_TC01	Value of parameter `c1`.
GSP_TC02	Value of parameter `c2`. This goes on depending the order.

2.6.3 Non-linear wavelength solution

Since writing non-linear wavelength solutions to the headers using the FITS standard (reference) is extremely complex and not necessarily well documented, we came up with the solution of simply describing the mathematical model from astropy's `modeling`. This allows for maintaining the data *untouched* while keeping a reliable description of the wavelength solution.

The current implementation will work for writting any polinomial model. Reading is implemented only for `Chebyshev1D` which is the model by default.

Table 4: Keywords used to describe a non-linear wavelength solution.

Keyword	Purpose
GSP_FUNC	Name of mathematical model from astropy's `modeling`
GSP_ORDR	Order of the model used.
GSP_NPIX	Number of pixels.
GSP_C000	Value of parameter `c0`.
GSP_C001	Value of parameter `c1`.
GSP_C002	Value of parameter `c2`. This goes on depending the order.

2.6.4 Combined Images

Every image used in a combination of images is recorded in the header of the resulting one. The order does not have importance but most likely the header of the first one will be used.

The combination is made using the `combine()` method with the following parameters

- `method='median'`
- `sigma_clip=True`
- `sigma_clip_low_thresh=1.0`
- `sigma_clip_high_thresh=1.0`

At this moment these parameters are not user-configurable.

Table 5: Keywords that list all the images used to produce a combined image.

Keyword	Purpose
GSP_IC01	First image used to create combined.
GSP_IC02	Second image used to create combined.

2.6.5 Detected lines

The *reference lamp library* maintains the lamps non-linearized and also they get a record of the pixel value and its equivalent in angstrom. In the following table a three-line lamp is shown.

Table 6: Description of all the keywords used to list lines in lamps in Pixel and Angstrom.

Keyword	Purpose
GSP_P001	Pixel value for the first line detected.
GSP_P002	Pixel value for the second line detected.
GSP_P003	Pixel value for the third line detected.
GSP_A001	Angstrom value for the first line detected.
GSP_A002	Angstrom value for the second line detected.
GSP_A003	Angstrom value for the third line detected.

2.7 Cosmic Ray Removal

Warning: The parameters for either cosmic ray removal method are not fully understood neither tuned but they work for most common instrument configurations. If your extracted spectrum shows weird features, specially if you use a custom mode, the most likely culprit are the parameters of the method you chose. Please let us know.

The argument `--cosmic <method>` has four options but there are only two real methods.

`default` (default): Different methods work different for different binning. So if `<method>` is set to `default` the pipeline will decide as follows:

`dcr` for binning `1x1`

`lacosmic` for binning `2x2` and `3x3` though binning `3x3` has not being tested.

`dcr`: It was already said that this method work better for binning `1x1`. More information can be found on dcr. The disadvantages of this method is that is a program written in C and it is required to write the file to the disk, process it and read it back again. Still is faster than `lacosmic`.

The parameters for running `dcr` are written in a file called `dcr.par` a lookup table and a file generator have been implemented but you can parse custom parameters by placing a `dcr.par` file in a different directory and point it using `--dcr-par-file <path>`.

`lacosmic`: This is the preferred method for files with binning `2x2` and `3x3`. This is the Astroscrappy's implementation and is run with the default parameters. Future versions might include some parameter adjustment.

`none`: Skips the cosmic ray removal process.

Asymetric binnings have not been tested but the pipeline only takes in consideration the dispersion axis to decide. This does not mean that the spatial binning does not impact the performance of any of the methods, we just don't know it yet.

2.8 Flat Normalization

There are three possible `<method>` (s) to do the normalization of master flats. For the method using a model the default model's order is `15`. It can be set using `--flat-norm-order <order>`.

`mean`: Calculates the mean of the image using numpy's `mean()` and divide the image by it.

`simple` (default): Collapses the master flat across the spatial direction, fits a `Chebyshev1D` model of order `15` and divide the full image by this fitted model.

full: Fits a `Chebyshev1D` model to every line/column (dispersion axis) and divides it by the fitted model. This method takes too much to process and it has been left in the code for experimentation purposes only.

2.9 Extraction Methods

The argument `--extraction <method>` has two options but only `fractional` is implemented.

fractional: *Fractional pixel extraction* differs from a simple and rough extraction in how it deals with the edges of the region. `goodman_pipeline.core.core.extract_fractional_pixel()`

optimal: Unfortunately this method has not been implemented yet.

2.10 File Prefixes

There are several ways one can do this but we selected adding prefixes to the file name because is easier to add and also easy to filter using a terminal, for instance.

```
ls cfzsto*fits
```

or in python

```
import glob

file_list = glob.glob('cfzsto*fits')
```

So what does all those letter mean? Here is a table to explain it.

Table 7: Characters and meaning of prefixes

Letter	Meaning
o	Overscan Correction Applied
t	Trim Correction Applied
s	Slit trim correction applied
z	Bias correction applied
f	Flat correction applied
c	Cosmic rays removed
e	Spectrum extracted to 1D
w	1D Spectrum wavelength calibrated

So, for an original file named `file.fits`:

```
o_file.fits
```

Means the file have been overscan corrected while

```
eczsto_file.fits
```

Means the spectrum has been extracted to a 1D file but the file has not been flat fielded (`f` missing).

Ideally after running `redccd` the file should be named:

```
cfzsto_file.fits
```

And after running `redspec`:

```
wecfzsto_file.fits
```

2.11 File Suffixes

After extraction, suffixes may appear in new files created. There are two scenarios where this can happen:

- More than one spectroscopic target for extraction. `*target_X`.
- More than one comparison lamp. `*ws_Y`.
- Both above

Let's consider the following scenario: We start with 3 reduced files.

Table 8: Sample files in example.

File Name	Obstype	Comment
sci_file.fits	OBJECT	Science file with two spectra.
lamp_001.fits	COMP	Reference lamp valid for sci_file.fits
lamp_002.fits	COMP	Another valid reference lamp

Assuming the two targets in *sci_file.fits* are extracted we'll end up with

CHAPTER

THREE

SETUP FOR REMOTE USE

The Goodman Spectroscopic Data Reduction Pipeline has been installed on a dedicated computer at SOAR. The procedure requires to open a VNC session, for which you need to be connected to the SOAR VPN. The credentials for the VPN are the same you used for your observing run, provided by your *Support Scientist*, who will also give you the information for the data reduction computer VNC connection.

Note: IRAF is available in the data server at SOAR. Running `iraf` will open an *xgterm* and *ds9* windows. `iraf-only` will open *xgterm* but not *ds9*

3.1 Establish a VNC connection

Separately, you should receive a server hostname, IP, display number and VNC-password.

Table 1: VNC display number and working folder assigned to each partner.

Display	Partner/Institution	Folder
:1	NOAO	`/home/goodman/data/NOAO`
:2	Brazil	`/home/goodman/data/BRAZIL`
:3	UNC	`/home/goodman/data/UNC`
:4	MSU	`/home/goodman/data/MSU`
:5	Chile	`/home/goodman/data/CHILE`

For this tutorial we will call the vnc server host name as `<vnc-server>` the display number is `<display-number>` and your password is `<password>`.

The VNC connection should work with any VNC Client like TightVNC, TigerVNC, RealVNC, etc. The first two run on Linux and can be used directly with the `vncviewer` command line.

Important: Please, help us to create an organized enviroment by creating a new folder using the format `YYYY-MM-DD` within your institution's directory and using it to process your data.

3.2 VNC from the Terminal

Find the <display-number> that corresponds to you from the *VNC Displays table*. Open a terminal, and assuming you have installed vncviewer.

```
vncviewer <vnc-server>:<display-number>
```

You will be asked to type in the <password> provided.

Important: The real values for <vnc-server> and <password> should be provided by your support scientist.

If the connection succeeds you will see a *Centos 7* Desktop using *Gnome*.

CHAPTER

FOUR

INSTALL

We do not have the resources to provide installation support, thus we provide a server with the latest and older versions installed that users with access rights can use. However, the installation process is simple, and is described below (we hope that in enough detail) so that users can follow the steps and end up with a running version of the pipeline.

Note: In this tutorial we use Miniconda3 as but you can also do it with Anaconda by visiting https://www.anaconda.com/download/ for downloading the Anaconda installer for either operating system.

4.1 Ubuntu

This installation process has been tested in a live version (previously known as *Live CD*) of Ubuntu 18.04.

1. Install `make` and `gcc`

   ```
   sudo apt install gcc make
   ```

2. Download Miniconda

   ```
   wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.
   sh -O miniconda.sh
   ```

4.2 Mac OS

This installation was tested on MacOS High Sierra.

1. Install Xcode command Line Tools. You can do it from the *App Store* or from command line.

   ```
   xcode-select --install
   ```

2. Download Anaconda or Miniconda

   ```
   curl https://repo.continuum.io/miniconda/Miniconda3-latest-MacOSX-x86_64.
   sh --output miniconda.sh
   ```

4.3 Common Steps

Warning: This installation instructions work with **BASH** only.

1. Install Miniconda

   `bash miniconda.sh`

   Answer the questions and reopen the terminal at the end.

2. Configure conda to use the Astroconda Channel

   `conda config --add channels http://ssb.stsci.edu/astroconda`

3. Get the latest release of the *Goodman Spectroscopic Pipeline* from github. There is a `.zip` and `.tar.gz` files, the following steps will continue with the latter.

   Make sure there is a green tag that says Latest release. For instance, for the version `v1.1.2` the file is called.

   `goodman_pipeline-1.2.1.tar.gz`

4. Extract the file (you can't copy and paste this line)

   `tar -xvf goodman_pipeline-<latest tag>.tar.gz`

   where `<latest tag>` is the version number of the latest release.

5. Move into the directory containing the package.

   `cd goodman_pipeline-<latest tag>`

   If you do `ls` you should find some interesting files on it such as: `setup.py` and `environment.yml` and `install_dcr.sh`.

6. Create the virtual environment. The `environment.yml` file contains a preset definition of a virtual environment that Anaconda will understand, also ensures that the *Goodman Spectroscopic Pipeline* will work. Even the name of the virtual environment is set there.

   `conda env create -f environment.yml`

   This will create a virtual environment called `goodman_pipeline`. To activate it

   `source activate goodman_pipeline`

7. Install `DCR`. This script requires a virtual environment activated.

   `sh install_dcr.sh`

   To test if it worked you can do:

   `dcr`

   You should get something like this:

```
(goodman_pipeline) [user@servername goodman_pipeline]$ dcr

     This is a modified version of DCR! for the Goodman Spectroscopic␣
↪Pipeline
     Please visit the author's site to get the original version:
     Modification Version: 0.0.1

     http://users.camk.edu.pl/pych/DCR/
```

(continues on next page)

(continued from previous page)

```
     USAGE:  dcr  input_file  cleaned_file  cosmicrays_file

File 'dcr.par' must be present in the working directory.
    ~~~~~~
```

8. Run tests.

   ```
   python setup.py test
   ```

9. Install the pipeline

   ```
   python setup.py install
   ```

4.4 Using pip

The *Goodman Spectroscopic Pipeline* Can also be installed using pip, but it does not install `dcr`, so if you are updating your goodman pipeline only you can use:

```
pip install goodman-pipeline
```

CHAPTER

FIVE

LICENSE

BSD 3-Clause License

Copyright (c) 2018, SOAR Telescope

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER

SIX

AUTHORS AND CREDITS

6.1 Development Team

- Simón Torres (SOAR Telescope Data Analyst - main code developer)
- César Briceño (SOAR Telescope Scientist - team lead)
- Bruno Quint (Brazil Support Astronomer - code development adviser)

Bruno Quint dedicated part of his time as post-doc to this project. Given that, Bruno Quint would like to acknowledge **CNPq** for the fellowship which allowed him to contribute to the development of the pipeline.

6.2 Contributors

- David Sanmartim (Gemini Astronomer)
- Tina Armond (Brazil Support Astronomer)

We acknowledge the important contribution of David Sanmartim, who developed the initial incarnation of the redccd module. We thank Tina Armond for her invaluable help in adding calibrated comparison lamps to the library of reference comparison lamps for wavelength solution.

CHAPTER

SEVEN

ACKNOWLEDGEMENTS

Our work would not be possible without the friendly work atmosphere at CTIO headquarters in La Serena, were we can interact with our SOAR and CTIO colleagues in lively and useful discussions that have been important in making the Goodman pipeline possible. We also acknowledge fruitful discussions and suggestions from our colleagues Bart Dunlop, Chris Clemens, and Erik Dennihy, at University of North Carolina at Chapel Hill.

CHAPTER

EIGHT

QUESTIONS & ANSWERS

1. What is the Goodman High Throughput Spectrograph?.

   This is thoroughly documented in SOAR web site and links within.

2. How does the pipeline select the reference lamp?.

   The lamps are selected comparing two keywords. `OBJECT` and `WAVMODE`

3. How should I organize the data?.

   More than organized your data should be *cleaned* of undesired files. There are some general assumptions in the implementation of the pipeline's data organization system that might get confused by files that are not supposed to be there.

4. What is *slit trim*?.

   Is a process to trim the 2D spectroscopic images to the *slit illuminated area* only. It works by fitting a box function to the dispersion-axis-collapsed spatial profile.

   The box function is `Box1D` . The reason for doing it is because the non-illuminated area causes all sorts of problems in later steps, such as: existence of `nan` in master flats.

CHAPTER

NINE

CHANGE HISTORY

9.1 V1.3.2 09-10-2020

- Fixed Github Actions setup
- Removed pandas version constraint and implemented workaround to be able to use latest pandas version.
- Modified installation of dcr on travis and Github Actions

9.2 V1.3.1 23-09-2020

- Documentation improvements:
  - More docstrings
  - New **`File Suffixes`_** section
- Added more test code.
- Cleaned .travis.yml and created special dcr installation script for travis.
- Changed the way *core.combine_data* names new files.
- Fixed version checker due to deprecation of access token as url parameter.
- New *core.identify_technique*. Was developed in the web application context.
- Created *–skip-slit-trim* argument to provide more control for certain use cases.
- Removed python 3.5 because it will not be supported anymore.
- Improved AEON Support:
  - Added values for OBSTYPE and required logic.
  -
- Bugs Fixed:
  - Serial and Parallel binning extraction from header was not working
  - Changed url for astroplan's server

9.3 V1.3.0 06-03-2020

- Made it compatible with Astropy 4.0
- All versions are free except for Pandas [#314]
- *wavelength.WavelengthCalibration.__call__* can now return a json output.
- *core.setup_logging* can now create a generic logger (same format).
- Modified how master bias are named.
- Removed bias overscan and trimming correction on master bias creation.
- Bugs Fixed:
  - *–max-targets* was not being used, missed connection in *MainApp*.
- Updated keyword values of all reference lamps in the library according to [#292]
- Refactored *wavelength.WavelengthCalibration* class moving several methods to *core* [#300, #303]
- Refactored *wavelength.WavelengthCalibration* to be instantiated without arguments.
- Improved messages at critical stages of wavelength calibration.
- Moved *setup_logging* call from main package *__init__* to scripts or entry points, this allows to re use other master loggers.
- Changed *–background-threshold* to multiply by detection limit instead of background level
- Created standard JSON output for `WavelengthCalibration`.

9.4 V1.2.1 19-08-2019

- Bugs fixed
  - Bias process was not fully ignored when *–ignore-bias* was used [#289].
  - *pandas* version was not specified in *environment.yml* [#288, #290]
  - Target extraction failed for low signal targets because background subtraction was being ignored at the step of actually identifying targets.
- Install instructions updated [#290]
- Moved static methods from *ImageProcessor* to *core*.
- Added function to validate ccd regions using regular expressions.
- Using lamps keywords to select reference lamps.
- Replaced *target_stddev* by *target_fwhm* in function *extract* and *extract_fractional*.
- Replaced *nsigmas* by *nfwhm* everywhere.
- Added argument *–background-threshold* with default value *3*.
- Added argument *–fit-targets-with* with options *moffat* and *gaussian*.

9.5 V1.2.0 26-10-2018

- Bugs removed:
  - If there was more than one lamp for a science target the lamp recorded as used was one of them only.
  - A percentage symbols was added to the help of `--saturation` argument, this caused a crash when `redccd -h` or `redccd --help` was used.
- Numpy is fixed to the version `1.15.2` until further notice.
- Reference lamps now get the extraction window added to the end of the file name. This is to avoid overwritting the lamps when they were used for more than one target.
- DCR install script is now more advanced and requires a virtual environment to work on.
- Added SOAR Logo to ReadTheDocs page.
- Changed install instruction with exact steps and commands instead of referencing documentation.
- Improved method to detect saturated images. Added a table with the *half full well* for all the readout modes possible and created a method to easily retrieve the value. This is a big improvement since in earlier versions the saturation limit was set to 65000 ADU regardless the input data and the user had to set a different one using the argument `--saturation`.
- Repurposed the `--saturation` command line argument, now is used to define the percentage of pixels above the saturation level, which for simplicity is the value of half full well. A default value of 1 percent was set as default.
- Added record information of target trace into the header and logs.
- Added record of background extraction regions into the header and logs.
- Made all plots full screen and the images using the `gray` cmap.
- Trace information is printed in the logs and also is recorded in the image's header
- Added sigma clipping to target tracing functions

9.6 V1.1.2 05-10-2018

- Version 1.1.2 is pip instalable

  ```
  pip install goodman-pipeline
  ```

- Project and package renamed to `goodman_pipeline` this is because the previous was too generic. Now we have this structure:

```
goodman_pipeline/
   docs/
   goodman_pipeline/
      core/
      images/
      ..etc
   setup.py
   ..etc
```

- Bugs Fixed:
  - `DataFrame` index is unusable when partial parts are eliminated. Added `index_reset(drop=True)`

  - Data conversion from string to integer needed to be converted to float first.
  - For low SNR data there was confusion of noise with targets, added a median filter and increased the the `order` value of peak detection.

- Created several new keywords:

  GSP_EXTR: Extraction window at the first column.

  GSP_SCTR: Used for extracted comparison lamps, contains the name of the file of science target that the lamp was extracted for.

  GSP_LAMP: For science targets, it records the name of the lamp used for the wavelength calibration.

- "Sliding" cross correlation window (to trace non-linearity of wavelength solution) is set to the maximum value between the length of the lamp spectrum in pixels and four times the global cross correlation of the reference lamp to the new one.
- Iterations in sigma clipping of differences between obtained wavelength values and laboratory values was increased from 1 to 3. This is for removing bad fitted lines and also RMS error calculation.
- Gaussian Kernel size for reference lamp convolution is now dependent on slit size and binning
- Added reference lamps for all gratings and their modes except `1200M0`
- Created script `install_dcr.sh`
- Increased code coverage
- Eliminated `None` elements in list of instances of `goodman_pipeline.core.core.NightDataContainer`
- Improved several logging messages
  - In general, it informs more, when it does an action and when it does not. What files are discarded,
  - Debugging plots are more complete for `identify_targets`.
- Created new argument `--debug-plot` dedicated for *graphical debugging*, the old `--debug` will show additional messages but will not produce any graphical output.
- Removed ability to process several folders in sequence, now the pipeline has to be run for each folder separately.

9.7 V1.1.1 23-08-2018

- Bugs Fixed:
  - Added clean exit when pipeline is unable to determine `instrument` or `technique` used.
  - Conversion from string to integer not always works, added intermediate float conversion.
  - Abrupt exit when there were non-fits-compliant keywords. Now it attempts to fix them all automatically and warns the user. Also, it ends the execution and informs the user to try again.
- Removed unused code and tools.
- Relocated module `goodman_pipeline.core.check_version` to `pipeline/core`.
- Implemented Authorized GitHub API access and added actual version check
- Moved *command line interface* from `goodman/bin/` to `goodman/pipeline/script/`
- Specified version of `cython` to be able to build.
- Added reference lamps for all usable modes for the grating 600 l/mm

- Created method to use automatic keyword fix from `ccdproc`.
- Improved help information of arguments
- Documentation updates

9.8 V1.1.0 24-07-2018

- Bugs fixed
  - `--keep-cosmic-file` would work for `dcr` but not for `lacosmic`
- Changed organization of ReadTheDocs information
  - New structure
  - Added references to external packages
  - This page is the single place to add changes information. CHANGES.md still exist but contains a link here.
- Added `--version` argument.
- Implemented *astroscrappy's* LACosmic method
- removed ccdproc's `cosmicray_lacosmic()`.
- created `default` method for cosmic ray rejection.
  - For binning 1x1 default is dcr
  - For binning 2x2 default is lacosmic
  - For binning 3x3 default is lacosmic

methods `dcr`, `lacosmic` or `none` can still be forced by using `--cosmic <method>`

9.9 V1.0.3 11-07-2018

- Bugs fixed
  - programatically access to the version number did not work because it was based purely on `setup.cfg` now `setup.py` has a function that creates the file `pipeline.version` which is accessed by `pipeline/__init__.py`
  - File naming was making some file dissapear by being overwritten for files that contained more than one target the next file name would match the previous one. A differentiator was added.

9.10 V1.0.2 10-07-2018

- Removed module `goodman/pipeline/info.py` and placed all metadata in `goodman/setup.cfg`.
- Several updates to documentation
  - Added comment on how to organize data on `soardata3`.
  - Added link to licence on footer.
  - User manual now is in ReadTheDocs and no longer available as a pdf.

  - Improved information on debug plots
- Bugs Fixed.
  - fixed `GSP_FNAM` value for reference lamps
  - Spectral limit calculation by including binning into the equation
  - Included binning in the calculation of the wavelength solution
  - Corrected messages and conditions under which the prefix for cosmic ray rejection is used
  - Image combination call and messages
- Other additions + Added lookup table `dcr.par` file generator and found optimal parameters for Red camera and binning 2x2

9.11 V1.0.1 xx-xx-2018

- Moved user manual from external repo to `goodman/docs/`
- Added version checker
- Centralised metadata (`__version__`, `__licence__`, etc) in `goodman/setup.cfg`
- Added `CHANGES.md`

9.12 V1.0.0 29-04-2018

- First production ready release

CHAPTER

TEN

API DOCUMENTATION

- genindex
- modindex
- search