Goodman HTS Pipeline User Manual

version 1.0

Simón Torres, César Briceño and Bruno Quint

May 07, 2018

Contents

Introduction

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.

License

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 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. Instructions for running the software are provided in the Running Pipeline section of this guide. The Goodman Spectroscopic Data Reduction Pipeline project is hosted at GitHub at it's GitHub Repository.

Currently the pipeline is separated into two main components. The initial processing is done by `redccd`, which trims the images, and carries out bias and flat corrections. 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

Features Available

- Self-contained, full data reduction package for the most commonly used predefined setups with Goodman HTS. Given the almost limitless number of possible configurations available with the Goodman instrument, only the most popular configurations will be supported, though we will try to add as many modes as possible.
- Python based, using existing Astropy libraries as much as feasible.
- Extensively documented, using general coding standards: PEP8 – Style Guide, PEP257 – Docstrings Convention (in-code documentation) – Google Style
- Multiplataform compatibility (tested on Linux Ubuntu, CentOS and MacOSX).
- Modular design. Could be used as a library within other Python applications.

Supported Data

We are currently supporting data obtained since March, 2018, however you should be able to process older data making sure you data meets the data requirements.

Future Implementation

Here is a list of the features we are planning to implement.

Additions:

- Extraction of extended sources
- Flux calibration module
- Source deblending
- *Live* version of *The Goodman Pipeline*.

Improvements:

- Method to calculate wavelength solution, giving more options in terms of models.
- Code clean up
- Documentation

Requirements

We present here the data and software requirements that assure that the pipeline can perform smoothly. We recommend to check your data and your system before running it. Otherwise, the pipeline can perform in an unexpected way or even crash.

Data Requirements

The Goodman High Throughput Spectrograph's data has seen some evolution in the past years, in shape and most importantly in its headers. The *Goodman Spectroscopic Pipeline* relies heavily on the data's header so this is in fact very important.

The headers must be FITS Compliant, otherwise the software exits with errors.

Bear in mind that the Goodman Spectrograph has two cameras, *Blue* and *Red*.

Recent data obtained with the Red Camera already meets all the requirements in terms of format and header cards. Data obtained with the Blue Camera before March, 2018 is expected to have several format issues:

1. There were non fits-compliant characters in some comments. To solve that, you can edit the header using the most recent version of AstroPy, IRAF or WCSTOOLS to remove the following keywords: `PARAM0`, `PARAM61`, `PARAM62` and `PARAM63`.
2. The data was defined as 3D, just like a single frame of a data cube. To solve this, you will have to read the data and rewrite it with only two dimensions using AstroPy or IRAF.
3. Some keywords were added with time.
    - INSTCONF: contains the name of the Goodman Camera used, e.g., "Blue" or "Red".
    - WAVMODE: contains the ruling number of the used grating and the mode name, e.g., "400 m1" or "400 m2".
    - ROI: the name of the region of interest, e.g., "Spectroscopic 1x1", "user-defined", etc.
4. Duplicated keywords. Make sure that your data does not contain duplicated keywords.

Reference Lamp Files

Having an *automatic wavelength calibration method* relies on having previously calibrated reference lamps obtained in the same configuration or mode. It is also important that the lamp names are correct, for instance `HgAr` is quite different than `HgArNe`. Spaces between lamps are not allowed. And the name is case insensitive: you may write "HgAr", "HGAR" or "hgar". The list of current lamps is the following.

List of Goodman Spectrograph supported modes

Grating	Mode	Filter	Lamp
400	M1	None	HgAr, HgArNe
400	M2	GG455	Ar, Ne, HgAr, HgArNe, CuHeAr, FeHeAr

> ***Important***
>
> More lamps will be made public shortly.

Headers Requirements

Goodman HTS spectra have small non-linearities on their wavelength solutions. They are small but must be taken into account.

It was necessary to implement a custom way of storing non-linear wavelength solutions that at the same time allowed for keeping data as *untouched* as possible. The main reason is that linearizing the reference lamps made it harder to track down those non-linearities on the new data being calibrated. Also, The documentation on how to write non-linear solution to a FITS header is far too complex for our use case and there is no apparent effort on trying to simplify it. Below we compile a list of required keywords for comparison lamps if they want to be used as reference lamps. The full list of keywords is listed under New Keywords.

General Custom Keywords:

Every image processed with the *Goodman Spectroscopic Pipeline* will have the general keywords. The one required for a reference lamp is the following:

```
GSP_FNAM = file-name.fits / Current file name
```

Record of detected lines in Pixel and Angstrom:

Reference lamps have a record of usable lines in its header. Initially the lamp is run through a tool that identifies the lines and records its pixel value. The root string is `GSP_P` followed by a zero-padded three digit sequential number (001, 002, etc). For instance.

```
GSP_P001= 499.5377036976768  / Line location in pixel value
GSP_P002= 810.5548319623747  / Line location in pixel value
GSP_P003= 831.6984711087946  / Line location in pixel value
```

Later, the equivalent values in angstrom are then recorded with the root string `GSP_A` and the same numerical pattern as before.

```
GSP_A001= 5460.75            / Line location in angstrom value
GSP_A002= 5769.61            / Line location in angstrom value
GSP_A003= 5790.67            / Line location in angstrom value
```

`GSP_P001` and `GSP_A001` are a match. If any of the angstrom value entries have a value of `0` (default value) the equivalent pair pixel/angstrom entry is ignored. Also they must be organized in an *always increasing* way, if they are not, they will be ignored too.

Important

Those keywords are used to calculate the mathematical fit of the wavelength solution and are not used on normal operation. Our philosophy here is that the line identification has to be done only once and then the model can be fitted several times, actually you can try several models if you want. (On your own)

Non-linear wavelength solution:

The method for recording the non-linear wavelength solution is actually very simple. It requires: `GSP_FUNC` which stores a string with the name of the mathematical model from `astropy.modeling.models`. `GSP_ORDR` stores the order or degree of the model. `GSP_NPIX` stores the number of pixels in the spectral axis. Then there is N+1 parameter keywords where N is the order of the model defined by `GSP_ORDR`. The root string of the keyword is `GSP_C` and the rest is a zero-padded three digit number starting on zero to N. See the example below.

```
GSP_FUNC= Chebyshev1D          / Mathematical model of non-linearized data
GSP_ORDR= 3                    / Mathematical model order
GSP_NPIX= 4060                 / Number of Pixels
GSP_C000= 4963.910057577853    / Value of parameter c0
GSP_C001= 0.9943952599223119   / Value of parameter c1
GSP_C002= 5.59241584012648e-08 / Value of parameter c2
GSP_C003= -1.2283411678846e-10 / Value of parameter c3
```

Warning

This method has been developed and tested to write correctly polynomial-like models. And ONLY reads `Chebyshev1D` models. Other models will just be ignored. More development will be done based on request, suggestions or needs.

File organization

redccd and redspec will look for all FITS files inside the current working directory or inside the path provided with the `--raw-path` (redccd)/`--data-path` (redspec) flag non-recursively. Make sure to have only data that contains relevant signal. Data obtained during the focusing process, saturated flats, etc, must be removed.

Also, we recommend you follow these good practices:

- Delete all unnecessary files (focus, test, acquisition, unwanted exposures, etc)
- Don't mix different ROI (Region Of Interest), Gain and Readout Noises.
- Make sure all the required file types are present: BIAS, FLAT, COMP, OBJECT.

Software Requirements

Using the pipeline remotely is the recommended method, in which case you don't need to worry about software requirements.

However, for users who wish to go ahead with a local installation, we provide simple instructions in the Setup for local installation section.

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*

Establish a VNC connection

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

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.

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*.

Setup for local installation

The *The Goodman Pipeline* is completely written in Python 3.x and relies on several libraries like:

- NumPy
- SciPy
- MatPlotLib
- Pandas
- AstroPy
- AstroPy/ccdproc
- AstroPy/astroplan
- DCR

We **do not** recommend the installation of these libraries or the *The Goodman Pipeline* in your system since updates and upgrades may ruin it. We rather recomment the use of Virtual Environments. If you are not familiar with this term, please check the official documentation by visiting the link below:

https://docs.python.org/3/tutorial/venv.html

or

http://docs.python-guide.org/en/latest/dev/virtualenvs/

Another option is to install **Conda**, a Virtual Environment Manager, or **AstroConda**, the same but for astronomers. Everything you need to know about installing both can be found in the link below:

https://astroconda.readthedocs.io/

Warning

You may find that *ccdproc* and *astroplan* do not come with Astroconda. They are not available on any Conda channel either. That means that you will have to install them separately. You can do so by downloading the source files and installing them by hand, or simply activate your Virtual Environment and then install these two packages using pip with

```
pip install ccdproc astroplan
```

System installation is not recommended because it can mess things up specially in Linux and Mac OS. Before you proceed, make sure that your system has all the required libraries, as described in Setup for local installation.

Once you have Python running and all the libraries installed either using Conda/AstroConda or not, you may download the last version available in the following address:

https://github.com/soar-telescope/goodman/releases/latest

Before continuing, make sure that your Virtual Environment is active if this is the case. There are several ways of doing this but normally the command below should work:

```
$ source activate <my_environment_name>
```

Where `<my_environment_name>` is the name of your Virtual Environment (e.g. astroconda).

Now you can finally install the *The Goodman Pipeline*. Download the file, decompress it, and enter the directory created during the file decompression. Test the installation by typing:

```
$ python setup.py test
```

If you have any errors, check the traceback. If you find difficulties carring on at this poing, you may contact us by opening a new issue or using the e-mail *goodman-pipeline@ctio.noao.edu*.

If no error messages start popping up in your screen, you are good to carry on with the installation.

```
$ python setup.py install
```

Note

This will install the pipeline in the currently active Python version. If you have Virtual Environments, make sure that they are active. If not, you can add the `--user` option to install only for your user and avoid needing root access.

DCR (optional)

Acknowledgement Note

Please cite: Pych, W., 2004, PASP, 116, 148

In terms of cosmic ray rejection we shifted to a non-python package because the results were much better compared to LACosmic's implementation in Astropy. LACosmic was not designed to work with spectroscopy.

The latest version of the Goodman Spectroscopic Pipeline uses a modified version of `dcr` to help with the pipeline's workflow. It is included under

```
<path_to_download_location>/goodman/pipeline/data/dcr-source/dcr/
```

`goodman` is the folder that will be created once you untar or unzip the latest release of the *The Goodman Pipeline*.

> ***Important***
>
> The changes we made to DCR include deletion of all `HISTORY` and `COMMENT` keywords, which we don't use in the pipeline. And addition of a couple of custom keywords, such as: `GSP_FNAM`, which stores the name of the file being created. `GSP_DCRR` which stores the reference to the paper to cite.

You are still encouraged to visit the official Link. We remind again that users of the Goodman Pipeline should cite the DCR paper with the reference indicated above.

Compiling DCR

Compiling `dcr` is actually very simple.

```
cd <path_to_download_location>/goodman/pipeline/data/dcr-source/dcr/
```

Then simply type:

```
make
```

This will compile *dcr* and also it will create other files. The executable binary here is `dcr`.

We have successfully compiled *dcr* right out the box in several platforms, such as:

- Ubuntu 16.04
- Centos 7.1, 7.4
- MacOS Sierra
- Solaris 11

Installing the DCR binary

This is a suggested method. If you are not so sure what you are doing, we recommend you follow the steps shown below. If you are a more advanced user and you want to do it your own way, all you have to achieve is to have the `dcr` executable binary in your `$PATH` variable.

1. Open a terminal
2. In your home directory create a hidden directory `.bin` (Home directory should be the default when you open a new terminal window)

   ```
   mkdir ~/.bin
   ```
3. Move the binary of your choice and rename it `dcr`. If you compiled it, most likely it's already called `dcr` so you can ignore the renaming part of this step.

   ```
   mv dcr.Ubuntu16.04 ~/.bin/dcr
   ```

   Or

   ```
   mv dcr ~/.bin/dcr
   ```
4. Add your `$HOME/.bin` directory to your `$PATH` variable. Open the file `.bashrc` and add the following line.

   ```
   export PATH=$PATH:/home/myusername/.bin
   ```

   Where `/home/myusername` is of course your home directory.
5. Close and reopen the terminal or load the `.bashrc` file.

   ```
   source ~/.bashrc
   ```

Running the Pipeline

The *Goodman Spectroscopic Pipeline* is designed to be simple to use, however help is always necessary.

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
```

Working with Virtual Environments

Virtual environments are a very useful tool, the main contribution of them being:

- Portability
- Protection to the host environment
- Flexibility

If you know nothing about them we recommend you to start in the Conda site.

For the purpose of this manual we will just say that a *Virtual Environment* lets you have a custom set of libraries/tools in one place, and most importantly is independent of your host system. Installation will not be discussed here but you can visit this link for information.

Discover what environments exist in your system.

```
conda env list
```

Will print a list where the first column is the name.

Activate (enter) the virtual Environment.

```
source activate <venv-name>
```

Where `<venv-name>` is the name of your virtual environment. Your shell's prompt will change to:

```
(<venv-name>) [user@hostname folder-name]$
```

Deactivate (leave) the virtual environment.

```
source deactivate
```

This time the prompt will change again to:

```
[user@hostname folder-name]$
```

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 though.

Run redccd

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

Run redspec

Is the spectroscopy reduction script. The tasks 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

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

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.

General Purpose Keywords

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

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_NORM	Master flat normalization method.
GSP_COSM	Cosmic ray rejection method.
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).

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.modeling.models`. 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.

Keywords used to describe a non-linear wavelength solution.

Keyword	Purpose
GSP_FUNC	Name of mathematical model. `astropy.modeling.models`
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.

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

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.

Detected lines

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

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.

Acknowledgements

We acknowledge the important contribution of David Sammartin, 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 wavelngth solution.

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.