Goodman HTS Pipeline User Manual

version 0.10

Simón Torres, César Briceño and Bruno Quint

April 24, 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

License is under discussion

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 multiple targets (spectra of more than one object in the slit)
- Trace the spectra
- Extract the spectra
- Estimate and subtract background
- Saves extracted (1D) spectrum, without wavelength calibration.
- Find the wavelength solution. Defaults to automatic wavelength solution, but can be done interactively
- Linearize data (resample)
- Write wavelength solution to FITS header
- Create 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

To write

Future Implementation

To add

Requirements

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 *The Goodman Pipeline* relies heavily on the data's header so this is in fact very important.

The headers must be FITS Compliant first of all, if not the software exits with errors.

Remember that the Goodman Spectrograph has **two** cameras, *Blue* and *Red*.

The Red camera does not create any problems.

The Blue camera instead had some issues until **ESTIMATED DATE**. They can be simplified in three groups.

- There were non fits-compliant characters in some comments.
- The data was defined as 3D, just like a single frame of a data cube.
- There were several differences in keyword names and some other did not exist.

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`. The list of current lamps is the following.

List of Goodman Spectrograph supported modes

Grating	Mode	Filter	Lamp
400	M1	None	HgAr
400	M1	None	HgArNe
400	M2	GG455	Ar
400	M2	GG455	Ne
400	M2	GG455	HgAr
400	M2	GG455	HgArNe
400	M2	GG455	CuHeAr
400	M2	GG455	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 big enough that they **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 harder to track down those non-linearities on the new data being calibrated and also; The documentation on how to write non-linear solution to a FITS header is not good, besides it appears that nobody is trying to improve it neither trying to implement it. Below I 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:

Every line detected in the reference lamp is recorded both in its pixel value and later (most likely entered by hand) in angstrom 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
```

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.

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

There is no special requirements for files but you will avoid problems if you follow these points.

- 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, we provide simple instructions below.

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 all three data servers. 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. If you don't you can ask for it. We have decided to use a similar organization of vnc displays as for `soaric7`:

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>`.

We are not recommending a particular *VNC Client* since there are several options, so if you feel that you are not getting the best image quality feel free to explore different clients. For this tutorial we use `vncviewer`.

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

6. Get latest release of the *Goodman Spectroscopic Pipeline*

   visit https://github.com/soar-telescope/goodman/releases/latest and download the `*.zip` or `*.tar.gz`

   ```
   cd <download_location>
   tar -xvf <pipeline_file>.tar.gz
   ```

   or

   ```
   unzip <pipeline_file>.zip
   ```

7. Install requirements from `requirements.txt`

   ```
   cd <goodman_pipeline_unpacked_location>
   pip install -r requirements.txt
   ```

8. Install the pipeline

   ```
   pip install .
   ```

9. Upgrading the pipeline

   ```
   pip install . --upgrade
   ```

DCR (optional)

Warning

Don't forget to 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 though.

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 *Goodman Spectroscopic Pipeline*.

Important

The changes includes 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 own by the author and let me remind you once more that you have to cite the paper mentioned several times in this manual.

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* in several platforms, such as:

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

Install binary DCR

This is a suggested method. If you are not so sure what you are doing, we recommend you following this suggestion. If you are a more advanced user you just need 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
   ```

System Installation (not recommended)

System installation is not recommended because can mess things up specially in Mac OS. If you are really committed to install the pipeline in your system we recommend the Conda Installation

6. Get latest release of the *Goodman Spectroscopic Pipeline*

   visit https://github.com/soar-telescope/goodman/releases/latest and download the `*.zip` or `*.tar.gz` file.

   ```
   cd <download_location>
   ```

   ```
   tar -xvf goodman-<version>.tar.gz
   ```

   or

   ```
   unzip goodman-<version>.zip
   ```

7. Install requirements from `requirements.txt`

   ```
   cd goodman-<version>
   ```

   ```
   pip install -r requirements.txt
   ```

8. Install the pipeline

   ```
   pip install .
   ```

9. Upgrading the pipeline

```
pip install . --upgrade
```

Conda Installation

We strongly recommend installing the pipeline using *virtual environments*. Below you will find a summary of installation steps.

> **Warning**
>
> Remember that we are not providing any kind of support for installation. After this documentation you are on your own.

The following list provides a summary of all the steps (follow the links for instructions).

- Install Anaconda
- Add astroconda channel
- Create virtual environment
- Activate environment
- Install requirements
- Install pipeline

New Virtual Environment

Creating virtual environments is well documented on the Conda documentation site just make sure you are using `Python 3.5` or `3.6`. which are the versions against *The Goodman Pipeline* is regularly tested.

Existing Virtual Environment

We provide a predefined environment through a `environment.yml` file that you can use to create a virtual environment with all the pipeline's dependencies. It goes as follows:

```
conda create -f environment.yml
```

The new environment will be called `goodman`.

Pipeline Installation

Finally, in order to install *The Goodman Pipeline* using a virtual environment you need to activate it first.

```
source activate goodman
```

And in case you used a different name replace `goodman` by the name of your environment.

In a terminal go to `<path to download>/goodman-<version>/`, then:

```
python setup.py test
```

If all the tests run successfully you can then install the pipeline with:

```
python setup.py install
```

Running the Pipeline

Working with Virtual Environments

Prepare Data for Reduction

Run redccd

Run redspec

Description of custom keywords

The pipeline adds several keywords to keep track of the process and in general for keeping important information available. In the following table is 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 way it is currently implemented will work for writting for any polynomial kind of 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 paramter `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.

Conclusion

Performance

Benchmarks

(Time elapsed)

Wavelength Calibration

Comparison with IRAF

Acknowledgements

Acknowledge: Simon, Cesar, Bruno, Tina, David, DCR, cite papers.

Known Issues