User Manual for Extract_Orders.py
Current version: V0.0.11

Neil Cook
Physics, Astronomy and Maths, University of Hertfordshire

July 7, 2017

Contents

1 Introduction **1**

2 Installation and dependencies **2**
2.1 With Anaconda and python 2.7 2
2.2 Module dependencies if not using Anaconda 2

3 Running the code **3**
3.1 Using default values/Entering at run time 3
3.2 Entering command line options 3
3.3 Description of variables 4

4 The Extraction **5**
4.1 Min-max clipping 5
4.2 Locating Orders 5
4.3 Removing unwanted orders 7
4.4 The outputted, extracted spectra 7

5 Extracting from previously fitted image **10**

1 Introduction

This program is designed to take a fits image file (i.e. a CCD image) and extract out orders from an Echelle spectrograph (regardless of separation and curvature, as long as orders are distinguishable from one-another).

2 Installation and dependencies

2.1 With Anaconda and python 2.7

This program was written to work with modules installed in Anaconda 2 (https://www.continuum.io/downloads) and python 2.7. It is recommended to use an installation of anaconda 2 to use this program.
Once Anaconda is installed one other module is needed (tqdm), if anaconda is installed correctly then pip can be used to install this module

```
pip install tqdm
```

2.2 Module dependencies if not using Anaconda

This program relies on the following modules in python 2.7

This program relies on the following modules in python 2.7 (with earliest tested version listed)

- numpy (1.11.2)
- astropy (1.1.2)
- matplotlib (1.5.1)
- tqdm (4.4.0)
- skimage (0.12.3)
- Tkinter

Or in python 3.4

- numpy (1.12.1)
- astropy (1.3.2)
- matplotlib (2.0.2)
- tqdm (4.11.2)
- tkinter
- skimage (0.13.0)

Other versions may work but these are the ones tested.

3 Running the code

To run the code, change to the folder containing the Extract_Orders.py file. This program can be run four ways:

1. Using default values (or modifying default values in the .py file - Not recommended)
2. Modifying the values in the config file ('config.txt' in the .py file directory)
3. Entering options at run time via user interface (once python file has started)
4. Entering command line options

Note that any in-putted during the running of the code will override the command line inputs which will override the config file, which will override the values in the python code

The priority is given in the following order (i.e. the lowest value overrides higher values)

1. Input at run time
2. Input on command line
3. Input in config file
4. Input in the python file

As such currently all inputs are defined in the config file and therefore changing the python file will do nothing. To use default values just comment variables out with a '#' (See Section 3.3 for description of the variables) and follow instructions in the config file regarding format.

3.1 Using default values/Entering at run time

Type:

```
python Extract_Orders_V#.#.#.py
```

First option the program gives is:

```
Set up input parameters? [Y]es [N]o:
```

If yes the program will display current default values, describe the values and ask if the user wishes to change the values (Values are described in Section 3.3 below.)

3.2 Entering command line options

This program allows values to be entered on the command line
Type:

```
python Extract_Orders_V#.#.#.py argument=value
```

where argument is selected from the list in Section 3.3 and value is the value to be set in the program.

i.e.

```
python Extract_Orders_V#.#.#.py files="CCDimage1.fits" xblur=4 yblur=0.25
```

3.3 Description of variables

Variables that this program required are as follows:

1. 'filepath' = string, location of the folder containing fits files to extract"
2. 'plotpath' = string, location of place to save plots"
3. 'savepath' = string, location to save extracted order fits files"
4. 'files' = list or string, names of image fits files to extract"
5. 'order_direction' = string, direction of the orders.
   If 'horizontal' or 'H' or 'h' then orders are assumed to increase along the x axis
   If 'vertical' or 'V' or 'v' then orders are assumed to increase along the y axis
6. 'xblur' = float, blur pixels in wavelength direction (to aid finding orders), if xblur and yblur are zero not Gaussian fit will be applied.
7. 'yblur' = float, blur pixels in order direction (to aid finding orders), if xblur and yblur are zero not Gaussian fit will be applied.
8. 'pcut' = float, the percentile to mask at, 50 is equivalent to the median (i.e. all pixels below this will not be used in making the polynomial fit)
9. 'fitmin' = integer, lowest order polynomial to fit
10. 'fitmax' = integer, highest order polynomial to fit
11. 'width' = integer, number of pixels to use for width of each order
12. 'minpixelsinorder' = minimum number of pixels to use order (else it is discarded as noise/unusable)

4 The Extraction

Once the variables have been set (see Section 3) the code begins to extract the orders.

4.1 Min-max clipping

The first user option is min-max clipping. The aim of this is to produce an image where the orders are **as clearly visible and distinguishable** from the background as possible.
The program will display the following stats in the title:

```
      Min max clipping
low sigma = X high sigma = X

           Stats :
   Median is X  Std is X
 Low is at X  High is at X
```

and a matplotlib greyscale window of the original image.
The low clip is by default set to the median and the high clip is set to the median plus two standard deviations (median + 2std).
In the right hand panel two text entry boxes appear, with two buttons below "update" and "next".
Low and High are calculated using the following form:

$$Low = median(image) - lowsigma \times std(image) \tag{1}$$

$$High = median(image) + highsigma \times std(image) \tag{2}$$

the graph will refresh each time the user presses the "update" button. Repeat these steps until the user is satisfied with the image.
All values below lowsigma are set to 0, all values above highsigma are set to the value High.

IMPORTANT NOTE: This clipping is needed to identify the orders and thus the user should try to produce an image where the orders are **as clearly visible and distinguishable** from the background as possible (Sometimes this requires a negative low value (indicating the user wants to use a value above the median, see Equation 1).
Once the user is done press "next" to go to the next step.

4.2 Locating Orders

The next user interface may take some time to first load (depending on the size of your CCD image and the computational power of your comptuer). The code uses a python skimage called measure.label (see here http://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.label) algorithm to locate discrete 'regions' in the modified (min-max clipping) image.
Again a UI should pop up and the title should show:

```
            Found orders
   horizontal blur = X [pixels]
```

Figure 1: Image where regions would not be easily identifiable.

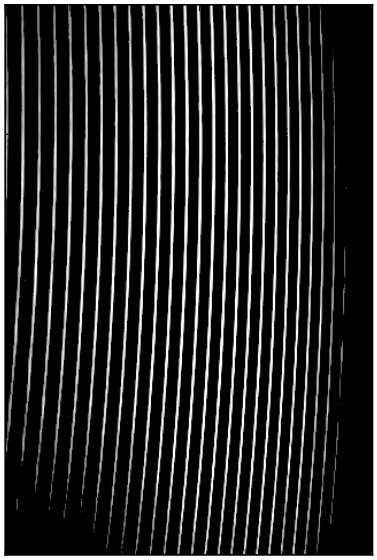

Figure 2: Image where clipping produces identifiable distinct regions.

```
vertical blur = X [pixels]
cut percentile = X [%]
min pixels selected = X [pixels]
```

The blur is done in a Gaussian fashion (see http://scikit-image.org/docs/dev/api/skimage.filters.html#skimage.filters.gaussian_filter) where:

$$\text{sigma} = [\text{horizontal blur, vertical blur}] \tag{3}$$

The cut is then applied using the 'percentile' (pcut) variable. This creates a mask:

$$mask = \begin{cases} 1 & \text{if pixel value} > \text{percentile} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

and additional requirement is that each "blob" as "min pixels selected" number of pixels in its region. If it has less it will not be identified as a useful "blob". **This mask is used to define unique regions (where touching pixels values = 1). Thus one should choose values of horizontal blur, vertical blur and percentile such that the orders are separated (the graph displayed will show the regions found).**
To update the values use the "update" button (again this may take some time to process). To accept changes click the "next" button.

Figure 4 shows a good region selection.

4.3 Removing unwanted orders

This loads up another user interface with the fitted orders shown in red. Orders are calculated by fitting a polynomial fit (numpy.polyfit) to each order (for powers between 'fitmin' and 'fitmax', see 3.3) the chosen polynomial fit is the polynomial with the lowest chi squared value.
The user interface will have one text entry box and three buttons. To remove orders list them in the text entry box (separated by a white space or a comma). All orders that the user wishes to remove must be entered, deleting an order from this text entry box will re-add it to the kept orders.
Pressing the "update" button will update the figure with the remaining orders. "Start fitting process again" will restart all steps (from Section 3.1 onwards). Pressing the "Accept Orders" button will proceed to running the extraction with the remaining orders (currently shown in the figure). You **must** click "update" to remove orders **before** clicking "Accept Orders".

(IMPORTANT: Look out for stray orders in the middle or at the edges of the graph) one can simply start the extraction again or remove some badly fitted or incorrect orders (see Figure 5).

4.4 The outputted, extracted spectra

The program will produce two types of output. It will produce 'pixel number' vs 'counts' graphs and fits files for each extracted order (in the location defined in 'plotpath' and 'savepath' respectively, see 3.3) , see example of a plot in Figure 7.
Fits files have columns: 'pixel number' and 'counts'.

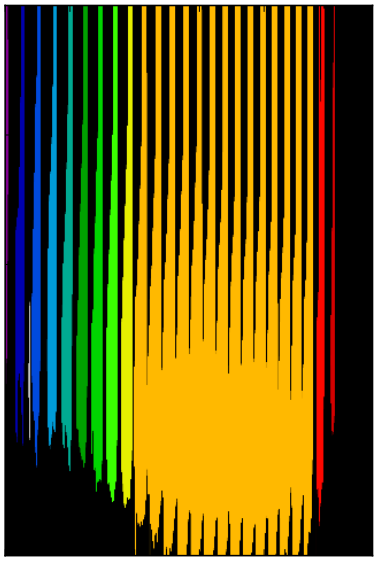

Figure 3: Example of a bad region identification (distinct regions are detectable and thus Gaussian blur parameters (horizontal/vertical blue) OR the percentile cut parameter (percentile) need to be changed.)

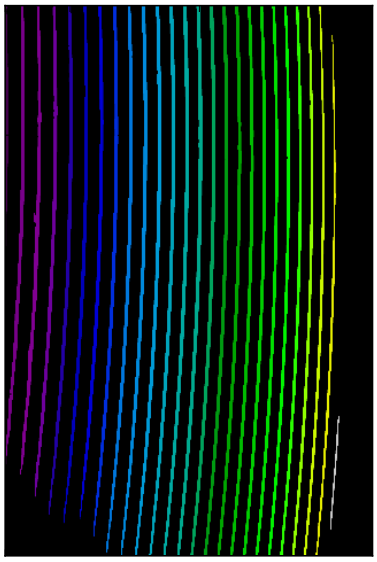

Figure 4: Example of a good region identification (distinct regions are detectable)

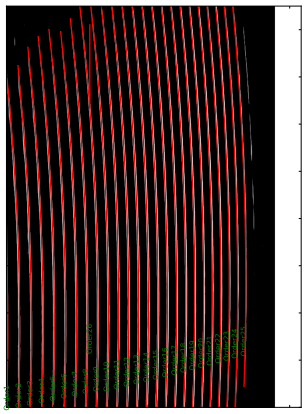

Figure 5: Fitted spectrum before order removal

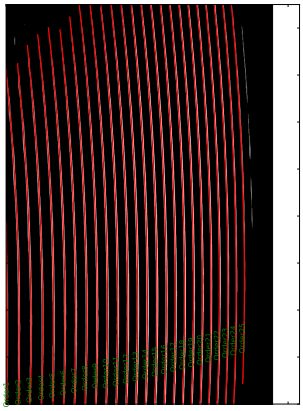

Figure 6: Fitted spectrum after order removal

5 Extracting from previously fitted image

Once you have run the code once you may choose to use the polynomial fits from a previous image (henceforth the old image) to fit a new image. This can be done by running the program Extract_Orders_from_polyfits.py.

This code essentially accesses parts of the Extract_Order.py code and just runs the last steps (see Section 4.3).

One needs to set the parameters for 'polypath' such that it links to the polyfit file created in the previous run (i.e. in './Savedfits/All_saved_polynomials.fits') with the old image and set the 'width' and 'files' variables as in Section 3. This can be done by editing the program or by modifying them on run time (again in Section 3).

The outputs will save in a new folder (defined by the name of the file) and no polyfits file will be saved.

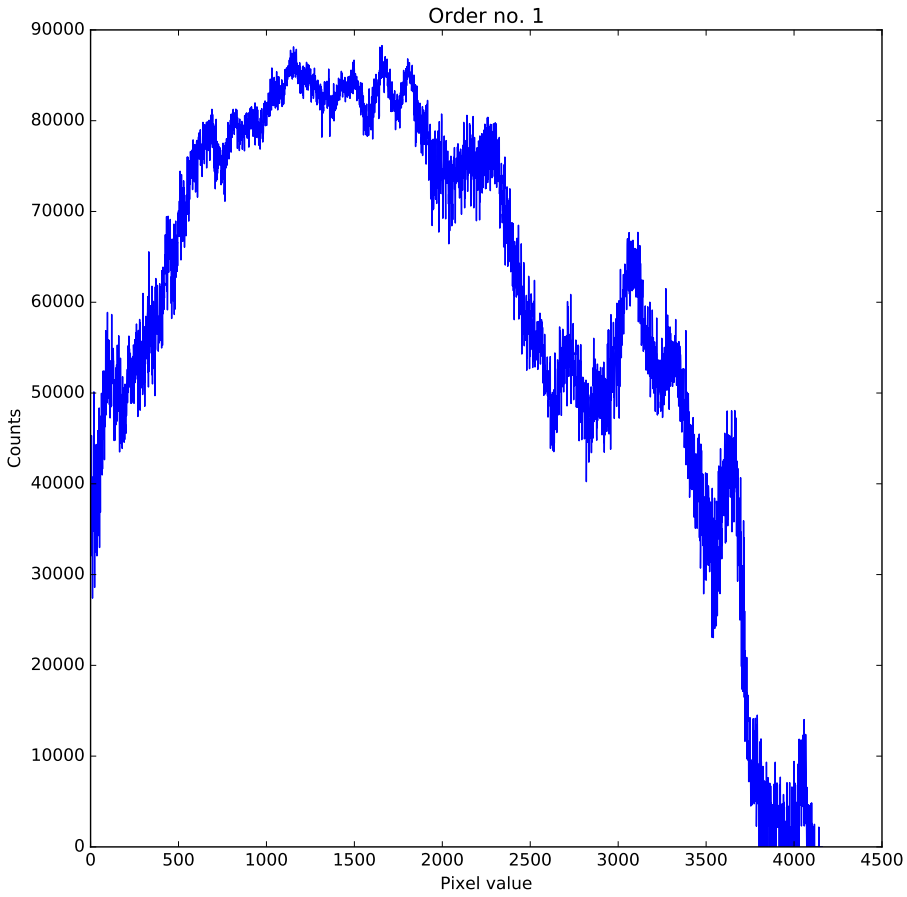

Figure 7: Example output spectrum of one order.