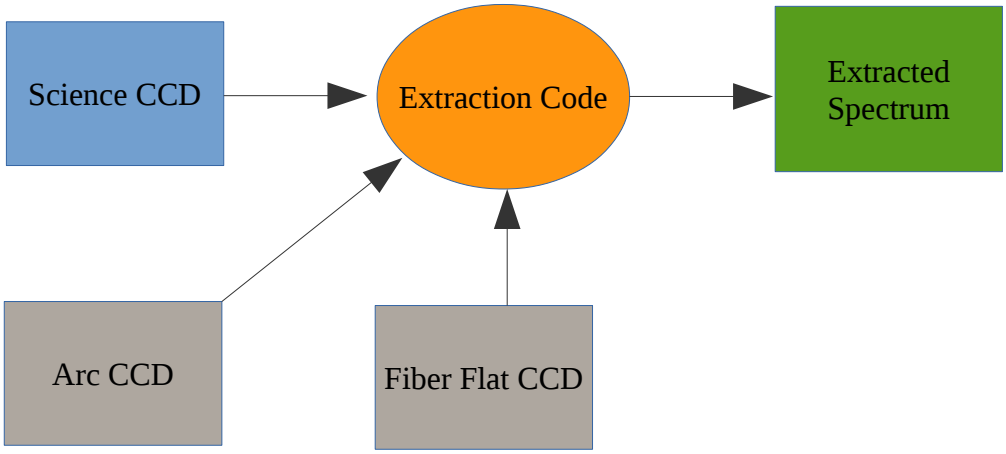
MINERVA Raw Reduction/Spectral Extraction Documentation (Optimal Extraction Version)

Purpose: Convert ccd frames to extracted spectra with wavelength solution (not currently normalized with flats or flux calibration)



Inputs:

Calibration Data

- Arc frames (two per telescope, one with, one without iodine)
- Fiber flats (one per telescope)
- Slit flats (not presently taken)

Science Data (All the same from point of view of code)

- Daytime Sky
- B-Stars
- Target Stars

Code:

- special.py – general functions (Gaussian profile, chi^2 fitting, etc.)
- minerva_utils.py – MINERVA specific functions, includes trace fitting and optimal extraction code.
- dark_flat_avg.py – Meant to find mean dark, bias, frames to feed other functions
- arc_calibrate.py – Finds the wavelength solution from the arc frames
- simple_opt_ex.py – Old code to perform optimal extraction (still in use, but will be deprecated)
- optimal_extract.py – Code to run optimal extraction. Calls heavily on functions in special and minerva_utils
- SP_extract.py – Code for testing 2D PSF and extraction
- bsplines.py – Generates functions related to splines in support of 2D PSF
- psf_utils.py – Special functions for fitting 2D PSF
- fit_slit.py – Imperfect code for flat fielding with slit flats. Not currently in use
- minervasim.py – Builds a model minerva ccd from kiwispec data, outdated, doesn't perfectly

match actual spectrograph
- modify_fits.py – Resaves fits files from some of the earliest files that were in a weird format. Not needed for recent data.

Outputs:

Extracted Spectrum - .fits file
Four HDU extensions
1. Photon Counts (proxy for flux)
2. Wavelength
3. Inverse Variance
4. Pixel Mask

Each extension contains a 3D array:
1. Axis 1 = pixel position (along trace)
2. Axis 2 = order
3. Axis 3 = telescope

A few notes. These are large files, ~7.3MB. Mask could be merged with inverse variance to save space. Also, wavelength is the solution from the arc frames. This is *identical* in all extracted spectra since (right now) I only have one arc frame. In the future, this may not be true, but space could be saved by making a wavelength solution fits and adding a header in each extracted spectrum to show which wavelength file should be used.

Installation

Code is on github under MinervaCollaboration/minerva-pipeline/spectral_extraction

Clone this onto the desired machine under $$$/minerva-pipeline/spectral_extraction (this will also clone $$$/minerva-pipeline/minerva_dopcode – the radial velocity pipeline)

To run, the following environmental variables must be set:
MINERVA_DIR – points toward $$$/minerva-pipeline/spectral_extraction/python
MINERVA_DATA_DIR – points toward wherever the data is stored (right now must have a local copy on this machine). Under this directory, data should be organized as:

nYYYYMMDD/nYYYYMMDD.exp_ID.exp_num[####].fits

Example:

n20160216/n20160216.HR2209.0017.fits
n20160216/n20160216.slitFlat.0009.fits

MINERVA_REDUX_DIR – points toward wherever you would like the reduced (extracted) data to be stored. Code will automatically save it with the same name, appended with a .proc.

Example:

n20160216/n20160216.HR2209.0017.proc.fits

MINERVA_SIM_DIR – points toward wherever you choose to store support files needed for extraction. These include arc lamplines, order estimates, iodine FTS spectrum, etc.

I like to set a MINERVA_ROOT, then put the other directories underneath:
MINERVA_ROOT/
 -data
 -sim
 -redux
 -software/master/minerva_pipeline/...

Load necessary sim files (I think I can just move these into github so they come automatically with the clone)

- table1.dat (HARPS vacuum reference for ThAr lines, with only wavelength range relevant to MINERVA)
- arc_order_estimates.csv (file I made manually with five estimated pixel position and corresponding wavelengths for the orders that show up on the MINERVA ccds. These get the automatic fitter close enough to lock on to the correct lines)
- [Optional] Iodine.fits (fits file of iodine FTS spectrum. I'm not using this right now (but I think the dopcode is))

Make sure calibration data is loaded. This includes:

1. n20160130/n20160130.fiberflat*.fits
2. n20160130/n20160130.thar*.fits
3. n20160115/n20160115.Bias*.fits
4. n20160115/n20160115.Dark*.fits
5. n20160115/n20160115.slitFlat*.fits

Note that I can (and probably should) change code to use a dynamic bias/dark basis rather than this fixed one.

Operation:

Generate calibration references (to be used in all subsequent extractions). Run the following commands (later I may change these to executables).

1. python $MINERVA_DIR/dark_flat_avg.py
2. python $MINERVA_DIR/arc_calibrate.py

These will make the files:

- $MINERVA_REDUX_DIR/bias_avg.fits
- $MINERVA_REDUX_DIR/dark_avg.fits
- $MINERVA_REDUX_DIR/slit_avg.fits
- $MINERVA_REDUX_DIR/wavelength_soln.fits

Set up crontab (or similar) to run daily:

1. Download data from minerva_main (unless already on minerva_main) to MINERVA_DATA_DIR
2. Run extraction(s). Depending on your system and preferences, these can be run in a batch, parallel, or similar to increase speed.
3. Upload reduced data from $MINERVA_REDUX_DIR to minerva_main

Right now I use the files (stored on Utah CHPC under the non-publicly accessible /uufs/chpc.utah.edu/common/home/bolton_data0/minerva/software/bin). I will post these somewhere

more accessible later, if they will be of wider use:

- load_minerva_data.sh
- extract
- launch_minerva_batch
- send_proc_data.sh

Details

(Note, all orientations described here are with reference to ccd frames with traces running horizontally, left to right in increasing wavelength. Bluest orders are on top, reddest on bottom)

Trace fitting

I use the high SNR fiber flats to find the traces. I start with a cross section of column 1 and find all of the peaks. I then discard a couple on the edges because they do not stay on the ccd for their entire length (and are outside of the range of the Iodine spectrum, so they wouldn't be used in the RV pipeline anyway). I then have a list of fiber numbers and their corresponding peak pixel location.

From there, I step by 20 pixels to the right, find the next peaks, and continue until the entire ccd is covered. At each step, I fit a modified gaussian profile (described in the extraction section) to the region around the peak. From this I get a more precise center as well as the height, width, and power parameters. From these positions, I run a quadratic fit (within each trace/fiber) for the peak vertical ('y') position as a function of the horizontal ('x') pixel position. The fit is repeated for the height, width, and power coefficients (essentially saving not only the trace position, but also the trace profile as a function of pixel).

The fiber flats appear insensitive to cosmic rays and other disruptions so this procedure tends to be robust. The coefficients are saved into $MINERVA_SIM_DIR/trace.fits.

Optimal Extraction

Here I follow the optimal extraction algorithm detailed in Horne, 1986. For the model, I use a modified gaussian profile as a function of pixel x:

$$I(x) = h \cdot \exp\left(\frac{x - xc}{2\sigma}^{p}\right) \quad (1)$$

Where the parameters are:

- h = height (relative intensity, function of sigma, plus photon counts)
- xc = center pixel position
- σ = width paramter of gaussian
- p = power parameter (2 for true gaussian)

This seemed to do an adequate job of matching the data, but it is well worth exploring other profiles.

Before starting an extraction, I re-fit the traces (since the position of the fiber flats and the position of the data are shifted by most of a pixel). I start with the estimates from the trace fitting, then repeat the procedure, just with science data. I fit this to a 6^{th} order polynomial for more precise centering.

In the extraction loop, I start with the first order and take a cross-section 5 pixels to either side

of the center (this is enough for good fitting, but small enough that it doesn't pick up much signal from neighboring traces. Right now, I can't handle overlapping traces, though this could, in priniciple, be done). This is the data reference and also serves to generate the inverse variance at each point, which I assume to be independent of adjacent pixels. Inverse variance is given by:

$$invar(x) = \frac{1}{data(x) + readnoise^2} \quad (2)$$

I then fit (nonlinearly, in a least-squares sense) a modified gaussian profile, using the mean, width, and power parameters from the trace fitting algorithm. With the fitted values, I reconstruct a model and find the relative height (pixel counts), and re-estimate the inverse variance.

Then I go through a loop to reject cosmic rays. This check the residuals between data and model and rejects any that are too large (and positive). There is some tuning required to set a sensible threshold. I found 25 counts to give a good result. If too many pixels are rejected for a particular model, the entire point is masked. Once the cosmic ray rejection is finished, final amplitude and inverse variance values are calculated.

This is repeated for every horizontal ('x') pixel along each of the 112 “good” traces (28 good fibers x 4 telescopes).

Final values are saved into an array as detailed above in **Outputs.**