Documentation for mmirs_pipeline_taskfile.py

Overview

The MMT and Magellan Infrared Spectrograph (MMIRS) has a data reduction pipeline, developed by Igor Chillingarian, that is written in the Interactive Data Language (IDL). That code, called mmirs-pipeline, is available here:

https://bitbucket.org/chil_sai/mmirs-pipeline

In order to reduce longslit spectroscopy and multi-object spectroscopy (MOS) data from MMIRS, a series of input files (called pipeline control task files) are needed for each individual exposure. Due to: (1) the difficulty of creating such files manually and the possibility for mistakes, and (2) the lack of experience with IDL by a larger number of MMIRS users, this standalone code has been developed to aid MMIRS users in reducing their data.

A stable beta release of a Python code call mmirs_pipeline_taskfile.py, created by Chun Ly, is now available here:

https://github.com/astrochun/MMTtools

Note: Improvements to the code is intended, specifically in algorithms to find the nearest telluric star.

The documentation for this code is provided below:

```
mmirs_pipeline_taskfile
====

Python code to create task files for MMIRS IDL pipeline:
http://tdc-www.harvard.edu/software/mmirs_pipeline.html

Operates in a given path containing raw files, find common files, and create
task files to execute with MMIRS IDL data reduction pipeline, mmirs-pipeline

This code will create several files:
(1) 'obs_summary.tbl'
     - A summary of observations in a specified 'rawdir' with FITS header
       information

(2) MMIRS input taskfiles
     - FITS header compliant ASCII files that are provided to the
       MMIRS IDL pipeline

(3) 'IDL_input_[name].lis'
     - A list of files to run the MMIRS IDL pre-processing (i.e., non-linearity
      correction)

   Here, [name] is a naming convention that contains the name of the
   target, LS or MOS mode, and grism and filter combinations. It is defined
   from mmirs_pipeline_taskfile.
```

TO EXECUTE:

0. First, you will need python. This code has been tested to work with v2.7.15 and 3.6.8.
   I recommend installing through anaconda:
     https://www.anaconda.com/download/

   Next you will need the Astropy package. This code has been tested to work with v1.3, v2.0.2, v3.1.2 of Astropy. Install via the conda command:
     conda install astropy

   In addition, you will need astroquery package. Install via pip command:
     pip install astroquery

1. After python, astropy, and astroquery are successfully installed, you need to clone Chun Ly's MMTtools package to your local machine:
     git clone https://github.com/astrochun/MMTtools.git

2. Within ipython (or python) import this code:
     from MMTtools import mmirs_pipeline_taskfile

   Note: If this does not work, MMTtools is not within your PYTHONPATH environment

3. Remove FITS files from your raw path that you do not want this code to detect. This code does a file search for '*.????.fits*'. If there are bad FITS files or those that are saturated, relocate them to another folder or delete them.

4. Call code for specified path in ipython (or python):
     rawdir = '/path/to/raw/files/' <- Should end with forward slash
     mmirs_pipeline_taskfile.create(rawdir, w_dir='', dither='ABApBp',
                                    bright=True, inter=True)

   Notes:
    1. w_dir can be changed. Default is to create a 'reduced' folder in [rawdir]
    2. dither: If NOT specified, code will determine dither pattern based on FITS header
    3. Set bright to True if there is a bright object in slit
    4. Set inter to True if multiple calibration datasets (including telluric star) are available. The code will prompt user to select.
    4. Note that mmirs-pipeline will look for a 'calib_MMIRS' folder. This is needed in the pre-processing (e.g., applying non-linearity correction). This Python code will prompt you to provide a path such that a symbolic link is created.
       For example, if mmirs-pipeline is installed in /codes/idl, then you would type '/codes/idl', and it would create a symbolic link as follow:
         ln -s /codes/idl/mmirs-pipeline/pipeline/calib_MMIRS [rawdir]/calib_MMIRS

   If your [rawdir] contains multiple targets, mmirs_pipeline_taskfile _should_ separate out the targets in a respective manner.

5. Next install IDL and set-up it up to have an appropriate license.
   Then have Igor Chilingarian's mmirs-pipeline on your computer and included in your IDL_PATH environment:

```
  git clone https://bitbucket.org/chil_sai/mmirs-pipeline

Also make sure that you have the IDL Astrolib installed and it is in your
IDL_PATH environment:
  git clone https://github.com/wlandsman/IDLAstro

Note: legend.pro has been deprecated (part of IDL as al_legend.pro),
which will cause mmirs-pipeline to crash. Either change mmirs-pipeline to
use al_legend or include this legend.pro somewhere in your IDL_PATH:
  https://idlastro.gsfc.nasa.gov/ftp/obsolete/legend.pro


6. Run the IDL script run_mmirs_pipeline_nonlin_script.idl that is
   automatically generated from step 3 in the [rawdir] path:
     idl run_mmirs_pipeline_nonlin_script.idl

   Note: All the pre-processed files will be placed in the 'preproc' folder
   within [rawdir].


7. After creating pre-processed files, you can now run the MMIRS pipeline via
   the IDL scripts (run_mmirs_pipeline_[name].idl) that are automatically
   generated from step 3 in the [rawdir] path

   If your [rawdir] contains multiple targets, mmirs_pipeline_taskfile _should_
   separate out the targets in a respective manner.  Thus, there should be
   multiple run_mmirs_pipeline.idl scripts



TIPS:
   This code has a log file that is created in rawdir called
   'mmirs_pipeline_taskfile.log'. It logs everything that is written to
   stdout

   If a bug is encountered please submit an issue ticket here:
     https://github.com/astrochun/MMTtools/issues

   Also, please email the creator, Chun Ly, at astrochun [at] gmail.com
   your mmirs_pipeline_taskfile.log, any error messages on the ipython
   or python screen, and your 'obs_summary.tbl' file
```