Documentation of the HSRL python data processing code

Brad Schoenrock

January 27, 2017

The purpose of this document is to aid in my (and others) understanding of the inner workings of the HSRL data processing taking place in the practical operation of the HSRL by NCAR. The goal of this work is to be able to maintain the software which is performing this data processing while replacing it with C++ processing for efficiency and maintainability.

1 Introduction

The High Spectral Resolution Lidar (HSRL) is a system that can be deployed on the National Science Foundation/NCAR HIAPER Gulfstream V (GV) aircraft or in a customized shipping container to study atmospheric phenomena around the world. The Gulfstream V High Spectral Resolution Lidar (G5-HSRL) is an eye-safe calibrated lidar system that can measure back scatter cross section, extinction and depolarization properties of atmospheric aerosols and clouds [**?**].

The HSRL was originally designed and built by the University of Wisconsin Lidar Group. This group also provided some data processing code in MATLAB, and later switched to python to suit its needs. One function of the UW processing is that it provides real time and archived data which can be viewed at the GVHSRL in Madison WI website. Scientists have expressed interest in keeping this functionality and on improving the error estimation on the extinction as a measure of data quality.

2 Python

To begin there is a wiki describing how to set up the python environment nessicary for the execution of the python processing code [**?**].

The HSRL_Python package can be checked out from the NCAR GitHub here:
https://github.com/NCAR/hsrl_python

Also of interest are the following repositories:
https://github.com/NCAR/hsrl_dpl_tools
https://github.com/NCAR/hsrl_configuration
https://github.com/NCAR/hsrl_instrument

I have a personal copy with some edits here aimed at probing for understanding:
https://github.com/BradSchoenrock/HSRL_python_Brad

2.1 the data processing

The data processing begins with some control scripts hsrl_dq and cset-cfradial.py which can be used to generate plots and create some Pseudo CFRadial files respectivly. These call functions in the python codebase which start the processing from Raw NetCDF files which are located on /scr/eldora1/HSRL_data. The first functions called are in maestro/rti_maestro.py.

2.1.1 write_netcdf

This function sanatizes some inputs, and acts as a switch for different lidar setups. Ultimitly it calls another function called makeNewNCOutputFor and pasess it the format of files and directory locations.

2.1.2 makeNewNCOutputFor

This function takes in a framestream (which is what?) as well as format and file location for output.

There are some try except statements in here which are not clear why they are needed.

This function has some strange lines of python which confuse me.

Line 1735: if True:#not cfradial: then it does stuff. Why the if true statement?

Line 1753: loop which does nothing in it (just a pass), but seems to be doing all the writing. The resulting CFRad file has no actual data in it if this is commented out. In the for loop it is itterating over an instance of the class dpl_netcdf_artist which is found in lg_dpl_toolbox/dpl/dpl_artists.py.

This is the line for reference.

for f in artists.dpl_netcdf_artist(framestream,template,outputfilename=filename,output=output,fo
pass

2.1.3 dpl_netcdf_artist

3 HSRL2Radx

4 Hawkeye

5 Commands

Raw NetCDF files to Pseudo CFRadial files
Python code

```
./scripts/csetcfradial.py
```

Pseudo CFRadial files to CFradial
Hsrl2Radx

```
/h/eol/brads/git/hsrl_configuration/projDir/ingest/params
Hsrl2Radx -params Hsrl2Radx.test -debug -f /tmp/gvhsrl_20150729T1930_20150729T2000.nc
```

CFRadial to HawkEye display

```
/h/eol/brads/test/display/params
HawkEye -params HawkEye.hsrl-brad
```

See the contents of a NETCDF file

```
ncdump gvhsrl_20150729T1930_20150729T2000.nc — less
```