LDOPE Python Re-projection Tool (V1.0)

LDOPE/GSFC/NASA 06/04/2020.

1. Introduction

NASA Land Data Operational Product Evaluation (LDOPE) group has been using different projection tools that have been developed in-house, to re-project NASA's MODIS and VIIRS Land data products in both HDF4 and HDF5/NetCDF4 data formats. These re-projection tools, developed in C, can handle all MODIS and VIIRS Land data products, including L3/L4 gridded Sinusoidal 10-degree tiled data products as well as swath based L1B and L2 products. These tools are now being distributed to the user community as part of this package, with the addition of a Python based simple wrapper interface to make it easy for general users to invoke and run these core C-based re-projection utilities. The users are strongly encouraged to use this Python wrapper (*projbrowse.py*) to call the LDOPE re-projection utilities as it hides a lot of internal details of the core utilities and handles some of the intermediate processing of the data products that may be necessary to make them ready for re-projection. Specifically, the current version of the LDOPE core re-projection utilities can only handle input files in HDF4 format. Hence, with support from other LDOPE QA tools, all HDF5/NetCDF4 are first converted to HDF4 before invoking the core tools. In addition, a few low-level swath data products have no geolocation data fields (Latitude and Longitude) associated with them and hence, the two geolocation data fields may have to be first copied from the corresponding geolocation data products to the swath data file before projection. For users, who are not familiar with the LDOPE QA Tools and MODIS and VIIRS Land data products, directly invoking the LDOPE reprojection tools and associated LDOPE QA Tools can be challenging and the "*projbrowse.py*" Python tool is intended to facilitate users in this regard.

In general, the "*projbrowse.py*" Python wrapper will perform all the required pre-processing such as file format conversion, copying geolocation to swath file, checking for all necessary inputs etc. In essence, the Python package free users from learning the details of the actual reprojection utilities and helps them to quickly re-project their data products of interest, using the most commonly used projection parameters and options.

This document has two subsections below: Section 2 deals with the "Installation of the Package" and Section 3 illustrates how to use *"projbrowse.py"* to project different MODIS/VIIRS land data products.

2. Installation of the Package

Users are required to have Python3.5+ and the "numpy" module installed. The "*projbrowse.py*" is also dependent on the set of LDOPE QA Tools, therefore, users should have the LDOPE QA Tools installed before using this Python package. The steps needed for installing the LDOPE QA Tools have been outlined below. All the dependent tools require HDF4/HDF5 and HDFEOS2/HDFEOS5 libraries and all the tools have been compiled and tested successfully with HDF4 V4.2.10 and HDF5 V1.8.12 on a Linux (Centos 7.4) Operating System.

Step 1, Installation of LDOPE QA Tools V3.6

Users may download LDOPE QA Tools V3.6 from NASA's LAADS DAAC at (https://ladsweb.modaps.eosdis.nasa.gov/tools-and-services/#ldope). Please follow the instructions given in the user's guide for the LDOPE QA Tools (Appendix A: Installation Instructions on page 88) for installing the LDOPE QA Tools.

After LDOPE QA Tools V3.6 is installed, users need to update their personal shell script file (suppose ~/.personal is the environmental shell file) by adding the below line to the file.

```
export PATH=$PATH:<path_QA_tools>/LDOPE_tools_V3.6/bin
```

Then source the personal shell script

```
source ~/.personal
```

With this, users can invoke the LDOPE QA Tools from any working directory. Once installed, please run the following three commands to check if the three needed LDOPE QA Tools are successfully installed and setup.

```
read_meta -help
```

```
read_sds_attributes -help
```

```
copy_sds -help
```

Step 2, Installation of LDOPE core projection utilities

Decompress the tar.gz file using the command below

```
tar -zxvf projbrowse_pkg.tar.gz
```

Users can see that there are five directories under the root path "projbrowse_pkg" and they are: "ANCILLARY", "bin", "obj", "proj_lib", and "src".

First enter the directory "proj_lib", and run the command below:

```
make linux
```

A file named *libproj.a* should be created, implying that projection library has been successfully installed.

Then change directory to the package root directory and "cd" to the directory "src", and run the command below:

```
make
```

Note that if SZlib 2.1 is not installed for HDF libraries in the step 1 above, user may need to remove -lsz in the Makefile.

Users can ignore the lines of warning, while the compilation is going on, as long as there is no error in compilation.

Change directory again to the package root directory and "cd" to the directory "bin", user should see three files there, core C projection binaries: *projbrowse* and *projbrowse_NPP*, and the Python wrapper interface, *projbrowse.py*

projbrowse and *projbrowse_NPP* are created by the above make command, whereas "*projbrowse.py*" is the Python package. To invoke *projbrowse*, *projbrowse_NPP*, and *projbrowse.py* from any working location, users should setup environment variables as detailed below.

Step 3, setup environment variables to make projbrowse.py work

Under the "bin" directory, first make *projbrowse.py* an executable file by running

```
chmod 755 projbrowse.py
```

Users should update their personal shell script file (suppose ~/.personal) by adding the below two lines to the file.

```
export PATH=$PATH:<path_projbrowse_pkg>/projbrowse_pkg/bin
export ANCPATH=<path_projbrowse_pkg>/projbrowse_pkg/ANCILLARY
```

Users may want to run the following three commands just to make sure that all LDOPE projection utilities are successfully installed and setup and if the Python package "*projbrowse.py*" works

```
projbrowse -help
```

```
projbrowse_NPP -help
```

```
projbrowse.py -help
```

3. Use projbrowse.py to Project MODIS/VIIRS Land Data Products

User can run the below command to get the help information

```
projbrowse.py -help
```

The synopsis of the *projbrowse.py* is below

projbrowse.py -if=input_HDFs -sds=SDSs_name -of=output_HDF [-proj=projection_code]
[-pixsz=pixsz_in_KM] [-geoif=geolocation_files] [-tmp_path=temp_dir]
[-DN=1,...,3] [-vector=1] [-box=startcol,startrow,width,height]
[-fill=specified_fill_value] [-global]

The order of the above arguments is not static but variable.

3.1. Required Arguments

It requires at least three arguments to invoke the re-projection process, including the input files, "-if=", Scientific DataSet (SDS) names "-sds=", and output HDF4 file name "-of=". If users don't know the SDS_names in the input files, they may simply run *projbrowse.py* by using just the input files' argument "-if=", and the package will screen-print all valid SDS names which can be used against the argument "-sds=". Or users can use free software tools such as ncdump or hdfview, to check the contents of any HDF files.

For "-if=" argument, the input can be a single file or multiple files separated by comma, or shell-style wildcards, or a directory containing all the input files. If there are spaces between the file names, uses should use double quotes to enclose the entire argument like "-if=hdf_1, hdf_2, etc.".

For "-sds=", users can input a single SDS name or multiple names separated by comma. If there are spaces in the SDS name or names, uses should use double quotes to enclose the entire argument like "-sds=SDS_name_1, SDS_name_2, etc.".

For "-of=", users can specify ONE output HDF4 file name to store re-projected data from single or multiple input files. Current version doesn't support HDF5 or NetCDF4 output.

Note that after the tool finishes the reprojection processes, the screen will print out many lines of information with some words like "Cannot" or "unable". Users may ignore these information as long as the tool creates the specified output file.

Below is an example to re-project 4 tiles of S-NPP C1 VNP15A2H, produced by NASA LandSIPS, for the 8-day period of 2015193.

```
projbrowse.py -if=VNP15A2H.A2015193.h09v04.001.2018152101753.h5,VNP15A2H.A2015193.h09v05.001.2018152102058.h5,VNP15A2H.A2015193.h10v04.001.2018152101753.h5,VNP15A2H.A2015193.h10v05.001.2018152102101.h5 -of=out_VNP15A2H.A2015193.hdf -sds=Fpar,Lai
```

For 3-dimensional SDSs, users must specify which layer will be re-projected by specifying the layer number appended to the SDS name like "sdsname.n" where "n" is 1-based layer number. For example, to re-project all the layers for Band 1 BRDF parameters from two MODIS C6 MCD43A1 tiles, follow the example below:

```
projbrowse.py -if=MCD43A1.A2009111.h10v05.006.2016125000150.hdf,MCD43A1.A2009111.h09v05.006.2016124233116.hdf -of=out_MCD43A1.hdf -sds=BRDF_Albedo_Parameters_Band1.1,BRDF_Albedo_Parameters_Band1.2,BRDF_Albedo_Parameters_Band1.3
```

3.2. Optional Arguments

The optional arguments are in the brackets as listed below (Note that do not use the brackets when using any optional arguments)

[-proj=projection_code] [-pixsz=pixsz_in_KM] [-geoif=geolocation_files] [-tmp_path=temp_dir] [-DN=1,...,3] [-vector=1] [-box=startcol,startrow,width,height] [-fill=specified_fill_value] [-global]

Users can get the brief descriptions of each optional arguments by running below command

```
projbrowse.py -help
```

Below give more details of each optional arguments

"-proj=" is for output map projection code. Though there are 22 map projections listed, only the Hammer-Aitoff, Lambert-Azimuthal and Geographic projections have been tested thoroughly at LDOPE while other projections should work but they have not been fully tested. The default output map projection is Hammer-Aitoff.

"-pixsz=" is for the output pixel size (pixsz) or spatial resolution in Kilometers (km). If users don't specify pixsz, the default value would be inferred by the Python package based on HDFEOS grid level information for a L3/L4 tiled product or based on scan level information for a swath product.

"-geoif=" is the corresponding geolocation files for the swath data without geolocation data fields: Latitude and Longitude. If there are spaces between the file names, uses should use double quotes to enclose the entire argument like "-geoif=geo_HDF _1, geo_HDF_2, etc.".

"-tmp_path=" is for users to specify a temporary path to store intermediate files if they don't want to store all the intermediate results in their current working directory. The default path is current working folder (cwd), cwd/temp_dir which "*projbrowse.py*" will automatically create it if it doesn't exist. After reprojection, the temporary path including all the created intermediate files will be automatically removed by the package.

"-DN=" can be used to filter data from day, night, or mixed. For example, for VIIRS night light, users may just want to consider nighttime granules and not daytime, and DN=2 can be used in this case.

Below example shows how the above optional arguments are used to project two granule VNP02DNB night light observations which need geolocation data fields from VNP03DNB.

```
projbrowse.py -proj=22 -
if=VNP02DNB.A2019200.0606.001.2019200125817.nc,VNP02DNB.A2019200.0612.001.20192
00125647.nc -
geoif=VNP03DNB.A2019200.0606.001.2019200124844.nc,VNP03DNB.A2019200.0612.001.20
19200124614.nc -of=out_VNP02DNB.A2019200.hdf -sds=DNB_observations -DN=2 -
tmp_path=./temp_VNP02DNB -pixsz=0.75
```

"-vector=1" will overlay coastline on top of the projected data.
"-fill=" is for users to specify a different fill value than the one mentioned as part of the SDS attributes in the HDF file. Specified fill value can be just one for all SDSs or different values separated by comma for different SDSs.

"-global" is for users to specify if the output projected domain should be extended to cover the whole global.

3.3. Conversion to GeoTIFF file format

The output HDF can be converted to Geotiff, a standard GIS file format. In the last example given above, -proj=22 refers to Geographic map projection. You can find descriptions of all the supported map projections by invoking the projection command with "-help" argument. In LDOPE, the projection types of 16 (Hammer-Aitoff) and 22 (Geographic) are extensively used and while other map projections should also work, we cannot guarantee their correctness and users are encouraged to check them for accuracy by importing them in some GIS tools etc.

The projected output HDF file, as in our example above, out_VNP02DNB.A2019200.hdf, can be imported and opened in any GIS or Image processing tools that can handle HDF4 files. The projection information may need to be manually defined in that tool. The projection information is provided as part of the output projected dataset. Users may however find it easier to convert the projected HDF4 file to a GIS friendly format like GeoTIFF, using any open source tool package, like GDAL. One such example is provided below for our example output file

```
gdal_translate -a_srs EPSG:4326 -co TFW=YES -a_ullr -90.799904  24.062447 -52.259520 -
21.512873 out_VNP02DNB.A2019200.hdf out_VNP02DNB.A2019200.tif
```

this will create a GeoTIFF file, out_VNP02DNB.A2019200.tif, with a "*.tfw*" world file. You can get the upper-left (UL) and lower-right (LR) extents by doing a ncdump on the projected file or opening it in HDFView.

Note that trying to project too many input files, over the entire globe and at finer native resolutions of 250m or 500m (MODIS) or 375m (VIIRS) could sometime cause memory issue, given the large size of the output global projected datasets and may lead to the failure of projection. We suggest that to avoid such issues, users split the input files into smaller subsets and re-project them separately and then use other tools such as GDAL to merge the different subsets for any pixsz finer than 1km. In addition, because current version projbrowse.py supports HDF4 not HDF5, thus, the output data file size should not exceed 2G limits of HDF4.