National Aeronautics and Space Administration Goddard Earth Science Data Information and Services Center (GES DISC)

README Document for

A Multi-Sensor Water Vapor Climate Data Record Using Cloud Classification: AIRS-MODIS Matchup Indices

Goddard Earth Sciences Data and Information Services Center (GES DISC)
http://disc.gsfc.nasa.gov
NASA Goddard Space Flight Center
Code 610.2
Greenbelt, MD 20771 USA

Last Revised Jun 8, 2017

Prepared By:

Gerald Manipon

Name
GES DISC
GSFC Code 610.2
6/17/2017

Date

Eric Fetzer, JPL

Name

Reviewed By:

Reviewer Name
GES DISC
GSFC Code 613.2

Date

Revision History

Revision Date	*Changes*	*Author*
2017-06-08	Added initial content.	Gerald Manipon
2017-06-17	Editorial comments.	Eric Fetzer
2017-07-06	Collocation comments	Mathias Schreier
2017-09-18	Added python example usage	Gerald Manipon

Table of Contents

1.0 Introduction

This document provides basic information for using the WVCC (Water Vapor Cloud Climatology) AIRS-MODIS matchup index dataset.

The WVCC AIRS-MODIS matchup index dataset consists of products generated for the focus on multi-sensor water vapor climatology using cloud classification.

1.1 Dataset/Mission Instrument Description

The basic task is to bring together retrievals of temperature, water vapor and cloud properties from multiple "A-train" instruments, classify each "scene" (instrument field of view) using the cloud information, and develop a merged, multi-sensor climatology of atmospheric temperature and water vapor as a function of altitude, stratified by the cloud classes. This is a large science analysis project that will require the use of the HySDS (Hybrid Cloud Science Data System) and SciFlo technologies to discover and organize all of the datasets, move and cache datasets as required, find space/time "matchups" between pairs of instruments, and scale up processing of years of satellite data to produce the climate data records.

Figure 1.1: The A-train

1.1.1 AIRS & AMSU

AIRS is a continuously operating cross-track scanning sounder, consisting of a telescope that feeds an echelle spectrometer. The AIRS infrared spectrometer acquires 2378 spectral samples at resolutions, $\lambda/\Delta\lambda$, ranging from 1086 to 1570, in three bands: 3.74 to 4.61 µm, 6.20 to 8.22 µm, and 8.8 to 15.4 µm. The angular extent of the infrared channels is 1.1° in diameter, which corresponds to a horizontal footprint of about 15x15 km in the nadir.

AMSU-A is a 15-channel microwave temperature sounder implemented as two independently operated modules. Module 1 (AMSU-A1) has 12 channels in the 50-58 GHz oxygen absorption band which provide the primary temperature sounding capabilities and 1 channel at 89 GHz which provides surface and moisture information. Module 2 (AMSU-A2) has 2 channels: one at 23.8 GHz and one at 31.4 GHz which provide surface and moisture information (total precipitable water and cloud liquid water). Like AIRS, AMSU-A is a cross-track scanner. AMSU-A scans once per 8 seconds, 1/3 the rate of AIRS, and its footprints are approximately three times as large as those of AIRS (45 km at nadir). This results in three AIRS scans per AMSU-A scans and nine AIRS footprints per AMSU-A footprint. Each AIRS/AMSU granule includes six minutes of scanning (240 per day), and contains 30x45 AMSU and 90x135 AIRS footprints alongtrack x crosstrack.

AIRS and AMSU are on the Aqua spacecraft, in a sun-synchronous orbit with local equator crossing times of 1:30 PM (northward moving orbit) and 1:30 AM.

1.1.2 MODIS

The moderate-resolution imaging spectroradiometer (MODIS) is a payload scientific instrument built by Santa Barbara Remote Sensing that was launched into Earth orbit by NASA in 1999 on board the Terra (EOS AM) Satellite, and in 2002 on board the Aqua (EOS PM) satellite. The instruments capture data in 36 spectral bands ranging in wavelength from 0.4 µm to 14.4 µm and at varying spatial resolutions (2 bands at 250 m, 5 bands at 500 m and 29 bands at 1 km). Together the instruments image the entire Earth every 1 to 2 days. They are designed to provide measurements in large-scale global dynamics including changes in Earth's cloud cover, radiation budget and processes occurring in the oceans, on land, and in the lower atmosphere. The swaths of Aqua MODIS are slightly wider than AIRS/AMSU swaths, so all AIRS/AMSU fields of view are completely covered by Aqua MODIS observations.

1.1.3 SciFlo

SciFlo is a semantically-enabled ("smart") Grid Workflow system that ties together a peer-to-peer network of computers into an efficient engine for distributed computation. The SciFlo workflow engine enables scientists to do multi-instrument Earth Science by assembling remotely-invokable Web Services (SOAP or http GET URLs), native executables, command-line scripts, and Python codes into a distributed computing flow. SciFlo also deploys a variety of Data Grid services to: query datasets in space and time, locate & retrieve on-line data granules, and provide on-the-fly variable and spatial subsetting.

1.2 Algorithm Background

Production of the AIRS-MODIS matchup index dataset is divided into two workflows producing the following final product:

1) Footprint Matchups to HDF4
2) HDF4 to NetCDF4 conversion and metadata extraction

1.2.1 Footprint Matchup

The collocation is based on normal AIRS resolution (90 scans per line) and 1km MODIS resolution (1354 scans per line). In the following, *column* refers to the scan angle and *row* to the scan line number.

The matchups are created in a way to be storage efficient and avoid a possible overlapping of footprints. It used the fact, that the footprint coverage of AIRS with respect to the underlying MODIS pixels is only dependent on the scan angle – means row number. The geolocation and the footprints can therefore be separated into two sets of data and merged together during data processing:

- They first set of data provides the coordinates (means: row-number and column-number) of the nearest MODIS pixel to the center of the AIRS footprint. It is created aon a 1x1km pixel size. If you are using a 5x5km MODIS product, then just divide the indices by 5.
- The second set of data provides the row-dependent footprint sizes and gives every MODIS pixel around the center pixel from the first set an index and a weighting with respect to the fooprint shape. It is again on a 1x1 km footprint size. For 5x5 km

products, the arrays just have to be averaged by a factor of 5. We provide 4 different options for these footprints, based on pro-launch measurements and calculations after wards. The most reasonable is the “smeared” function, which includes a theoretical movement of the instrument. We are referring to Schreier et al., 2010 (JAOT) for further explanations.

There are therefore several possibilities to average MODIS cloud products or radiances within an AIRS footprint:

- Use your own footprint estimate (e.g. a simple circle) around the center pixel from the set 1 and average the MODIS pixel within the circle. In this case, there is no need to bother about the weighting functions from the second set.
- Use one of the row dependent spatial response functions from the second set to select and weight the remaining MODIS pixels, which are surrounding the center pixel. The functions just get overlaid over the given center points and help to identify the remaining pixels of interest.

The slides in the appendix of this section will explain in a more graphic way, how the process of merging set 1 and set 2 are supposed to happen.

However, the length of the files in set 1 has to be discussed, before the figures below become clearer. We saved our collocation data in 30 min intervals. The reason becomes obvious if you look at the different ways of the instruments of saving data with respect to time:

- AIRS has 6 minute granule, MODIS 5 minute granules
- AIRS files have always 135 rows, MODIS rows can vary between 2030 and 2040 lines. They do this to compensate for time gaps.

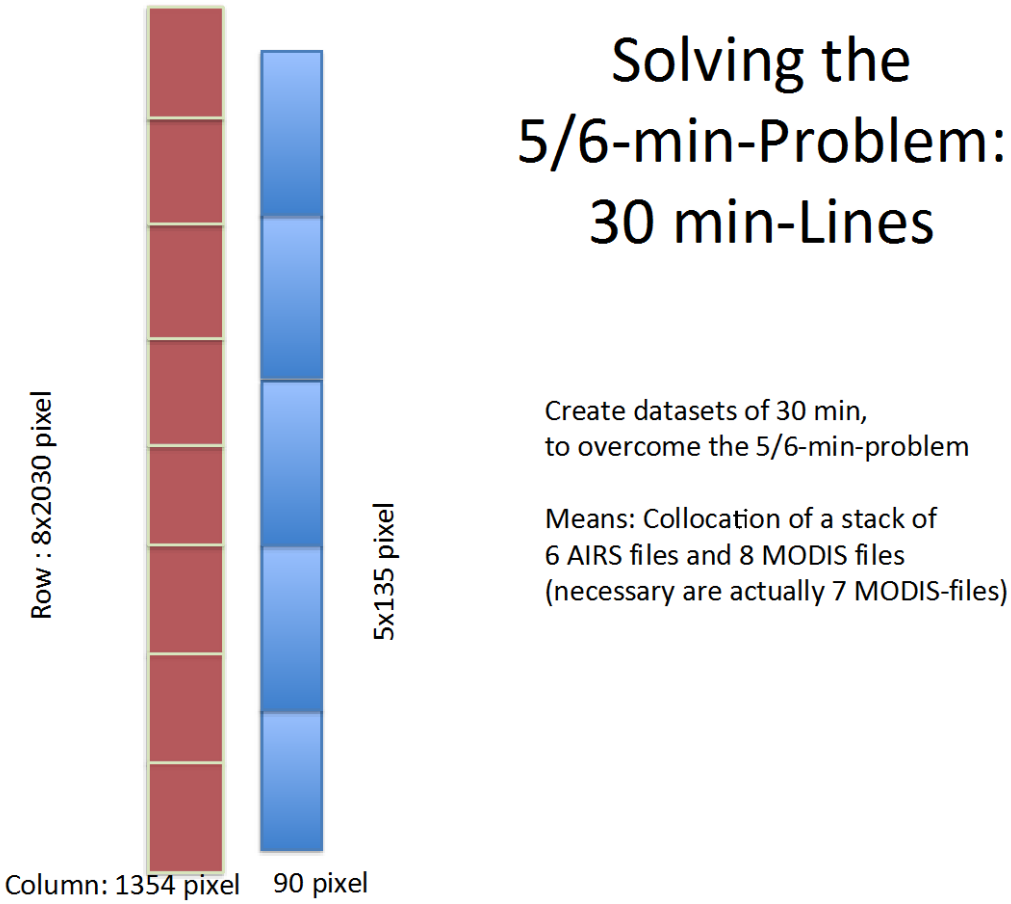
To overcome this problem, our collocation data (set 1) is saved on a 30 min range per file (the least common denominator of 5 and 6 min) and correspond to “stacked” AIRS and MODIS data of roughly 30 min. This ensures that all AIRS and MODIS pixels within thee given 30 min are covered and there is no need to worry if a MODIS files has 2030 or 2040 lines.

So, with respect to AIRS, a collocation file of 30 minutes covers 5 “stacked” AIRS files. If HH is the time of the hour and the time of the collocation file is given by HH00, then it covers AIRS granules HH0 to HH4. A collocation file with time HH30 covers AIRS granules HH5 to HH9. A collocation file has therefore 90 columns and 675 rows, which corresponds to 5x135 rows.

Every point saves the index of a corresponding MODIS column and row and again of a stacked set of MODIS files. With respect to these MODIS indices, the 30 min range is a little bit more complicated: We decided to use more than 6 MODIS files to cover the respective 30 min range

and add additional two files. This causes some redundancy, but ensures that every collocation pixel is covered with an index. So, the range of the indices for MODIS files for a 30 min collocation file covers actually 8 (6+2) “stacked” files of 5 min MODIS data: a collocation file with time HH00 covers MODIS files from HH00 to HH35, a collocation file with time HH30 covers MODIS files from HH30 to (HH+1)05.

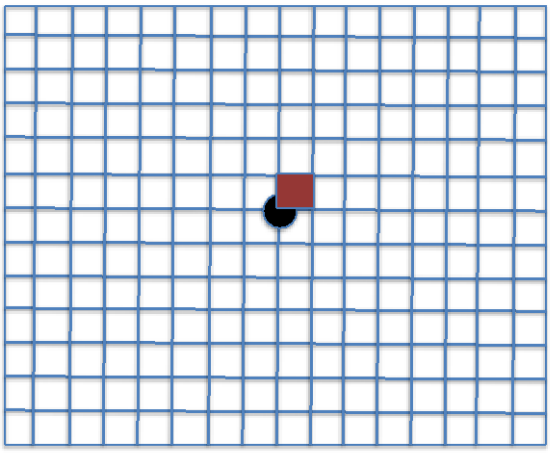
The scheme is better understandable in a visual way, especially because the overlay of footprints and the “stacked” datasets become much clearer, when visualized. Attached is therefore a presentation to explain the datasets and handling:



Step I: Find nearest point

Look for given AIRS-Lat/Lon
For every AIRS-point

Find nearest MODIS-Lat/Lon
in the dataset



Solving the data amount problem: Just save this center collocation

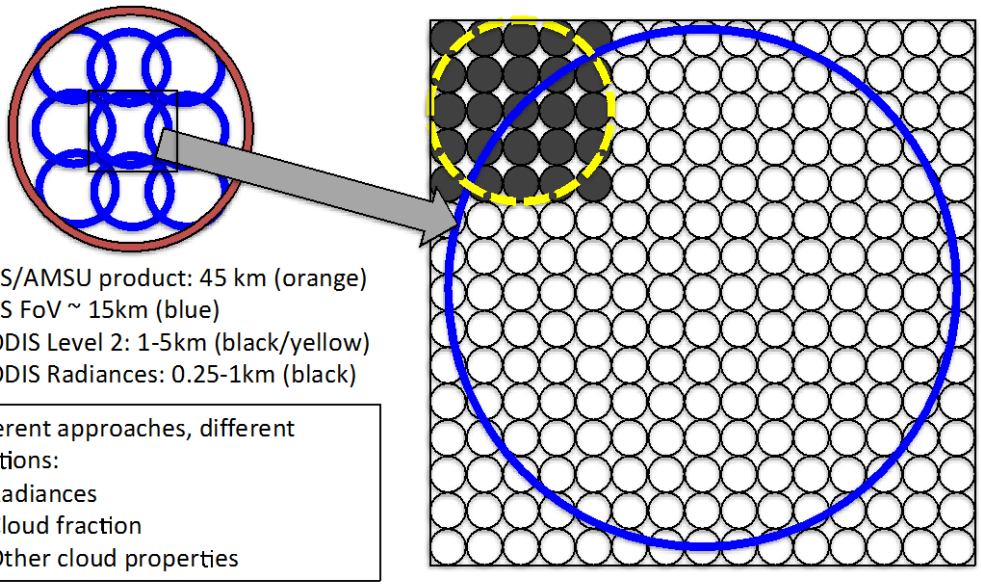
The resulting 30min-geolocation-file includes only 2 arrays of 675x90 points:

- An array of 675x90 pixels, which give you the column-index of the nearest MODIS-pixel
- Same array for row-index

- Example: if you want to know the collocation of AIRS-pixel(34/45) then row_array(35/45) gives you a number for the MODIS-row and column_array(34/45) gives you the column-number

- Additional information (Used modis-files and AIRS files) are saved as attributes to restore information

Sub-Pixel Information depending on problem



AIRS/AMSU product: 45 km (orange)
AIRS FoV ~ 15km (blue)
MODIS Level 2: 1-5km (black/yellow)
MODIS Radiances: 0.25-1km (black)

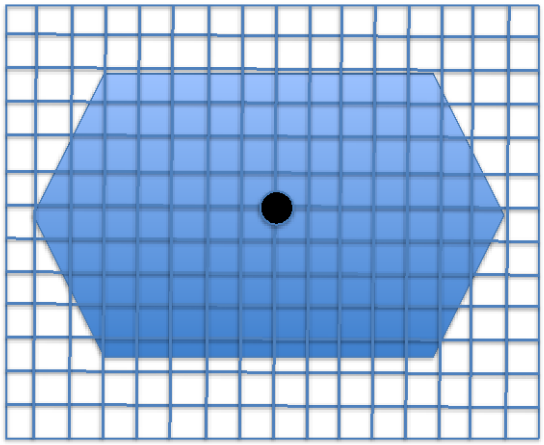
Different approaches, different solutions:

- Radiances
- Cloud fraction
- Other cloud properties

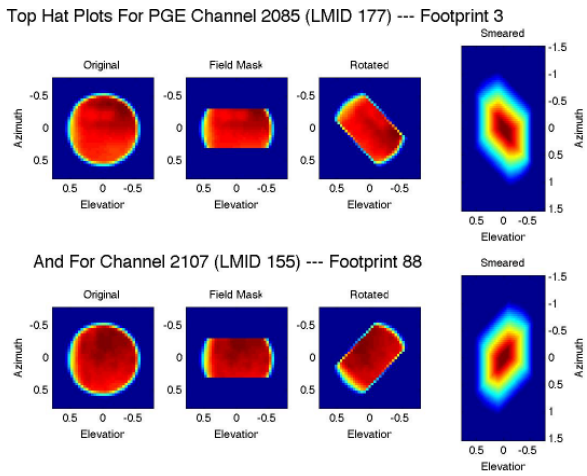
Step II: All you have to do now, is to overlay a spatial response function

Take a given spatial response function and use it to estimate the "participation" of surrounding MODIS pixel

However ....



Spatial response function is "Problem"-dependent !

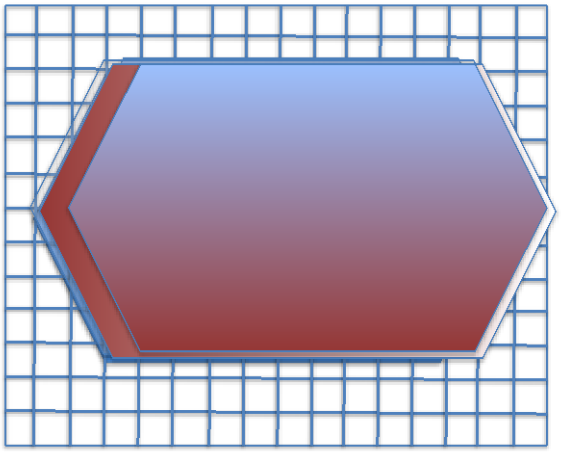


4 types
90 scan angles
2378 channels

Example: Radiance Comparison

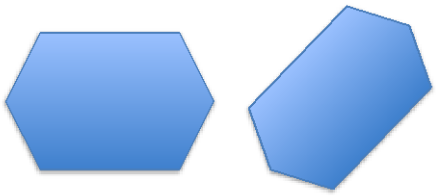
Put the center of the footprint
over the nearest point
and weight the Modis-pixel according
to the spatial response function
of AIRS and the spectral response
of the MODIS-channel

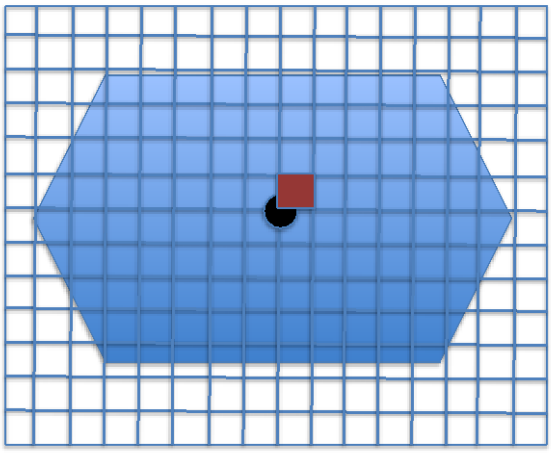
All footprints for all channels have to
be considered
and all scan angle dependence



Example: Cloud fraction

Simple average channel can be used
But scan angle dependence still important



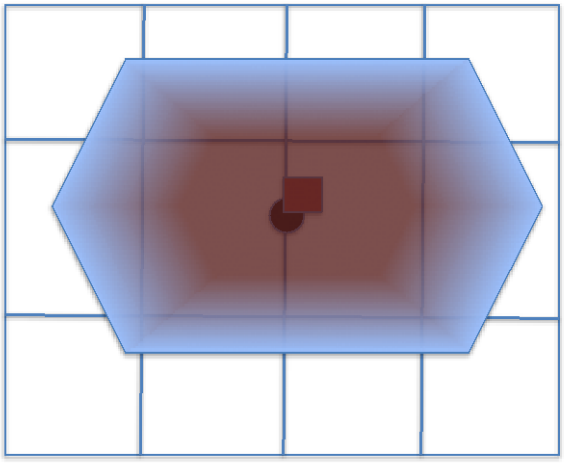


Simple average channel can be used

Example: Cloud properties 5km

An even simpler average channel can be used

Scan angle dependence:
If you want to …



Simple average channel can be used

The resulting product is an HDF-file with 2 arrays of 675x90 points, indicating the nearest MODIS-points to AIRS-footprint by column and line-number.

The following figure illustrates the matched AIRS and MODIS footprints for AIRS granules:

- AIRS.2016.12.31.001.L1B.AIRS_Rad.v5.0.23.0.G16366142111.hdf
- AIRS.2016.12.31.002.L1B.AIRS_Rad.v5.0.23.0.G16366142121.hdf
- AIRS.2016.12.31.003.L1B.AIRS_Rad.v5.0.23.0.G16366142205.hdf
- AIRS.2016.12.31.004.L1B.AIRS_Rad.v5.0.23.0.G16366152439.hdf
- AIRS.2016.12.31.005.L1B.AIRS_Rad.v5.0.23.0.G16366152428.hdf

And MODIS granules:

- MYD03.A2016366.0000.005.2016366161858.hdf
- MYD03.A2016366.0005.005.2016366161838.hdf
- MYD03.A2016366.0010.005.2016366161827.hdf
- MYD03.A2016366.0015.005.2016366161829.hdf
- MYD03.A2016366.0020.005.2016366161838.hdf
- MYD03.A2016366.0025.005.2016366161837.hdf
- MYD03.A2016366.0030.005.2016366161841.hdf
- MYD03.A2016366.0035.005.2016366161844.hdf

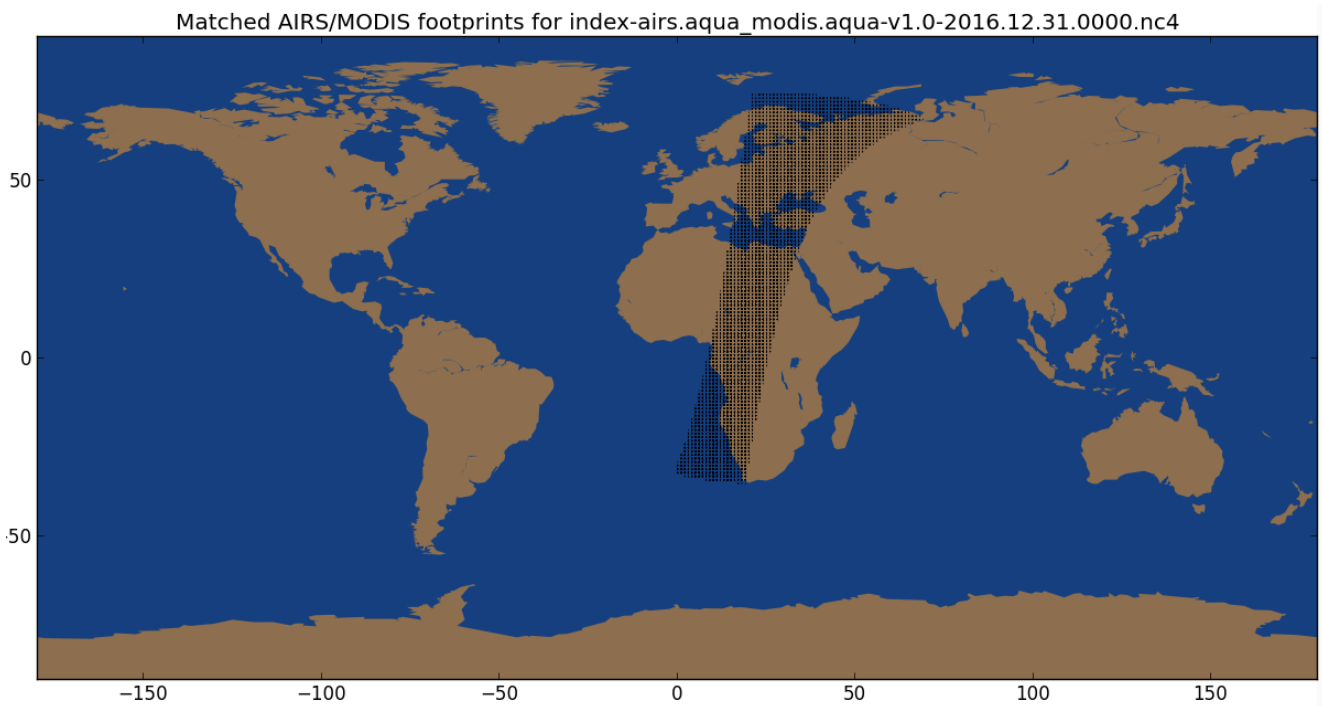


Figure 1.1: Footprint Plot

1.2.2 HDF4 to NetCDF4 conversion and metadata extraction

Conversion of the HDF4 output file to NetCDF4 is done using the HDF4 CF Conversion Toolkit. Metadata extraction is done using HySDS metadata extraction operators.

1.3 Data Disclaimer

For any questions regarding this dataset, please contact Gerald Manipon at Geraldjohn.M.Manipon@jpl.nasa.gov.

For detailed information on the v6 AMSU/AIRS data, consult the documentation located at https://disc.gsfc.nasa.gov/information/documents?title=AIRS Documentation

For detailed information on the collection 5/6 MODIS data, consult the documentation located at https://modis.ornl.gov/documentation.html.

For information and access tools pertaining to NetCDF, consult the documentation at http://www.unidata.ucar.edu/software/netcdf/.

For information on the Python programming language, consult the documentation at http://www.python.org.

For information on the HDF4 CF Conversion Toolkit, consult the documentation at http://hdfeos.org/software/h4cflib.php.

2.0 Data Organization

The data consists of:

1) the AIRS-MODIS matchup index files in NetCDF4 format
    - ShortName = AIRS_MDS_IND
    - LongName = AIRS-MODIS collocation indexes
    - VersionID = 010
    - DOI = 10.5067/MEASURES/WVCC/DATA206
    - DOI_Authority = http://dx.doi.org/

All products are indexed by 30 minute granules (6 AIRS granules to 8 MODIS granules).

2.1 File Naming Convention

2.1.1 AIRS_MDS_IND (Matchup Index NetCDF4 File)

index-airs.aqua_modis.aqua-*vm.m-yyyy.mm.dd.hhmm*.nc4

Where:
- m.m = algorithm version identifier is made up of major version and minor version, respectively.
- yyyy = 4 digit year number [2003-2017].
- mm = 2 digit month number [01-12]
- dd = day of month [01-31]
- hh = hour [00-23]
- mm = min [00, 30]

Filename example: index-airs.aqua_modis.aqua-v1.0-2016.12.31.0000.nc4

2.2 File Format and Structure

The AIRS_MDS_IND (matchup index file) data type is in NetCDF4 format (http://www.unidata.ucar.edu/software/netcdf/) and contains the AIRS-MODIS matchup indices. An example ncdump from the following matchup file, index-airs.aqua_modis.aqua-v1.0-2006.12.31.0000.nc4, will serve to explain its format:

```
{
   dimensions:
     AIRSX = 90;
     AIRSY = 675;
   variables:
     int Distance(AIRSY=675, AIRSX=90);
       :origname = "Distance";
       :long_name = "Distance";
       :_ChunkSizes = 675, 90; // int

     int Direction(AIRSY=675, AIRSX=90);
       :origname = "Direction";
       :long_name = "Direction";
```

```
        :_ChunkSizes = 675, 90; // int

      int Row_Point(AIRSY=675, AIRSX=90);
        :origname = "Row_Point";
        :long_name = "Row_Point";
        :_ChunkSizes = 675, 90; // int

      int Column_Point(AIRSY=675, AIRSX=90);
        :origname = "Column_Point";
        :long_name = "Column_Point";
        :_ChunkSizes = 675, 90; // int

      int Latitude_Point(AIRSY=675, AIRSX=90);
        :origname = "Latitude_Point";
        :long_name = "Latitude_Point";
        :_ChunkSizes = 675, 90; // int

      int Longitude_Point(AIRSY=675, AIRSX=90);
        :origname = "Longitude_Point";
        :long_name = "Longitude_Point";
        :_ChunkSizes = 675, 90; // int

      int Time_Point_EV_start_time_(AIRSY=675);
        :origname = "Time_Point (EV start time)";
        :long_name = "Time_Point (EV start time)";
        :_ChunkSizes = 675; // int

    // global attributes:
    :Big_X_Size = 1354; // int
    :Big_Y_Size = 16250; // int
    :Time_of_Creation = "Sun Jan 22 01:30:57 2017";
    :Complementary_Files_Start =
"/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0000.005.2016366161858.hdf";
    :Complementary_Files_End =
"/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0035.005.2016366161844.hdf";
    :Comp_FileNumberA = 8; // int
    :Comp_FileA0 = "/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0000.005.2016366161858.hdf";
    :Comp_FileA_sizeX0 = 1354S; // short
    :Comp_FileA_sizeY0 = 2030S; // short
    :Comp_FileA1 = "/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0005.005.2016366161838.hdf";
    :Comp_FileA_sizeX1 = 1354S; // short
    :Comp_FileA_sizeY1 = 2030S; // short
    :Comp_FileA2 = "/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0010.005.2016366161827.hdf";
    :Comp_FileA_sizeX2 = 1354S; // short
    :Comp_FileA_sizeY2 = 2030S; // short
    :Comp_FileA3 = "/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0015.005.2016366161829.hdf";
    :Comp_FileA_sizeX3 = 1354S; // short
    :Comp_FileA_sizeY3 = 2040S; // short
    :Comp_FileA4 = "/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0020.005.2016366161838.hdf";
    :Comp_FileA_sizeX4 = 1354S; // short
    :Comp_FileA_sizeY4 = 2030S; // short
    :Comp_FileA5 = "/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0025.005.2016366161837.hdf";
    :Comp_FileA_sizeX5 = 1354S; // short
    :Comp_FileA_sizeY5 = 2030S; // short
    :Comp_FileA6 = "/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0030.005.2016366161841.hdf";
    :Comp_FileA_sizeX6 = 1354S; // short
    :Comp_FileA_sizeY6 = 2030S; // short
    :Comp_FileA7 = "/raid15/mschreie/DATABASE/SATELLITE/LEVEL1/MODIS/MYD03/2016/366/MYD03.A2016366.0035.005.2016366161844.hdf";
    :Comp_FileA_sizeX7 = 1354S; // short
    :Comp_FileA_sizeY7 = 2030S; // short
    :Comp_FileNumberB = 5; // int
    :Comp_FileB0 = "/archive/AIRSOps/airs/gdaac/v5/2016/12/31/airibrad/AIRS.2016.12.31.001.L1B.AIRS_Rad.v5.0.23.0.G16366142111.hdf";
    :Comp_FileB_sizeX0 = 90S; // short
    :Comp_FileB_sizeY0 = 135S; // short
    :Comp_FileB1 = "/archive/AIRSOps/airs/gdaac/v5/2016/12/31/airibrad/AIRS.2016.12.31.002.L1B.AIRS_Rad.v5.0.23.0.G16366142121.hdf";
    :Comp_FileB_sizeX1 = 90S; // short
    :Comp_FileB_sizeY1 = 135S; // short
    :Comp_FileB2 = "/archive/AIRSOps/airs/gdaac/v5/2016/12/31/airibrad/AIRS.2016.12.31.003.L1B.AIRS_Rad.v5.0.23.0.G16366142205.hdf";
    :Comp_FileB_sizeX2 = 90S; // short
    :Comp_FileB_sizeY2 = 135S; // short
    :Comp_FileB3 = "/archive/AIRSOps/airs/gdaac/v5/2016/12/31/airibrad/AIRS.2016.12.31.004.L1B.AIRS_Rad.v5.0.23.0.G16366152439.hdf";
    :Comp_FileB_sizeX3 = 90S; // short
    :Comp_FileB_sizeY3 = 135S; // short
    :Comp_FileB4 = "/archive/AIRSOps/airs/gdaac/v5/2016/12/31/airibrad/AIRS.2016.12.31.005.L1B.AIRS_Rad.v5.0.23.0.G16366152428.hdf";
    :Comp_FileB_sizeX4 = 90S; // short
    :Comp_FileB_sizeY4 = 135S; // short
    :MINIMUM_COL = 73; // int
    :MAXIMUM_COL = 1281; // int
    :CREATOR = "Mathias Schreier <Mathias.Schreier@jpl.nasa.gov>";
    :COGNIZANT_ENGINEER = "Gerald Manipon <gmanipon@jpl.nasa.gov>";
    :VERSION = "1.0";
    :PRODUCTIONDATE = "2017-05-06T04:43:13Z";
    :IDENTIFIER_PRODUCT_DOI = "doi:10.5067/MEASURES/WVCC/DATA206";
    :IDENTIFIER_PRODUCT_DOI_AUTHORITY = "http://dx.doi.org/";
    :RANGEBEGINNINGDATE = "2016-12-31";
    :RANGEBEGINNINGTIME = "00:05:20";
    :RANGEENDINGDATE = "2016-12-31";
    :RANGEENDINGTIME = "00:35:12";
    :NORTHBOUNDINGCOORDINATE = 74; // int
    :SOUTHBOUNDINGCOORDINATE = -35; // int
    :EASTBOUNDINGCOORDINATE = 22; // int
    :WESTBOUNDINGCOORDINATE = 19; // int
```

```
}
```

There should be 48 files per day, with file containing over 30 minutes of data:

"_0100.nc4" indicates starttime 1:00 UTC to endtime 1:30 UTC.
"_0130.nc4" indicates starttime 1:30 UTC to endtime 2:00 UTC.

The size is similar to 5 stacked AIRS-granules. Reason: AIRS granules are 6 min, MODIS granules are 5 min. 30 min ensures therefore collocation without gaps.

Most arrays in the data have a size of 90 columns and 675 rows. That's the result of 5 stacked AIRS granules (90 scans per line and 5 x 135 scanlines). The AIRS files, which were used for the collocation, are granules XX1 - XX5 for the first 30 min and XX6-YY0 for the second 30 min.

Example:

The file index-airs.aqua_modis.aqua-v1.0-2006.12.31.0000.nc4
would refer to the 5 AIRS-granules AIRS.2006.12.31.001. to AIRS.2006.12.31.005.

The four main variables contained in the matchup index file are:

- Row_Point
- Column_Point
- Distance
- Direction

Row_Point gives you the MODIS-row of nearest MODIS-point to the referring AIRS-point.
Column_Point gives you the corresponding MODIS-row.
Distance and Direction give you the residual distance and direction of the two points, in case you want to do a collocation that's even more accurate than 1km.

2.3 Key Science Data Fields

2.3.1 Science Area 1: Atmospheric Moist Processes and Trends

Enter text here for this science area.

2.3.2 Science Area 2: Atmospheric Water Cycle

Enter text here for this science area.

3.0 Data Contents

3.1 Dimensions

3.1.1 AIRS_MDS_IND (Matchup Index File)

Name	Description
AIRSX	column
AIRSY	Row

3.2 Global Attributes

Global metadata are also stored in the files. Some metadata are required by standard conventions, some are present to meet data provenance requirements and others as a convenience to users of the WVCC AIRS-MODIS merged dataset. A summary of global attributes present in all netcdf files is shown in Table 3.3.

Global Attribute	Type	Description
COMP_FILEA[0-7]	String	Specifies MODIS granule used for the matchup
COMP_FILEB[0-5]	String	Specifies AIRS granule used for the matchup
PRODUCTIONDATE	String	Specifies production time in the format YYYYMMDDHHMMSS
RANGEBEGINNINGDATE	String	Start date of AIRS-MODIS matchup
RANGEBEGINNINGTIME	String	Start time of AIRS-MODIS matchup
RANGEENDINGDATE	String	Ending date of AIRS-MODIS matchup
RANGEENDINGTIME	String	Ending time of AIRS-MODIS matchup
NORTHBOUNDINGCOORDINATE	Float	North bounding coordinate of AIRS-MODIS matchup
SOUTHBOUNDINGCOORDINATE	Float	South bounding coordinate of AIRS-MODIS matchup
EASTBOUNDINGCOORDINATE	Float	East bounding coordinate of AIRS-MODIS matchup
WESTBOUNDINGCOORDINATE	Float	West bounding coordinate of AIRS-MODIS matchup
IDENTIFIER_PRODUCT_DOI	String	DOI identifier
IDENTIFIER_PRODUCT_DOI_AUTHORITY	String	URL for DOI authority
VERSION	String	Production version

Table 3.3: Global metadata attributes associated with each netcdf file

3.3 Products/Parameters

Refer to section 2.3 for product parameters.

4.0 Options for Reading the Data

4.1 Command Line Utilities

ncdump

The **ncdump** program can be found in bin directory of the HDF installation area. Consult your local computer system administrator for the specifics.

The **ncdump** tool can be used as a simple browser for NetCDF data files, to display the dimension names and sizes; variable names, types, and shapes; attribute names and values; and optionally, the values of data for all variables or selected variables in a NetCDF file. The most common use of **ncdump** is with the –h option, in which only the header information is displayed.

ncdump [-c|-h] [-v ...] [[-b|-f] [c|f]] [-l len] [-n name] [-d n[,n]] filename
Options/Arguments:
[-c] Coordinate variable data and header information
[-h] Header information only, no data
[-v var1[,...]] Data for variable(s) <var1>,... only data
[-f [c|f]] Full annotations for C or Fortran indices in data
[-l len] Line length maximum in data section (default 80)
[-n name] Name for netCDF (default derived from file name)
[-d n[,n]] Approximate floating-point values with less precision filename File name of input netCDF file

Note: the ncdump tool will only display variables whose ranks are great than 1. In other words, you will not see one dimensional vectors such as *satheight* using this tool.

4.2 Tools/Programming

4.2.1 Usage Example: AIRS and MODIS cloud top temperatures

The following example shows usage of the AIRS-MODIS matchup index to collocate and visualize AIRS and MODIS cloud top temperature. To run this script, it is assumed that you have:

- staged the AIRS-MODIS matchup index files of interest, e.g. index-airs.aqua_modis.aqua-v1.0-2006.01.01.0030.nc4

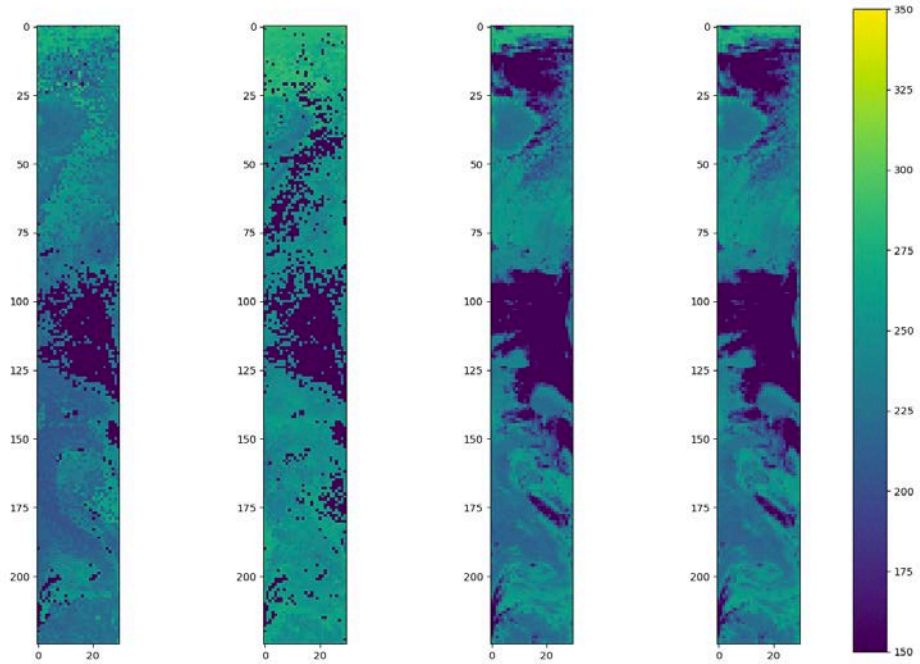
- have local disk access to the v6 AIRS AIRX2SUP products, e.g. AIRS.2006.01.01.006.L2.RetSup.v6.0.7.0.G13134230628.hdf
- have local disk access to the v6 MYD06_L2 products, e.g. MYD06_L2.A2006001.0030.006.2014035151123.hdf
- staged Evan Fishbein's AIRS-MODIS spatial spectral calibration database, e.g. https://docserver.gesdisc.eosdis.nasa.gov/public/project/MEaSUREs/Fetzer/AIRS_MODIS_spatial_spectral_calibration_database_2009.01.01.v9.5.3.nc4

```
#!/usr/bin/env python3
# Example program for AIRS/MODIS collocation data
# by M. Schreier
# 07/20/2017
# Shows comparison of cloud top temperatures from AIRS and MODIS
# uses nc-collocation files and hdf-satellite data

# use at your own risk


from netCDF4 import Dataset, Group
import glob
import calendar
import numpy as np
from pyhdf.SD import SD, SDC
import matplotlib.pyplot as plt


class airs_modis_collo:
    def __init__(self):
        self.row = -9999
        self.col = -9999
        self.circle = -9999
        self.spatial_average = -999


def get_yearday(sday, smon, syear):

    mondaynormal = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

    leapyes = calendar.isleap(int(syear))

    yday = 1
    for mo in mondaynormal[0:(int(smon) - 1)]:
        yday = yday + mo
        yday = yday + 1 if ((leapyes == 1) and (mo == 28)) else yday

    syearday = str(yday)
    syearday = '0' + syearday if (yday < 100) else syearday
    syearday = '0' + syearday if (yday < 10) else syearday
    return syearday


def read_modis_data(infiles, selectset):

    n = 0
    for f in infiles:
        file = SD(f, SDC.READ)
        sds_obj = file.select(selectset)
        data = np.vstack((data, sds_obj.get())) if n > 0 else sds_obj.get()
        n = n + 1
    for key, value in sds_obj.attributes().items():
        print(key, value)
        if key == 'add_offset':
            add_offset = value
        if key == 'scale_factor':
            scale_factor = value
    data = (data - add_offset) * scale_factor
    print(data.max())

    return data


def read_airs_data(infiles, selectset):

    n = 0
    for f in infiles:
        file = SD(f, SDC.READ)
        sds_obj = file.select(selectset)
        data = np.vstack((data, sds_obj.get())) if n > 0 else sds_obj.get()
        n = n + 1
        print(data.shape)
```

```
    return data


# change GIVENDIR to your data below - all subdirectories in GIVENDIR are by default :
# GIVENDIR/YYYY/MM/DD/DATA_SUBGROUP/
# you can adapt the code below accordingly ...

collodir = '/data/public/repository/products/wvcc/indexes/'
modisdir = '/data/public/repository/products/modis.aqua/v6/'
airsdir = '/archive/AIRSOps/airs/gdaac/v6/'

collo_info = airs_modis_collo()


# FOOTPRINT OPTION 1:
# Simple circle similar to an AIRS/AMSU footprint
a, b = 50, 50
n3 = 101
r = 22
y, x = np.ogrid[-a:n3 - a, -b:n3 - b]
mask = x * x + y * y <= r * r
array = np.zeros((n3, n3))
array[mask] = 1.
collo_info.circle = array


# FOOTPRINT OPTION 2 and 3:
# Averaged spatial response function or all spatial response functions

foot = Dataset(
    collodir +
    '/airs.aqua_modis.aqua/AIRS_MODIS_spatial_spectral_calibration_database_2009.01.01.v9.5.3.nc4',
    'r')

# to get MODIS channel averaged spatial response functions for AIRS - use
# variable here
collo_info.spatial_average = foot.variables['spatial'][:]
# we chose the one for channel 32
collo_info.spatial_average = collo_info.spatial_average[11, :, :, :]
nr = collo_info.spatial_average.shape[1] / 2
nc = collo_info.spatial_average.shape[2] / 2

print(nc, nr)

# to get ALL MODIS relevant response functions for AIRS - use variable here.
# only relevant for radiance comparisons
airsgrp = foot.groups['airs']
collo_info.spatial_channel = airsgrp.variables['spatial'][:]

collofiles = sorted(
    glob.glob(
        collodir +
        'airs.aqua_modis.aqua/v1.0/2006/01/01/index-airs.aqua_modis.aqua-v1.0-2006.01.01.0030/*nc4'))

for fil in collofiles:

    print(fil)

    foo = Dataset(fil, 'r')

    collo_info.row = foo.variables['Row_Point'][:]
    collo_info.col = foo.variables['Column_Point'][:]

    filsplit = fil.split('.')
    nn = len(filsplit)
    stime = filsplit[nn - 2]
    sday = filsplit[nn - 3]
    smon = filsplit[nn - 4]
    syear = filsplit[nn - 5].split('-')[1]

    syearday = get_yearday(sday, smon, syear)

    shour = stime[0:2]
    hourplus = int(shour) + 1
    shourplus = '0' + str(hourplus) if (hourplus < 10) else str(hourplus)
    smin = stime[2:4]

    if smin == '00':
        modistimes = ['00', '05', '10', '15', '20', '25', '30', '35']
        airstimes = ['1', '2', '3', '4', '5']
        modistimes = [shour + s for s in modistimes]
        airstimes = [shour + s for s in airstimes]

    if smin == '30':
        modistimes = ['30', '35', '40', '45', '50', '55', '00', '05']
        airstimes = ['6', '7', '8', '9', '0']
        modistimes1 = [shour + s for s in modistimes[0:6]]
        airstimes1 = [shour + s for s in airstimes[0:4]]
        modistimes2 = [shourplus + s for s in modistimes[6:9]]
        airstimes2 = [shourplus + s for s in airstimes[4:6]]
        modistimes = modistimes1 + modistimes2
        airstimes = airstimes1 + airstimes2
```

```
    myd06files = []
    airx2supfiles = []
    for mtim in modistimes:
        myd_glob = modisdir + 'MYD06_L2/' + syear + '/' + smon + '/' + \
            sday + '/' + '*A' + syear + syearday + '.' + mtim + '*hdf'
        print(myd_glob)
        myd06files.append(glob.glob(myd_glob)[0])
    print(myd06files)
    for atim in airstimes:
        airx2sup_glob = airsdir + '/' + syear + '/' + smon + '/' + sday + \
            '/airx2sup/' + '*.' + syear + '.' + smon + '.' + sday + '.' + atim + '*hdf'
        print(airx2sup_glob)
        airx2supfiles.append(glob.glob(airx2sup_glob)[0])
    print(airx2supfiles)

    # selected here: modis cloud top temperature
    modis_data = read_modis_data(myd06files, 'cloud_top_temperature_1km')
    # selected here: AIRS cloud top temperature
    airs_data = read_airs_data(airx2supfiles, 'TCldTopStd')

    airssize = airs_data.shape

    modis_aver1 = np.zeros((airssize[0], airssize[1]))
    modis_aver2 = np.zeros((airssize[0], airssize[1]))

    combine = np.copy(collo_info.circle)

    for r in range(0, airssize[0]):
        for c in range(0, airssize[1]):

            # using FOOTPRINT OPTION 1:

            lookr3 = r * 3 + 1
            lookc3 = c * 3 + 1

            modisr3 = collo_info.row[lookr3, lookc3]
            modisc3 = collo_info.col[lookr3, lookc3]

            x1 = int(modisr3 - n3 / 2)
            x2 = int(modisr3 + n3 / 2)
            y1 = int(modisc3 - n3 / 2)
            y2 = int(modisc3 + n3 / 2)
            #print("modisr3: {}".format(modisr3))
            #print("n3: {}".format(n3))
            #print(x1, x2, y1, y2)
            #print("modis_data.shape: {}".format(modis_data.shape))
            #print("modis_data[{}:{}, {}:{}].shape: {}".format(x1, x2, y1, y2, modis_data[x1:x2, y1:y2].shape))
            #print("collo_info.circle.shape: {}".format(collo_info.circle.shape))
            #print(np.sum(modis_data[x1:x2, y1:y2]*collo_info.circle[:,:])/np.sum(collo_info.circle))
            #print("modis_aver1.shape: {}".format(modis_aver1.shape))
            #print("modis_aver1[{}, {}] = {}".format(r, c, modis_aver1[r, c]))
            modis_aver1[r, c] = np.sum(modis_data[x1:x2, y1:y2] * collo_info.circle[:, :]) / np.sum(collo_info.circle)

            # using FOOTPRINT OPTION 2 :
            aver = 0.
            for cc in range(-1, 2):
                for rr in range(-1, 2):

                    lookc = c * 3 + 1 + cc
                    lookr = r * 3 + 1 + rr

                    modisc = collo_info.col[lookr, lookc]
                    modisr = collo_info.row[lookr, lookc]

                    x1 = int(modisr - nr)
                    x2 = int(modisr + nr)
                    y1 = int(modisc - nc)
                    y2 = int(modisc + nc)
                    aver = aver + np.sum(modis_data[x1:x2, y1:y2] * collo_info.spatial_average[lookc, :, :]) / \
                                  np.sum(collo_info.spatial_average[lookc, :, :])

            modis_aver2[r, c] = aver / 9.

data = [airs_data[:, :, 0], airs_data[:, :, 1], modis_aver1, modis_aver2]

fig, axes = plt.subplots(nrows=1, ncols=4)
for dat, ax in zip(data, axes.flat):
    im = ax.imshow(dat, vmin=150., vmax=350.)

cax = fig.add_axes([0.9, 0.1, 0.03, 0.8])
fig.colorbar(im, cax=cax)
plt.show()
```

This script should produce a plot similar to the following



5.0 Data Services

GES DISC provides basic temporal and advanced (event) searches and other services that can be drilled from the product pages:
https://disc.gsfc.nasa.gov/datasets?page=1&keywords=AIRS_CPR_IND%20AIRS_CPR_MAT%20AIRSM_CPR_MAT%20AIRS_MDS_IND

As the project progresses, more services will be added that will allow options for subsetting, format conversion, and previews using open source software such as OPeNDAP and Panoply.

If you need assistance or wish to report a problem:
Email: gsfc-help-disc@lists.nasa.gov
Voice: 301-614-5224
Fax: 301-614-5268
Address:
Goddard Earth Sciences Data and Information Services Center NASA Goddard Space Flight Center Code 610.2 Greenbelt, MD 20771 USA

6.0 More Information

All WVCC data types are published to the Global Change Master Directory:

http://gcmd.nasa.gov

This is a centralized depository of climate data information that is catalogued by popular keywords, which facilitate data discovery. Since the Directory is visited by a large number of people working on a broad range of research topics, it is an excellent forum to popularize data collections.

7.0 Acknowledgements

The creators for WVCC AIRS-MODIS are:

Mathias Schreier, Evan Fishbein, Eric Fetzer, Hook Hua, Brian Wilson and Gerald Manipon.
Please see the project's product pages at GES DISC for exact format of Data Citation:

https://disc.gsfc.nasa.gov/datasets?page=1&keywords=AIRS_CPR_IND%20AIRS_CPR_MAT%20AIRSM_CPR_MAT%20AIRS_MDS_IND

References

AIRS Version 6.0 Processing Files Description:
https://disc.gsfc.nasa.gov/information/documents?title=AIRS Documentation

MODIS Data Products: https://modis.gsfc.nasa.gov/data/dataprod/

Level 2 Cloud Scenario Classification Product Process Description and Interface Control Document, Version: 4.0, March 1, 2005.

NetCDF4: http://www.unidata.ucar.edu/software/netcdf/

For information on the Python programming language, consult the documentation:
http://www.python.org.