Polar2Grid Documentation

Release 3.0

Released through the
NOAA Community Satellite Processing Package (CSPP)

Feb 21, 2023

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Overview

Polar2Grid is a set of command line tools for extracting data from earth-observing satellite instrument files, remapping it to uniform grids if needed, and writing that gridded data to a new file format. It provides an easy way to create high quality projected images. Polar2Grid was created by scientists and software developers at the SSEC. It is distributed as part of the CSPP LEO project for processing of data received via direct broadcast antennas. Although Polar2Grid was created to serve the direct broadcast community, it can be used on most archived data files.

The features provided by Polar2Grid are accessible via bash scripts and binary command line tools. This is meant to give scientists an easy way to use and access features that typically involve complicated programming interfaces. Linux terminal commands included in these instructions assume the bash shell is used.

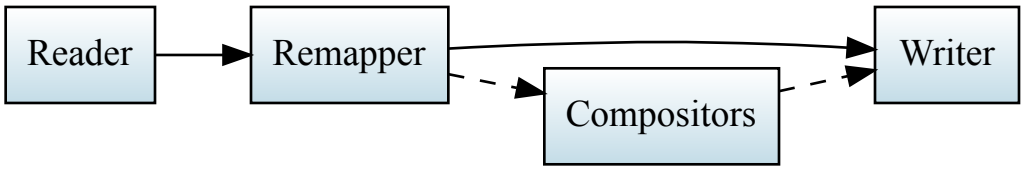
Documentation Website

Contact Us

GitHub Repository

CSPP LEO Forum

1.2 Software Design



Polar2Grid has a modular design operating on the idea of satellite "products" or "datasets"; data observed by a satellite instrument. These products can be any type of raster data, such as temperatures, reflectances, radiances, or any other value that may be recorded by or calculated from an instrument. There are 4 main steps or components involved when working with these products in Polar2Grid: reading, writing, compositing, and remapping. Polar2Grid makes it possible to access and configure these steps with a simple bash script called `polar2grid.sh` and other helper scripts. More information on accessing Polar2Grid's features and running its scripts can be found in the *Polar2Grid Basics* section or

the examples following each reader section. Note that although an example may be written for a specific reader the same operations can be applied to all readers unless mentioned otherwise.

For more low-level information on the design and responsibility of each component see the *Software Design Overview* Appendix.

In Polar2Grid a majority of the functionality is provided by the open source SatPy library created by the Pytroll group. More information on SatPy and the capabilities it provides to python users can be found in the SatPy documentation.

1.3 What's New?

Polar2Grid Version 3.0 is now available. This is a major update that includes changes to basic Polar2Grid execution. These changes bring Polar2Grid in conformity with the execution strategy of Geo2Grid, and takes advantage of the Xarray and Dask python libraries.

Please see the example executions listed at the end of every reader description in this document, as well as the updated examples in the *Examples* section. Finally, the Appendix includes a longer list of changes and direct comparisons of Polar2Grid V2.3 to V3.0 executions. See *Version 2.3 to Version 3.0 Command Changes*.

- New Implementation "polar2grid.sh -r <reader> -w <writer>"
- Reader name changes and replacements
- Writer name changes
- NOAA20 output file names standardized to "noaa20" prefix
- Alpha Band now included as default. Use `--fill-value 0` to not include
- Specify number of CPU's to use `--num-workers <num>`
- `--list-products` and `--list-product-all` now available.
- Scaling ".ini" files no longer supported. Replaced with ".yaml"

For more details on what's new in this version and past versions see the Polar2Grid Release Notes in the github repository.

1.4 System Requirements

System requirements for the Polar2Grid software are as follows:

- Intel or AMD CPU with 64-bit instruction support (2+ cores - 2.4GHz)
- 16 GB RAM (minimum)
- CentOS 7.9 64-bit Linux; the software has been tested successfully on Rocky Linux 8.5
- 5 GB disk space (minimum)

1.5 Improved Execution Times

Updates in Polar2grid Version 3.0 result in improved image creation times. The table below presents a comparison of the unix *real* time required to create VIIRS and MODIS imager GeoTIFF files for the given segments of data in the default WGS84 projection. In these examples, the default 4 computer threads were used in the Version 3.0 executions. Execution times decrease when fewer bands and smaller data segments are processed.

Table of Execution Times for Creating GeoTIFF Default Projection Images

Instrument Input	Polar2Grid V2.3 True and False Color	Polar2grid V3.0 True and False Color	Polar2Grid2 V2.3 All Bands plus True and False Color	Polar2Grid V3.0 All Bands plus True and False Color
VIIRS SDR 10 - 86 second granules	4m52s	2m46s	12m54s	4m32s
MODIS Level 1B 3 - 5 minute granules	4m11s	3m55s	9m08s	4m51s

1.6 License and Disclaimer

Original scripts and automation included as part of this package are distributed under the `GNU GENERAL PUBLIC LICENSE agreement version 3`. Software included as part of this software package are copyrighted and licensed by their respective organizations, and distributed consistent with their licensing terms.

The University of Wisconsin-Madison Space Science and Engineering Center (SSEC) makes no warranty of any kind with regard to the CSPP LEO software or any accompanying documentation, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. SSEC does not indemnify any infringement of copyright, patent, or trademark through the use or modification of this software.

There is no expressed or implied warranty made to anyone as to the suitability of this software for any purpose. All risk of use is assumed by the user. Users agree not to hold SSEC, the University of Wisconsin-Madison, or any of its employees or assigns liable for any consequences resulting from the use of the CSPP LEO software.

CHAPTER

TWO

INSTALLATION

Polar2Grid is released as an all-in-one tarball for Enterprise Linux systems. The tarball, or software bundle, provided by the CSPP for Low Earth Orbiter Satellites (CSPP LEO) team includes a python runtime and all of the necessary third-party software to run the features provided by Polar2Grid. The tarball uses bash scripts for conveniently calling the python software or utilities provided by third-party vendors. The software bundle is only supported on CentOS-7.9 compatible systems, but has also been tested on Rocky Linux 8.5 and may work on other compatible Linux 64-bit operating systems as well. There are other ways to install Polar2Grid on other operating systems, but the instructions to do so are beyond the scope of this documentation.

Please Contact Us if you have questions about installation.

2.1 Polar2Grid Software

The Polar2Grid tarball can be downloaded from the CSPP LEO website: http://cimss.ssec.wisc.edu/cspp/

To install the software, unpack the tarball:

```
tar xf CSPP_POLAR2GRID_V3.0.tar.gz
```

This will create a Polar2Grid software bundle directory, `polar2grid_v_3_0`. To simplify calling scripts included in the bundle the following line should be added to your `.bash_profile`:

```
export POLAR2GRID_HOME=/path/to/softwarebundle
```

All other environment information needed to run is automatically loaded by the scripts provided by Polar2Grid. Scripts are typically invoked using:

```
$POLAR2GRID_HOME/bin/<p2g_script.sh> ...
```

To execute commands without including the preceding directory path, or if using in a script in its own background environment, then set the path to include the /bin directory:

```
export PATH=$PATH:$POLAR2GRID_HOME/bin
```

Note: A one-time initialization process is performed the first time any of the bash scripts are run. The extracted directory can *NOT* be moved after this is performed. In a shared user installation (multiple users running the same installation), the user that extracted the tarball should run a script to perform this initialization before any other users (ex. `-h` to `polar2grid.sh`).

See *Polar2Grid Basics* for more information on running Polar2Grid.

2.2 Polar2Grid Test Data

To confirm a successful installation download the following verification test data set:

```
CSPP_POLAR2GRID_V3.0_TEST_DATA.tar.gz
```

The test data should be unpacked in a directory separate from the Polar2Grid installation:

```
cd $HOME
tar xf CSPP_POLAR2GRID_V3.0_TEST_DATA.tar.gz
```

This will create a `polar2grid_test` directory containing the test input, output, and verification scripts for both MODIS and VIIRS instruments.

See *Verifying your Polar2Grid Installation* for instructions on using the verification test data.

CHAPTER

THREE

POLAR2GRID BASICS

All of the tools provided by Polar2Grid can be found in the `bin` directory of the extracted tarball. The majority of the scripts in the software bundle are bash wrappers around python software.

3.1 Basic Usage

The most common use of Polar2Grid is to convert satellite data files in to gridded image files. As an example, the following command can be used to create GeoTIFF single band images of all S-NPP VIIRS imager SDR calibrated data with accompanying geolocation files found in `<path to files>/<list of files>`.

```
$POLAR2GRID_HOME/bin/polar2grid.sh -r viirs_sdr -w geotiff -f <path to files>/<list␣
↪of files>
```

This script takes advantage of the modular design of Polar2Grid; a user only needs to decide on a *Reader* and a *Writer* and provide them to `polar2grid.sh`. In Polar2Grid the `<path to files>` will be searched for the necessary files to make as many products as possible. Similarly if processing errors occur Polar2Grid will attempt to continue processing to make as many products as it can.

For example, executing the command above will create 8-bit GeoTIFF files of all M-Band, I-Band, and Day/Night Band SDR files it finds in the directory as long as it contains the matching geolocation files. If multiple granules are provided to `polar2grid.sh` they will be aggregated together. By default the above command resamples the data to a Google Earth compatible Platte Carrée projected grid at ~600m resolution, but this can be changed with command line arguments. The GeoTIFF contains 2 bands, including an Alpha band.

3.2 Common Script Options

Additional command line arguments for the `polar2grid.sh` script and their defaults are described in the related *Reader* or *Writer* sections. Options that affect remapping are described in the *Remapping* section. Additionally all Polar2Grid bash scripts accept a `-h` argument to list all the available command line arguments. Although the available command line arguments may change depending on the reader and writer specified, there are a set of common arguments that are always available:

-r	Instrument input files to read from.
-w	Output format to write to.
-h	Print helpful information.
--list-products	List all possible product options to use with -p from the given input data and exit.
--list-products-all	List available polar2grid products options and custom/Satpy products and exit.

-p	List of products you want to create.
-f	Input files and paths.
--grid-coverage	Fraction of grid that must be covered by valid data. Default is 0.1.
-g <grid_name>	Specify the output grid to use. Default is the Platte Carrée projection, also known as the wgs84 coordinate system. See *Grids* and *Custom Grids* for information on possible values.
--num-workers NUM_WORKERS	Specify number of worker threads to use (Default: 4).
--progress	Show processing progress bar (Not recommended for logged output).
-v	Print detailed log information.

Examples:

```
polar2grid.sh -r viirs_sdr -w geotiff -p i01 dynamic_dnb -g polar_alaska_300 --grid-
↪coverage=.25 -v -f <path to files>

polar2grid.sh -r modis_l1b -w geotiff --list-products -f <path to files>/<list of␣
↪files>
```

For information on other scripts and features provided by Polar2Grid see the *Utility Scripts* section or the various examples throughout the document.

3.3 Reader/Writer Combinations

The tables below provide a summary of the possible combinations of readers and writers and expectations for the inputs and outputs of `polar2grid.sh`. To access these features provide the "reader" and "writer" names to the `polar2grid.sh` script followed by other script options:

```
$POLAR2GRID_HOME/bin/polar2grid.sh -r <reader> -w <writer> --list-products <options> -
↪f /path/to/files
```

Table 3.1: Reader/Writer Summary Table (Subset of Readers)

Input Source	Input Filename Pattern	Output Type	Reader Name	Writer Name
Suomi-NPP VIIRS Sensor Data Records	SVI01_npp_*.h5 GITCO_npp_*.h5	8-bit single band GeoTIFF	viirs_sdr	geotiff
“	“	AWIPS NetCDF	viirs_sdr	awips_tiled
“	“	HDF5	viirs_sdr	hdf5
“	“	Binary	viirs_sdr	binary
“	“	24-bit true and false color GeoTIFF	viirs_sdr	geotiff
Aqua and Terra MODIS Level 1b (IMAPP or NASA archive files)	MOD021KM*.hdf MOD03*.hdf or t1.*1000m.hdf t1.*.geo.hdf	8 bit single band GeoTIFF	modis_l1b	geotiff
“	“	AWIPS NetCDF	modis_l1b	awips_tiled
“	“	HDF5	modis_l1b	hdf5
“	“	Binary	modis_l1b	binary
“	“	24-bit true and false color GeoTIFF	modis_l1b	geotiff
NOAA-18, NOAA-19,Metop-A,-B,-C AVHRR AAPP Level 1b	hrpt_noaa18_*.l1b	8 bit single band GeoTIFF	avhrr_l1b	geotiff
“	“	AWIPS NetCDF	avhrr_l1b	awips_tiled
“	“	HDF5	avhrr_l1b	hdf5
“	“	Binary	avhrr_l1b	binary
GCOM-W1 ASMR2 L1B	GW1AM2*L1DLBTBR*.h5	8 bit single band GeoTIFF	amsr2_l1b	geotiff
“	“	AWIPS NetCDF	amsr2_l1b	awips_tiled
“	“	HDF5	amsr2_l1b	hdf5
“	“	Binary	amsr2_l1b	binary
FY3-D MERSI2 L1B	tf*.FY3D-X_MERSI_*_L1B.HDF	8 bit single band GeoTIFF, 24-bit true and false color GeoTIFF	mersi2_l1b	geotiff
“	“	HDF5	mersi2_l1b	hdf5
“	“	Binary	mersi2_l1b	binary
CLAVR-x Cloud Retrievals	clavrx*.hdf	8 bit single band GeoTIFF	clavrx	geotiff
“	“	AWIPS NetCDF	clavrx	awips_tiled
“	“	HDF5	clavrx	hdf5
“	“	Binary	clavrx	binary
ACSPO Sea Surface Temperatures	*-STAR-L2P_GHRSST-SSTskin-*.nc	8 bit single band GeoTIFF	acspo	geotiff
“	“	AWIPS NetCDF	acspo	awips_tiled
“	“	HDF5	acspo	hdf5
“	“	Binary	acspo	binary

3.4 Creating Your Own Custom Grids

The Polar2Grid software bundle comes with a script for *Custom Grid Utility* that allows users to easily create Polar2Grid custom grid definitions over a user determined longitude and latitude region. Once these definitions have been created, they can be provided to Polar2Grid. To run the utility script from the software bundle wrapper run:

```
$POLAR2GRID_HOME/bin/p2g_grid_helper.sh ...
```

See the *script's documentation* for more information on how to use this script and the arguments it accepts.

CHAPTER

FOUR

READERS

Readers are the first component used in Polar2Grid processing. Their main responsibility is to extract input satellite imager data and the associated metadata from user provided input files. The data that readers distribute to other Polar2Grid components are called "products" ("datasets" in SatPy terminology).

The number and type of products that can be created is dependent upon the input datasets that are provided. Composites, such as RGBs, require a specific set of band combinations to be present. All products that can be created for a given input dataset can be determined by using the `--list-products` option.

4.1 VIIRS SDR Reader

The VIIRS SDR Reader operates on Sensor Data Record (SDR) HDF5 files from the Suomi National Polar-orbiting Partnership's (NPP) and/or the NOAA20 Visible/Infrared Imager Radiometer Suite (VIIRS) instrument. The VIIRS SDR reader ignores filenames and uses internal file content to determine the type of file being provided, but SDR are typically named as below and have corresponding geolocation files:

```
SVI01_npp_d20120225_t1801245_e1802487_b01708_c20120226002130255476_noaa_ops.h5
```

The VIIRS SDR reader supports all instrument spectral bands, identified as the products shown below. It supports terrain corrected or non-terrain corrected navigation files. Geolocation files must be included when specifying filepaths to readers and `polar2grid.sh`. The VIIRS reader can be specified to the `polar2grid.sh` script with the reader name `viirs_sdr`.

This reader's default remapping algorithm is `ewa` for Elliptical Weighted Averaging resampling. The `--weight-delta-max` parameter set to 40 and the `--weight-distance-max` parameter set to 2.

Product Name	Description
i01	I01 Reflectance Band
i02	I02 Reflectance Band
i03	I03 Reflectance Band
i04	I04 Brightness Temperature Band
i05	I05 Brightness Temperature Band
i01_rad	I01 Radiance Band
i02_rad	I02 Radiance Band
i03_rad	I03 Radiance Band
i04_rad	I04 Radiance Band
i05_rad	I05 Radiance Band
m01	M01 Reflectance Band
m02	M02 Reflectance Band
m03	M03 Reflectance Band

continues on next page

Table 4.1 – continued from previous page

Product Name	Description
m04	M04 Reflectance Band
m05	M05 Reflectance Band
m06	M06 Reflectance Band
m07	M07 Reflectance Band
m08	M08 Reflectance Band
m09	M09 Reflectance Band
m10	M10 Reflectance Band
m11	M11 Reflectance Band
m12	M12 Brightness Temperature Band
m13	M13 Brightness Temperature Band
m14	M14 Brightness Temperature Band
m15	M15 Brightness Temperature Band
m16	M16 Brightness Temperature Band
m01_rad	M01 Radiance Band
m02_rad	M02 Radiance Band
m03_rad	M03 Radiance Band
m04_rad	M04 Radiance Band
m05_rad	M05 Radiance Band
m06_rad	M06 Radiance Band
m07_rad	M07 Radiance Band
m08_rad	M08 Radiance Band
m09_rad	M09 Radiance Band
m10_rad	M10 Radiance Band
m11_rad	M11 Radiance Band
m12_rad	M12 Radiance Band
m13_rad	M13 Radiance Band
m14_rad	M14 Radiance Band
m15_rad	M15 Radiance Band
m16_rad	M16 Radiance Band
dnb	Raw DNB Band (not useful for images)
histogram_dnb	Histogram Equalized DNB Band
adaptive_dnb	Adaptive Histogram Equalized DNB Band
dynamic_dnb	Dynamic DNB Band from Steve Miller and Curtis Seaman. Uses erf to scale the data.
hncc_dnb	Simplified High and Near-Constant Contrast Approach from Stephan Zinke
ifog	Temperature difference between I05 and I04
i_solar_zenith_angle	I Band Solar Zenith Angle
i_solar_azimuth_angle	I Band Solar Azimuth Angle
i_sat_zenith_angle	I Band Satellite Zenith Angle
i_sat_azimuth_angle	I Band Satellite Azimuth Angle
m_solar_zenith_angle	M Band Solar Zenith Angle
m_solar_azimuth_angle	M Band Solar Azimuth Angle
m_sat_zenith_angle	M Band Satellite Zenith Angle
m_sat_azimuth_angle	M Band Satellite Azimuth Angle
dnb_solar_zenith_angle	DNB Band Solar Zenith Angle
dnb_solar_azimuth_angle	DNB Band Solar Azimuth Angle
dnb_sat_zenith_angle	DNB Band Satellite Zenith Angle
dnb_sat_azimuth_angle	DNB Band Satellite Azimuth Angle
dnb_lunar_zenith_angle	DNB Band Lunar Zenith Angle
dnb_lunar_azimuth_angle	DNB Band Lunar Azimuth Angle

continues on next page

Table 4.1 – continued from previous page

Product Name	Description
true_color	Ratio sharpened rayleigh corrected true color
false_color	Ratio sharpened rayleigh corrected false color

4.1.1 Command Line Arguments

```
usage: polar2grid.sh -r viirs_sdr -w <writer> [-h] [--i-bands] [--m-bands]
                                              [--dnb-angle-products]
                                              [--dnb-saturation-correction]
                                              [--i-angle-products]
                                              [--m-angle-products]
                                              [--m-rad-products]
                                              [--i-rad-products]
                                              [--awips-true-color]
                                              [--awips-false-color]
```

VIIRS SDR Reader

--i-bands Add all I-band raw products to list of products

--m-bands Add all M-band raw products to list of products

--dnb-angle-products Add DNB-band geolocation 'angle' products to list of products

--dnb-saturation-correction Enable dynamic DNB saturation correction (normally used for aurora scenes)

--i-angle-products Add I-band geolocation 'angle' products to list of products

--m-angle-products Add M-band geolocation 'angle' products to list of products

--m-rad-products Add M-band geolocation radiance products to list of products

--i-rad-products Add I-band geolocation radiance products to list of products

--awips-true-color Add individual CREFL corrected products to create the 'true_color' composite in AWIPS.

--awips-false-color Add individual CREFL corrected products to create the 'false_color' composite in AWIPS.

Examples:

```
polar2grid.sh -r viirs_sdr -w geotiff -f <path to files>/<list of files>

polar2grid.sh -r viirs_sdr -w geotiff -h

polar2grid.sh -r viirs_sdr -w geotiff --list-products -f ../sdr/*.h5

polar2grid.sh -r viirs_sdr -w geotiff --fill-value 0 -f ../sdr/*.h5

polar2grid.sh -r viirs_sdr -w geotiff -p true_color false_color --num-workers 8 --
↪tiled -f ../sdr/*.h5

polar2grid.sh -r viirs_sdr -w awips_tiled -p i04 adaptive_dnb dynamic_dnb --awips-
↪true-color --awips-false-color --sza-threshold=90.0 --letters --compress --sector-
↪id Polar -g polar_alaska_1km --dnb-saturation-correction -f <path to files>
```

(continues on next page)

(continued from previous page)

```
polar2grid.sh -r viirs_sdr -w hdf5 --compress gzip --m-bands -f ../sdr/*.h5

polar2grid.sh -r viirs_sdr -w binary -g lcc_fit -p m15 --num-workers 8 -f ../sdr/
↪SVM15*.h5 ../sdr/GMTCO*.h5
```

4.1.2 Product Explanation

True Color

The VIIRS SDR "true_color" composite produced by Polar2Grid provides a view of the Earth as the human eye would see it; or as close as we can come to with satellite data and the channels we have available. This means things like trees and grass are green, water is blue, deserts are red/brown, and clouds are white. The True Color GeoTIFF 24 bit image is an *RGB (Red, Green, Blue) image* consisting of a combination of Red: VIIRS M-Band 5, Green: VIIRS M-Band 4, and Blue: VIIRS M-Band 3 reflectance channels. Each channel goes through a series of adjustments to produce the final high quality image output by Polar2Grid.

Creation of true color RGBs includes the following steps:

- Atmospheric *Rayleigh Scattering Correction* of the RGB visible reflectances.
- Combining the 3 channels into a 24 bit output file.
- *Sharpening of the image* to full 350m resolution if the VIIRS I-Band 1 is provided as input.
- Application of a *non-linear enhancement*.

See the *Creating VIIRS SDR GeoTIFF Files* example to see how Polar2Grid can be used to make this product as a GeoTIFF and KMZ file.

False Color

A false color image is any combination of 3 bands outside of those used to create a "true color" image (see above). These combinations can be used to highlight features in the observations that may not be easily identified in individual band imagery. Polar2Grid can readily create a preconfigured legacy false color (product false_color) GeoTIFF 24 bit image that consists of a combination *RGB (Red, Green, Blue) image* using uses Red:VIIRS M-Band 11 (2.25 μm), Green:VIIRS M-Band 7 (.87 μm) and Blue:VIIRS M-Band 5 (.67 μm). This band combination is very effective at distinguishing land/water boundaries as well as burn scars.

Creation of VIIRS legacy false color RGBs includes the following steps:

- Atmospheric *Rayleigh Scattering Correction* of the RGB visible reflectances.
- Combining the 3 channels into a 24 bit output file.
- *Sharpening of the image* to full 350m resolution if the VIIRS I-Band 2 is provided as input.
- Application of a *non-linear enhancement*.

See the *Creating VIIRS SDR GeoTIFF Files* example to see how Polar2Grid can be used to make this product as a GeoTIFF and KMZ output file.

Fog - Temperature Difference

The VIIRS SDR reader can also produce a "ifog" product which is a simple difference of the infrared brightness temperatures between the I05 (11.45 μm) and I04 (3.74 μm) bands (I05 - I04). The result is scaled linearly between -10.0 and 10.0 Kelvin before being saved to an output image.

Day Night Band

Polar2Grid allows the user to create images from the VIIRS Day/Night Band, which contains observations of visible radiances for both day and night. Polar2Grid provides 4 options for enhancing and scaling the DNB data. A full description of these options are described in detail in the *Scaling of the VIIRS Day/Night Band in Polar2Grid* appendix.

Reflectance I-Bands 01-03 and M-Bands 01-11

The I01-I03 and M01-M11 bands are visible reflectance channels on the VIIRS instrument. Besides the basic calibration necessary to convert the radiance values to reflectances, the data is passed through a square root function before being written to a grayscale image. The square root operation has the effect of balancing the bright and dark regions of the image.

Infrared I-Bands 04-05 and M-Bands 12-16

The I04-I05 and M12-M16 bands are all brightness temperature (infrared (IR)) channels. To produce a grayscale image with dark land and white clouds, the data is inverted and scaled linearly in two segments. The first segment is from 163K to 242K, the second 242K to 330K. This is a common scaling used by the National Weather Service (NWS) for their AWIPS visualization clients.

4.2 VIIRS L1B Reader

The VIIRS Level 1B Reader operates on NASA Level 1B (L1B) NetCDF files.

The files are from the Visible/Infrared Imager Radiometer Suite (VIIRS) instrument. The VIIRS L1B reader analyzes the user provided filenames to determine if a file can be used. Files usually have the following naming scheme (example from the SNPP VIIRS instrument):

```
VL1BI_snpp_d20160101_t185400_c20160301041812.nc
```

The VIIRS L1B reader supports all instrument spectral bands, identified as the products shown below. Geolocation files must be included when specifying filepaths to readers and `polar2grid.sh`. Therefore, the creation of The VIIRS L1B frontend can be specified to the Polar2Grid glue script with the frontend name `viirs_l1b`.

The list of supported products includes true and false color imagery. These are created by means of a python based atmospheric Rayleigh Scattering algorithm that is executed as part of the Polar2Grid VIIRS L1B reader.

Note: The VIIRS L1B reader only supports the NASA L1B version 2.0 file format. Previous and future versions may work for some products, but are not guaranteed.

This reader's default resampling algorithm is `ewa` for Elliptical Weighted Averaging resampling. The `--weight-delta-max` parameter is set to 40 and the `--weight-distance-max` parameter is set to 2.

Product Name	Description
i01	I01 Reflectance Band
i02	I02 Reflectance Band
i03	I03 Reflectance Band
i04	I04 Brightness Temperature Band
i05	I05 Brightness Temperature Band
m01	M01 Reflectance Band
m02	M02 Reflectance Band
m03	M03 Reflectance Band
m04	M04 Reflectance Band
m05	M05 Reflectance Band
m06	M06 Reflectance Band
m07	M07 Reflectance Band
m08	M08 Reflectance Band
m09	M09 Reflectance Band
m10	M10 Reflectance Band
m11	M11 Reflectance Band
m12	M12 Brightness Temperature Band
m13	M13 Brightness Temperature Band
m14	M14 Brightness Temperature Band
m15	M15 Brightness Temperature Band
m16	M16 Brightness Temperature Band
DNB	Raw DNB Band (not useful for images)
histogram_dnb	Histogram Equalized DNB Band
adaptive_dnb	Adaptive Histogram Equalized DNB Band
dynamic_dnb	Dynamic DNB Band from Steve Miller and Curtis Seaman. Uses erf to scale the data
hncc_dnb	Simplified High and Near-Constant Contrast Approach from Stephan Zinke
ifog	Temperature difference between I05 and I04
i_solar_zenith_angle	I Band Solar Zenith Angle
i_solar_azimuth_angle	I Band Solar Azimuth Angle
i_sat_zenith_angle	I Band Satellite Zenith Angle
i_sat_azimuth_angle	I Band Satellite Azimuth Angle
m_solar_zenith_angle	M Band Solar Zenith Angle
m_solar_azimuth_angle	M Band Solar Azimuth Angle
m_sat_zenith_angle	M Band Satellite Zenith Angle
m_sat_azimuth_angle	M Band Satellite Azimuth Angle
dnb_solar_zenith_angle	DNB Band Solar Zenith Angle
dnb_solar_azimuth_angle	DNB Band Solar Azimuth Angle
dnb_sat_zenith_angle	DNB Band Satellite Zenith Angle
dnb_sat_azimuth_angle	DNB Band Satellite Azimuth Angle
dnb_lunar_zenith_angle	DNB Band Lunar Zenith Angle
dnb_lunar_azimuth_angle	DNB Band Lunar Azimuth Angle
true_color	Ratio sharpened rayleigh corrected true color
false_color	Ratio sharpened rayleigh corrected false color

For reflectance/visible products a check is done to make sure that at least 10% of the swath is day time. Data is considered day time where solar zenith angle is less than 100 degrees.

4.2.1 Command Line Arguments

```
usage: polar2grid.sh -r viirs_l1b -w <writer> [-h] [--i-bands] [--m-bands]
                                              [--awips-true-color]
                                              [--awips-false-color]
```

VIIRS l1b Reader

--i-bands Add all I-band raw products to list of products

--m-bands Add all M-band raw products to list of products

--awips-true-color Add the True Color product to the list of products

--awips-false-color Add the False Color product to the list of products

Examples:

```
$POLAR2GRID_HOME/bin/polar2grid.sh -r viirs_l1b -w geotiff -h

polar2grid.sh -r viirs_l1b -w geotiff --list-products -f ../data/*.nc

polar2grid.sh -r viirs_l1b -w geotiff -p m01 -f /l1b/VJ102MOD.A2022257.1748.001.
↪2022258055009.nc l1b/VJ103MOD.A2022257.1748.001.2022258054957.nc

polar2grid.sh -r viirs_l1b -w hdf5 -g lcc_fit --add-geolocation -p i01 i02 -f␣
↪VNP02IMG.A2022257.1842*.nc VNP03MOD.A2022257.1842*.nc

polar2grid.sh -r viirs_l1b -w geotiff -p true_color false_color -f VNP02*.A2022257.
↪1842*.nc VNP03*.A2022257.1842*.nc
```

4.3 MODIS L1B Reader

The MODIS Reader operates on HDF4 Level 1B files from the Moderate Resolution Imaging Spectroradiometer (MODIS) instruments on the Aqua and Terra satellites. The reader is designed to work with files created by the IMAPP direct broadcast processing system (file naming conventions such as a1.17006.1855.1000m.hdf), but can support other types of L1B files, including the NASA archived files (file naming conventions such as MOD021KM.A2017004.1732.005.2017023210017.hdf). The reader can be specified to the `polar2grid.sh` script by using the reader name `modis` or `modis_l1b`.

This reader's default remapping algorithm is ewa for Elliptical Weighted Averaging resampling. The `--weight-delta-max` parameter set to 10 and the `--weight-distance-max` parameter set to 1.

It provides the following products:

Product Name	Description
vis01	Visible 1 Band
vis02	Visible 2 Band
vis03	Visible 3 Band
vis04	Visible 4 Band
vis05	Visible 5 Band
vis06	Visible 6 Band
vis07	Visible 7 Band

continues on next page

Table 4.3 – continued from previous page

Product Name	Description
vis26	Visible 26 Band
bt20	Brightness Temperature Band 20
bt21	Brightness Temperature Band 21
bt22	Brightness Temperature Band 22
bt23	Brightness Temperature Band 23
bt24	Brightness Temperature Band 24
bt25	Brightness Temperature Band 25
bt27	Brightness Temperature Band 27
bt28	Brightness Temperature Band 28
bt29	Brightness Temperature Band 29
bt30	Brightness Temperature Band 30
bt31	Brightness Temperature Band 31
bt32	Brightness Temperature Band 32
bt33	Brightness Temperature Band 33
bt34	Brightness Temperature Band 34
bt35	Brightness Temperature Band 35
bt36	Brightness Temperature Band 36
ir20	Radiance Band 20
ir21	Radiance Band 21
ir22	Radiance Band 22
ir23	Radiance Band 23
ir24	Radiance Band 24
ir25	Radiance Band 25
ir27	Radiance Band 27
ir28	Radiance Band 28
ir29	Radiance Band 29
ir30	Radiance Band 30
ir31	Radiance Band 31
ir32	Radiance Band 32
ir33	Radiance Band 33
ir34	Radiance Band 34
ir35	Radiance Band 35
ir36	Radiance Band 36
fog	Temperature Difference between BT31 and BT20
true_color	Ratio sharpened rayleigh corrected true color
false_color	Ratio sharpened rayleigh corrected false color

For reflectance/visible products a check is done to make sure that at least 10% of the swath is day time. Data is considered day time where solar zenith angle is less than 90 degrees.

4.3.1 Command Line Arguments

```
usage: polar2grid.sh -r modis_l1b -w <writer> [-h] [--ir-products]
                                              [--bt-products] [--vis-products]
                                              [--awips-true-color]
                                              [--awips-false-color]
```

MODIS L1B Reader

--ir-products Add IR products to list of products

--bt-products Add BT products to list of products

--vis-products Add Visible products to list of products

--awips-true-color Add individual CREFL corrected products to create the 'true_color' composite in AWIPS.

--awips-false-color Add individual CREFL corrected products to create the 'false_color' composite in AWIPS.

Examples:

```
polar2grid.sh -r modis_l1b -w geotiff -f <path to files>/<list of files>

polar2grid.sh -r modis_l1b -w geotiff -h

polar2grid.sh -r modis_l1b -w geotiff --list-products-all -f /data

polar2grid.sh -r modis_l1b -w geotiff -p true_color false_color -f ../l1b/a1.22261.
↪1857.250m.hdf ../l1b/a1.22261.1857.500m.hdf  ../l1b/a1.22261.1857.1000m.hdf ../l1b/
↪a1.22261.1857.geo.hdf

polar2grid.sh -r modis_l1b -w geotiff -p vis01 -f terra/t1.22061.1654.250m.hdf terra/
↪t1.22061.1654.geo.hdf

polar2grid.sh -r modis_l1b -w geotiff --grid-configs ${POLAR2GRID_HOME}/grid_configs/
↪grid_example.yaml -g my_latlon -f ../l1b/a1.17006.1855.1000m.hdf ../l1b/a1.17006.
↪1855.geo.hdf

polar2grid.sh -r modis_l1b -w awips_tiled --awips-true-color --awips-false-color --
↪sector-id LCC -g lcc_conus_300 --letters --compress --grid-coverage=.05 -f MOD021KM.
↪A2017004.1732*.hdf MOD02HKM.A2017004.1732*.hdf MOD02QKM.A2017004.1732*.hdf MOD03.
↪A2017004.1732*.hdf

polar2grid.sh -r modis_l1b -w hdf5 --bt-products --add-geolocation  -f /data/MOD*.hdf

polar2grid.sh -r modis_l1b -w hdf5 -g wgs84_fit_250 -p vis01 vis02 -f /data/rad/
↪MOD02QKM.*.hdf /data/geo/MOD03.*.hdf
```

4.3.2 Product Explanation

True Color

The MODIS Level 1B "true_color" composite produced by Polar2Grid provides a view of the Earth as the human eye would see it; or as close as we can come to with satellite data and the channels we have available. This means things like trees and grass are green, water is blue, deserts are red/brown, and clouds are white. The True Color GeoTIFF 24 bit image is an *RGB (Red, Green, Blue) image* consisting of a combination of Red: MODIS Band 1, Green: MODIS Band 4, and Blue: MODIS Band 3 reflectance channels. Each channel goes through a series of adjustments to produce the final high quality image output by Polar2Grid.

Creation of true color RGBs includes the following steps:

- Atmospheric *Rayleigh Scattering Correction* of the RGB visible reflectances.
- Creation of reflectances through *division by the cosine of the soloar zenith angle*.
- Combining the 3 channels into a 24 bit output file.
- *Sharpening of the image* to full 250m resolution if the MODIS 250m Band 1 is provided as input.
- Application of a *non-linear enhancement*.

See the *Creating MODIS AWIPS Compatible Files* example to see how Polar2Grid can be used to make this product as a set of AWIPS compatible NetCDF files.

False Color

A false color image is any combination of 3 bands outside of those used to create a "true color" image (see above). These combinations can be used to highlight features in the observations that may not be easily identified in individual band imagery. Polar2Grid can readily create a preconfigured legacy false color (product false_color) GeoTIFF 24 bit image that consists of a combination *RGB (Red, Green, Blue) image* using Red: MODIS Band 7 (2.21 μm), Green: MODIS Band 2 (.86 μm) and Blue: MODIS Band 1 (.65 μm). This band combination is very effective at distinguishing land/water boundaries as well as burn scars.

Creation of MODIS legacy false color RGBs includes the following steps:

- Atmospheric *Rayleigh Scattering Correction* of the RGB visible reflectances.
- Creation of reflectances through *division by the cosine of the soloar zenith angle*.
- Combining the 3 channels into a 24 bit output file.
- *Sharpening of the image* to full 250m resolution if the MODIS 250m Band 2 is provided as input.
- Application of a *non-linear enhancement*.

See the *Creating MODIS AWIPS Compatible Files* example to see how Polar2Grid can be used to make this product as AWIPS compatible NetCDF files.

```
polar2grid.sh -r modis_l1b -w geotiff -p true_color false_color -f /aqua/a1.17006.1855.250m.hdf
/aqua/a1.17006.1855.500m.hdf /aqua/a1.17006.1855.geo.hdf
```

4.4 AVHRR Reader

The AVHRR reader is for reading AAPP L1B files for the AVHRR instrument.

These files are a custom binary format. The reader can be specified with the `polar2grid.sh` command using the `avhrr` or `avhrr_l1b_aapp` reader name.

This reader's default resampling algorithm is `ewa` for Elliptical Weighted Averaging resampling. The `--weight-delta-max` option is set to 10 and `--weight-distance-max` is set to 1.

The AVHRR reader provides the following products:

Product Name	Description
band1_vis	Band 1 Visible
band2_vis	Band 2 Visible
band3a_vis	Band 3A Visible
band3b_bt	Band 3B Brightness Temperature
band4_vis	Band 4 Brightness Temperature
band5_vis	Band 5 Brightness Temperature

4.4.1 Command Line Arguments

```
usage: polar2grid.sh -r avhrr_l1b_aapp -w <writer> [-h]
```

4.4.2 Execution Examples

```
polar2grid.sh -r avhrr_l1b_aapp -w awips_tiled --list-products -f /l1b/

polar2grid.sh -r avhrr_l1b_aapp -w geotiff -f ../input/hrpt_noaa19_20220917_1236_
↪70145.l1b

polar2grid.sh -r avhrr -w geotiff -p band3a_vis band4_bt -f /data/hrpt_M03*.l1b

polar2grid.sh -r avhrr -w awips_tiled -p band3b_bt -g lcc_conus_1km --sector-id LCC --
↪letters --compress -f hrpt_noaa18_20220918_1708_89324.l1b

polar2grid.sh -r avhrr_l1b_aapp -w awips_tiled --num-workers 6 --grid-coverage .002 -
↪g polar_alaska_1km --sector-id Polar --letters --compress -f /avhrr

polar2grid.sh -r avhrr -w hdf5 --add-geolocation --grid-configs /home/avhrr/grids/
↪local_grid.yaml -g my_grid  -f ../input/hrpt_M01*.l1b

polar2grid.sh -r avhrr -w binary --num-workers 8 -p band1_vis band4_bt -g lcc_eu -f /
↪data/avhrr/metoba
```

4.5 AMSR2 L1B Reader

AMSR2 L1B files contain various parameters from the GCOM-W1 AMSR2 instrument.

This reader can be used by specifying the reader name `amsr2_l1b` to the `polar2grid.sh` script. Supported files usually have the following naming scheme:

```
GW1AM2_201607201808_128A_L1DLBTBR_1110110.h5
```

This reader's default remapping algorithm is `nearest` for nearest neighbor resampling due to the instruments scan pattern and swath shape. The `--distance_upper_bound` flag defaults to 12.

Currently this reader provides only the following datasets:

Product Name	Description
btemp_36.5v	Brightness Temperature 36.5GHz Polarization Vertical
btemp_36.5h	Brightness Temperature 36.5GHz Polarization Horizontal
btemp_89.0av	Brightness Temperature 89.0GHz A Polarization Vertical
btemp_89.0ah	Brightness Temperature 89.0GHz A Polarization Horizontal
btemp_89.0bv	Brightness Temperature 89.0GHz B Polarization Vertical
btemp_89.0bh	Brightness Temperature 89.0GHz B Polarization Horizontal

4.5.1 Special AMSR2 Naval Research Lab (NRL) PNG Scaling

A common use case for the AMSR2 L1B reader is to generate PNG images similar to those generated by the U.S. Naval Research Lab (NRL) with a colormap and coastlines. This requires using an alternate non-default scaling configuration provided in the tarball. It can be used by providing the `--extra-config-path $POLAR2GRID_HOME/example_enhancements/amsr2_png` flag when generating AMSR2 L1B GeoTIFFs. This allows the scaling provided in the `$POLAR2GRID_HOME/example_enhancements/amsr2_png/enhancements/generic.yaml` file to be used. Once this rescaling has been done, colormap files can be found in `$POLAR2GRID_HOME/colormaps` which can then be applied using the the *add_colormap.sh* script.

See the example section *Creating AMSR2 Reprojections* for more information on generating these NRL-like PNGs.

4.5.2 Command Line Arguments

```
usage: polar2grid.sh -r amsr2_l1b -w <writer> [-h]
```

Examples:

```
polar2grid.sh -r amsr2_l1b -w geotiff --list-products-all -f ../data/*L1DLBTBR*.h5

polar2grid.sh -r amsr2_l1b -w geotiff  -f <path to files>/<AMSR2 Level 1B filename>

polar2grid.sh -r amsr2_l1b -w geotiff -g lcc_fit --fill-value 0 -f ../data/GW1AM2_
↪202209121053_051D_L1DLBTBR_1110110.h5

polar2grid.sh -r amsr2_l1b -w geotiff --extra-config-path $POLAR2GRID_HOME/example_
↪enhancements/amsr2_png --fill-value 0 -f ../gcom_data/

polar2grid.sh -r amsr2_l1b -w awips_tiled --list-products -f /amsr2/GW1AM2_
↪202209120729_019D_L1DLBTBR_2220220.h5
```

(continues on next page)

(continued from previous page)

```
polar2grid.sh -r amsr2_l1b -w awips_tiled -g lcc_conus_1km -p btemp_36.5h btemp_89.
↪0av --sector-id LCC --letters --compress -f GW1AM2_201607191903_137A_L1DLBTBR_
↪1110110.h5
```

4.6 NUCAPS Reader

The NUCAPS Reader supports reading NUCAPS Retrieval files. This reader can be used by specifying the name `nucaps` to the `polar2grid.sh` script. Files for this reader should follow the naming scheme:

NUCAPS-EDR_v1r0_npp_s201603011158009_e201603011158307_c201603011222270.nc

This reader's default resampling algorithm is `ewa` for Elliptical Weighted Averaging resampling. The `--weight-delta-max` parameter is set to 40 and the `--weight-distance-max` parameter is set to 1.

This reader can provide the following products:

Product Name	Description
Temperature_Xmb	Temperature at various pressure levels
H2O_MR_Xmb	Water Vapor Mixing Ratio at various pressure levels
Topography	Height at surface
Surface_Pressure	Pressure at surface
Skin_Temperature	Skin Temperature

Pressure based datasets are specified by the pressure level desired in millibars. The value used in the product name is listed in the table below for each corresponding pressure value:

Pressure Value	Name Value
0.016	0.016
0.038	0.038
0.077	0.077
0.137	0.137
0.224	0.224
0.345	0.345
0.506	0.506
0.714	0.714
0.975	0.975
1.297	1.297
1.687	1.687
2.153	2.153
2.701	2.701
3.340	3.340
4.077	4.077
4.920	4.920
5.878	6
6.957	7
8.165	8
9.512	10
11.004	11

continues on next page

Table 4.4 – continued from previous page

Pressure Value	Name Value
12.649	13
14.456	14
16.432	16
18.585	19
20.922	21
23.453	23
26.183	26
29.121	29
32.274	32
35.651	36
39.257	39
43.100	43
47.188	47
51.528	52
56.126	56
60.989	61
66.125	66
71.540	72
77.240	77
83.231	83
89.520	90
96.114	96
103.017	103
110.237	110
117.777	118
125.646	126
133.846	134
142.385	142
151.266	151
160.496	160
170.078	170
180.018	180
190.320	190
200.989	201
212.028	212
223.441	223
235.234	235
247.408	247
259.969	260
272.919	273
286.262	286
300.000	300
314.137	314
328.675	329
343.618	344
358.966	359
374.724	375
390.893	391
407.474	407

continues on next page

Table 4.4 – continued from previous page

Pressure Value	Name Value
424.470	424
441.882	442
459.712	460
477.961	478
496.630	497
515.720	516
535.232	535
555.167	555
575.525	576
596.306	596
617.511	618
639.140	639
661.192	661
683.667	684
706.565	707
729.886	730
753.628	754
777.790	778
802.371	802
827.371	827
852.788	853
878.620	879
904.866	905
931.524	932
958.591	959
986.067	986
1013.950	1014
1042.230	1042
1070.920	1071
1100.000	1100

4.6.1 Command Line Arguments

```
usage: polar2grid.sh -r nucaps -w <writer> [-h] [--no-mask-surface]
                                           [--no-mask-quality]
                                           [--pressure-levels PRESSURE_LEVELS␣
↪PRESSURE_LEVELS]
```

NUCAPS Reader

--no-mask-surface Don't mask pressure based datasets that go below the surface pressure

Default: True

--no-mask-quality Don't mask datasets based on Quality Flag

Default: True

NUCAPS Product Filters

--pressure-levels Min and max pressure value to make available

Default: (110.0, 987.0)

4.7 MIRS Reader

The MiRS frontend extracts data from files created by the Community Satellite Processing Package (CSPP) direct broadcast version of the NOAA/STAR Microwave integrated Retrieval System (MiRS). The software supports the creation of atmospheric and surface parameters from ATMS, AMSU-A, and MHS microwave sensor data. For more information on this product, please visit the CSPP LEO website: *https://cimss.ssec.wisc.edu/cspp/*.

When executed on Advanced Technology Microwave Sounder (ATMS) MiRS product files, a limb correction algorithm is applied for brightness temperatures reprojections for each of the 22 spectral bands. The correction software was provided by Kexin Zhang of NOAA STAR, and is applied as part of the MIRS ATMS Polar2Grid execution.

This reader's default resampling algorithm is `ewa` for Elliptical Weighted Averaging resampling. The `--weight-delta-max` option is set to 100 and the `--weight-distance-max` option is set to 1.

The ACSPO output product format is NetCDF4. The frontend can be specified with the *polar2grid.sh* command using the *mirs* frontend name.The frontend offers the following products:

Product Name	Description
rain_rate	Rain Rate
sea_ice	Sea Ice in percent
snow_cover	Snow Cover
tpw	Total Precipitable Water
swe	Snow Water Equivalence
clw	Cloud Liquid Water
sfr	Snow Fall Rate
surface_type	Surface Type
tskin	Skin Temperature
btemp_X	Brightness Temperature for channel X (see below)

For specific brightness temperature band products, use the `btemp_X` option, where `X` is a combination of the microwave frequency (integer) and polarization of the channel. If there is more than one channel at that frequency and polarization a "count" number is added to the end. To create output files of all available bands, use the `--bt-channels` option.

As an example, the ATMS band options are:

Table 4.5: ATMS Brightness Temperature Channels

Band Number	Frequency (GHz)	Polarization	Polar2Grid Dataset Name
1	23.79	V	btemp_23v
2	31.40	V	btemp_31v
3	50.30	H	btemp_50h
4	51.76	H	btemp_51h
5	52.80	H	btemp_52h
6	53.59±0.115	H	btemp_53h
7	54.40	H	btemp_54h1
8	54.94	H	btemp_54h2
9	55.50	H	btemp_55h
10	57.29	H	btemp_57h1
11	57.29±2.17	H	btemp_57h2
12	57.29±0.3222±0.048	H	btemp_57h3
13	57.29±0.3222±0.022	H	btemp_57h4
14	57.29±0.3222±0.010	H	btemp_57h5
15	57.29±0.3222±0.0045	H	btemp_57h6
16	88.20	V	btemp_88v
17	165.50	H	btemp_165h
18	183.31±7.0	H	btemp_183h1
19	183.31±4.5	H	btemp_183h2
20	183.31±3.0	H	btemp_183h3
21	183.31±1.8	H	btemp_183h4
22	183.31±1.0	H	btemp_183h5

4.7.1 Command Line Arguments

```
usage: polar2grid.sh -r mirs -w <writer> [-h] [--bt-channels]
```

VIIRS SDR Reader

--bt-channels Add all bands to list of products

4.7.2 Execution Examples

```
polar2grid.sh -r mirs -w geotiff --list-products -f /atms/NPR-MIRS-IMG_v11r8_n20_
↪s202208250741413_e202208250753249_c202208250817490.nc

polar2grid.sh -r mirs -w geotiff --list-products-all -f /atms/NPR-MIRS-IMG_v11r8_npp_
↪s202208250653413_e202208250704529_c202208250730310.nc

polar2grid.sh -r mirs -w geotiff -p btemp_88v btemp_183h3 swe -g lcc_fit -f /atms/

polar2grid.sh -r mirs -w awips_tiled --num-workers 4 --grid-coverage 0 -g merc_
↪pacific_1km --sector-id Pacific --letters --compress -p swe tpw sea_ice rain_rate␣
↪btemp_89v1 -f /noaa19/NPR-MIRS-IMG_v11r8_n19_s202208250310331_e202208250314143_
↪c202208251718490.nc

polar2grid.sh -r mirs -w awips_tiled --bt-channels -g lcc_conus_750 --sector-id LCC --
↪letters --compress -f ../input/NPR-MIRS-IMG_v11r1_NPP_s201611111032500_
↪e201611111044016_c201611111121100.nc

polar2grid.sh -r mirs -w hdf5 --add-geolocation --dtype float32 -f ../metopc/NPR-MIRS-
↪IMG_v11r8_ma3_s202208251542209_e202208251554261_c202208251742030.nc
```

4.8 ACSPO SST Reader

The ACSPO reader is for reading files created by the NOAA Community Satellite Processing Package (CSPP) Advanced Clear-Sky Processor for Oceans (ACSPO) system software. The ACSPO reader supports product files created from VIIRS, MODIS and AVHRR imaging sensors. For more information on this product, please visit the CSPP LEO website: *https://cimss.ssec.wisc.edu/cspp/*.

The ACSPO output product format is NetCDF4. The frontend can be specified with the `polar2grid.sh` command using the `acspo` frontend name. The ACSPO frontend provides the following products:

Product Name	Description
sst	Sea Surface Temperature

4.8.1 Command Line Arguments

```
usage: polar2grid.sh -r acspo -w <writer> [-h]
```

Examples:

```
polar2grid.sh -r aspo -w geotiff -h

polar2grid.sh -r acspo -w geotiff --list-products -f /noaa20/20220526060927-CSPP-L2P_
↪GHRSST-SSTskin-VIIRS_N20-ACSPO_V2.80-v02.0-fv01.0.nc

polar2grid.sh -r acspo -w geotiff --grid-coverage=0.0 -f /aqua/20220524205044-CSPP-
↪L2P_GHRSST-SSTskin-MODIS_A-ACSPO_V2.80-v02.0-fv01.0.nc

mpolar2grid.sh -r acspo -w hdf5 -p sst sea_ice_fraction --compress gzip --add-
↪geolocation -g lcc_fit --grid-coverage=.02 -f /metopc/20220803024121-CSPP-L2P_
↪GHRSST-SSTskin-AVHRRF_MC*.nc
```

(continues on next page)

(continued from previous page)

```
polar2grid.sh -r acspo -w awips_tiled --num-workers 4 --grid-coverage 0 -g lcc_conus_
↪750 --sector-id LCC --letters --compress -f 20220526060927-CSPP-L2P_GHRSST-SSTskin-
↪VIIRS_N20*.nc

polar2grid.sh -r acspo -w awips_tiled -g merc_pacific_300 --sector-id Pacific --
↪letters --compress -f *CSPP-L2P_GHRSST-SSTskin-VIIRS_NPP-ACSPO*.nc
```

4.9 CLAVR-x Cloud Product Reader

The CLAVR-x reader is for reading files created by the Community Satellite Processing Package (CSPP) Clouds from AVHRR Extended (CLAVR-x) processing system software. The CLAVR-x reader supports CSPP CLAVR-x product files created from VIIRS, MODIS and AVHRR imaging sensors in the native HDF4 format. For more information on this product, please visit the CSPP LEO website: *https://cimss.ssec.wisc.edu/cspp/*.

The reader can be specified with the `polar2grid.sh` command using the `clavrx` reader name. The CLAVR-x reader provides the following products, which include support for the VIIRS Day/Night Band Lunar Reflectance:

Product Name	Description
cloud_type	Cloud Type
cld_height_acha	Cloud Top Height (m)
cld_temp_acha	Cloud Top Temperature (K)
cld_emiss_acha	Cloud Emissivity
cld_opd_dcomp	Cloud Optical Depth Daytime
cld_opd_nlcomp	Cloud Optical Depth Nighttime
cld_reff_dcomp	Cloud Effective Radius Daytime (micron)
cld_reff_nlomp	Cloud Effective Radius Nighttime (micron)
cloud_phase	Cloud Phase (5 categories)
rain_rate	Rain Rate (mm/hr)
refl_lunar_dnb_nom	Lunar Reflectance (VIIRS DNB nighttime only)

4.9.1 Command Line Arguments

```
usage: polar2grid.sh -r clavrx -w <writer> [-h]
```

Examples:

```
polar2grid.sh -r clavrx -w geotiff -h

polar2grid.sh -r clavrx -w awips_tiled --sector-id LCC --list-products -f clavrx_npp_
↪d20220902_t0742031_e0756141_b56210.level2.hdf

polar2grid.sh -r clavrx -w geotiff -p cld_height_acha cloud_phase cloud_type -f␣
↪noaa20/clavrx_j01*.hdf

polar2grid.sh -r clavrx -w hdf5 --grid-coverage 0.002 -p cld_opd_nlcomp cld_reff_
↪nlcomp refl_lunar_dnb_nom -f snpp/night/clavrx_npp*.hdf

polar2grid.sh -r clavrx -w binary -f clavrx_a1.22245.0759.1000m.level2.hdf
```

(continues on next page)

(continued from previous page)

```
polar2grid.sh -r clavrx -w awips_tiled --num-workers 6 -g lcc_conus_300 --sector-id␣
↪LCC --letters --compress --grid-coverage 0.002 -p cld_temp_acha cld_height_acha␣
↪cloud_phase cld_opd_dcomp -f noaa19/clavrx_hrpt_noaa19_*.hdf
```

4.10 VIIRS EDR Active Fires Reader

The VIIRS EDR Active Fires reader operates on CSPP NetCDF I-Band (AFIMG) Resolution or M-Band Resolution (AFMOD) Environmental Data Record files.

Files supported usually have the following naming schemes:

AFIMG_j01_d20221006_t2101052_e2102297_b25304_c20221006214545032016_cspp_dev.nc and/or,
AFMOD_npp_d20221006_t2017005_e2018247_b56701_c20221006205259096916_cspp_dev.nc

For more information about the this CSPP product, please visit the CSPP LEO website: *https://cimss.ssec.wisc.edu/cspp/*.

This reader's default resampling algorithm `--method` is `nearest` for Nearest Neighbor resampling. The frontend can be specified with the `polar2grid.sh` command using the `viirs_edr_active_fires` frontend name. The VIIRS Active Fire frontend provides the following products:

Product Name	Description
confidence_cat	Fire Confidence Category (AFIMG Resolution Only)
T4	I-Band 4 Temperature (AFIMG Resolution Only)
power	Fire Radiative Power
confidence_pct	Fire Confidence Percentage (AFMOD Resolution Only)
T13	M-Band 13 Temperature (AFMOD Resolution Only)

4.10.1 Command Line Arguments

```
usage: polar2grid.sh -r viirs_edr_active_fires -w <writer> [-h]
```

Some output GeoTIFF fire products are color enhanced:

AFIMG

confidence_cat - Low (Yellow), Nominal (Orange), High (Red)

power - 1 - 250 and above (MW) Yellow->Red

AFMOD

confidence_pct - 1-100% Yellow->Red

power - 1 - 250 and above (MW) Yellow->Red

Examples:

```
$POLAR2GRID_HOME/bin/polar2grid.sh -r viirs_edr_active_fires -w geotiff -h

polar2grid.sh -r viirs_edr_active_fires -w geotiff --list-products -f ../active_fire_
↪edr/AFIMG*.nc

polar2grid.sh -r viirs_edr_active_fires -w geotiff --list-products -f ../active_fire_
↪edr/AFMOD*.nc
```

(continues on next page)

(continued from previous page)

```
polar2grid.sh -r viirs_edr_active_fires -w geotiff -p confidence_cat T4 img_edr/
↪AFIMG*.nc

polar2grid.sh -r viirs_edr_active_fires -w geotiff -g lcc_aus -p confidence_pct T13  -
↪f AFMOD_j01_d20191120_t1513353_e1514581_b10389_c20191121192444396115_cspp_dev.nc
```

NOTE: The active fire images can be overlaid onto another GeoTIFF. See *Overlay GeoTIFF Images* for instructions.

4.11 MERSI-2 L1B Reader

The FY3-D MERSI2 Level 1B reader operates on Level 1B (L1B) HDF5 files.

The files come in four varieties; band data and geolocation data, both at 250m and 1000m resolution. Files usually have the following naming scheme:

tf{start_time:%Y%j%H%M%S}.{platform_shortname}-{trans_band:1s}_MERSI_1000M_L1B.{ext}

This reader's default resampling algorithm is `ewa` for Elliptical Weighted Averaging resampling. The `--weight-delta-max` parameter is set to 40 and the `--weight-distance-max` parameter is set to 1.

The frontend can be specified with the `polar2grid.sh` command using the `mersi2_l1b` frontend name. The MERSI2 frontend provides the following products:

Product Name	Description	Central Wavelength (um)
1	Channel 1 Reflectance Band	0.47
2	Channel 2 Reflectance Band	0.55
3	Channel 3 Reflectance Band	0.65
4	Channel 4 Reflectance Band	0.865
5	Channel 5 Reflectance Band	1.38
6	Channel 6 Reflectance Band	1.64
7	Channel 7 Reflectance Band	2.13
8	Channel 8 Reflectance Band	0.412
9	Channel 9 Reflectance Band	0.443
10	Channel 10 Reflectance Band	0.490
11	Channel 11 Reflectance Band	0.555
12	Channel 12 Reflectance Band	0.670
13	Channel 13 Reflectance Band	0.709
14	Channel 14 Reflectance Band	0.746
15	Channel 15 Reflectance Band	0.865
16	Channel 16 Reflectance Band	0.905
17	Channel 17 Reflectance Band	0.936
18	Channel 18 Reflectance Band	0.940
19	Channel 19 Reflectance Band	1.24
20	Channel 20 Brightness Temperature Band	3.80
21	Channel 21 Brightness Temperature Band	4.05
22	Channel 22 Brightness Temperature Band	7.20
23	Channel 23 Brightness Temperature Band	8.55
24	Channel 24 Brightness Temperature Band	10.8
25	Channel 25 Brightness Temperature Band	12.0
true_color	Rayleigh corrected true color RGB	N/A
false_color	False color RGB (bands 7, 4, 3)	N/A
natural_color	Natural color RGB (bands 6, 4, 3)	N/A

4.11.1 Command Line Arguments

```
usage: polar2grid.sh -r mersi2_l1b -w <writer> [-h]
```

Examples:

```
$POLAR2GRID_HOME/bin/polar2grid.sh -r mersi2_l1b -w geotiff -h

polar2grid.sh -r mersi2_l1b -w geotiff --list-products -f tf2019259173245.FY3D-X_
↪MERSI*.HDF

polar2grid.sh -r mersi2_l1b -w geotiff -p 1 2 3 4 6 7 20 25 -f tf2019233172521.FY3D-X_
↪MERSI_0250M_L1B.HDF tf2019233172521.FY3D-X_MERSI_1000M_L1B.HDF tf2019233172521.FY3D-
↪X_MERSI_GEOQK_L1B.HDF tf2019233172521.FY3D-X_MERSI_GEO1K_L1B.HDF

polar2grid.sh -r mersi2_l1b -w geotiff -p true_color false_color -g lcc_fit -f ../
↪mersi/tf2019259173245.FY3D-X_MERSI*.HDF

polar2grid.sh -r mersi2_l1b -w hdf5 -p 20 21 22 23 24 25  --grid-configs ${HOME}/my_
↪grid.yaml -g shanghai seoul -f ../data/*.HDF

polar2grid.sh -r mersi2_l1b -w binary --sza-threshold=90  -p 1 2 3 4 6 7 20 25 -f␣
↪tf2019226095418.FY3D-X_MERSI_*.HDF
```

(continues on next page)

(continued from previous page)

CHAPTER

FIVE

REMAPPING

Remapping is the process of mapping satellite data to a uniform grid. Mapping data to a uniform grid makes it easier to view, manipulate, and store the data. Some instrument data is provided to the user already gridded (ex. VIIRS EDR Flood, ABI L1B data) and others are not (ex. VIIRS SDR or older GOES satellites).

In Polar2Grid it is possible to perform the gridding (reprojecting) process for ungridded data or to re-project already gridded data. Mapping input data in order to create a high quality image can be a complicated process. There are different techniques that can be used to create an output image based on what grid (projection) is chosen and what algorithm is used to map input pixel to output pixel. Polar2Grid offers various options that are described below. Defaults are also configured to provide a good result without any customization necessary.

5.1 Native Resampling

Native resampling is a special type of resampling that keeps input data in its original projection, but replicates or averages data when necessary to make other processing in Polar2Grid easier. Native resampling is the default for all data that is already gridded (ABI, AHI, etc) or when a native grid is specified by the user on the command line (`-g MIN`). It can also be specified on the command line by using `--method native`. See the Command Line Arguments section below for more details and the options available.

5.2 Elliptical Weighted Averaging Resampling

Elliptical Weighted Averaging (EWA) resampling is the default resampling method for a lot of scan-based polar-orbiting instrument data. This method uses the size of each instrument scan to determine a weight for each pixel. All input pixels that map to output pixels are weighted and averaged. This helps produce an image that is typically higher quality than those produced by nearest neighbor. It fits an ellipse to the data in the two axes based upon the `--weight-delta-max` and the `--weight-distance-max` options and then filters the texture with a Gaussian filter function. It can be specified on the command line by using `--method ewa`.

5.3 Nearest Neighbor Resampling

Nearest neighbor resampling is the most basic form of resampling when gridding data to another grid. This type of resampling will find the nearest valid input pixel for each pixel in the output image. If a valid pixel can't be found near a location then an invalid (transparent) pixel is put in its place. Controlling this search distance and other options are described below in the Command Line Arguments section. Nearest neighbor resampling can be specified on the command line with `--method nearest` and is the default when non-native grids are specified to the command line (`-g my_grid`) for gridded data or if polar-orbiting instrument data is not scan-based (required for EWA).

Note that nearest neighbor resampling can cache intermediate calculations to files on disk when the same grid is used. For example, the calculations required to resample ABI L1B data to the same output grid for each time step are the same. If a directory is specified with the `--cache-dir` command line flag, this can greatly improve performance. This has no benefit for polar-orbiting swath-based data.

5.4 Grids

Polar2Grid uses the idea of "grids" to define the output geographic location that images will be remapped to. Grids are also known as "areas" in the SatPy library. These terms may be used interchangeably through this documentation, especially in low-level parts.

Polar2Grid uses grids defined by a PROJ.4 projection specification. Other parameters that define a grid like its width and height can be determined dynamically during this step. A grid is defined by the following parameters:

- Grid Name
- PROJ.4 String (either lat/lon or metered projection space)
- Width (number of pixels in the X direction)
- Height (number of pixels in the Y direction)
- Cell Width (pixel size in the X direction in grid units)
- Cell Height (pixel size in the Y direction in grid units)
- X Origin (upper-left X coordinate in grid units)
- Y Origin (upper-left Y coordinate in grid units)

Polar2Grid supports static and dynamic grids. Grids are static if they have all of the above attributes defined. Grids are dynamic if some of the attributes are not defined. These attributes are then computed at run time based on the data being remapped. Only width/height and x/y origin can be unspecified in dynamic grids. SatPy areas are also supported by Polar2Grid, but must be specified in SatPy's typical "areas.yaml" file.

For information on defining your own custom grids see the *Custom Grid* documentation.

5.5 Remapping and Grid Command Line Arguments

```
usage: polar2grid.sh -r <reader> -w <writer> [-h]
                                             [--method {ewa,native,nearest}]
                                             [-g [GRIDS [GRIDS ...]]]
                                             [--weight-delta-max WEIGHT_DELTA_MAX]
                                             [--weight-distance-max WEIGHT_DISTANCE_
↪MAX]
                                             [--maximum-weight-mode]
                                             [--rows-per-scan ROWS_PER_SCAN]
                                             [--grid-coverage GRID_COVERAGE]
                                             [--cache-dir CACHE_DIR]
                                             [--grid-configs GRID_CONFIGS [GRID_
↪CONFIGS ...]]
                                             [--antimeridian-mode {modify_extents,
↪modify_crs,global_extents}]
                                             [--radius-of-influence RADIUS_OF_
↪INFLUENCE]
```

5.5.1 Resampling

--method Possible choices: ewa, native, nearest

resampling algorithm to use (default: <sensor specific>)

-g, --grids Area definition to resample to. Empty means no resampling (default: "wgs84_fit" for non-native resampling)

--weight-delta-max Maximum distance in grid cells over which to distribute an input swath pixel (–method "ewa"). This is equivalent to the old "–fornav-D" flag. Default is 10.0.

--weight-distance-max Distance in grid cell units at which to apply a minimum weight. (–method "ewa"). This is equivalent to the old "–fornav-d" flag. Default is 1.0.

--maximum-weight-mode Use maximum weight mode (–method "ewa"). Default is off.

--rows-per-scan Number of data rows making up one instrument scan. (–method "ewa"). Defaults to value extracted from reader.

--grid-coverage Fraction of target grid that must contain data to continue processing product.

Default: 0.1

--cache-dir Directory to store resampling intermediate results between executions. Not used with native resampling or resampling of ungridded or swath data.

--grid-configs Specify additional grid configuration files. (.conf for legacy CSV grids, .yaml for SatPy-style areas)

Default: ()

--antimeridian-mode Possible choices: modify_extents, modify_crs, global_extents

Behavior when dynamic grids are converted to 'frozen' grids and data crosses the anti-meridian. Defaults to 'modify_crs' where the prime meridian is shifted 180 degrees to make the result one contiguous coordinate space. 'modify_extents' will attempt to surround the data but will often cause artifacts over the antimeridian. 'global_extents' will force the X extents to -180 and 180 to create one large grid. This currently only affects lon/lat projections.

Default: "modify_crs"

--radius-of-influence Specify radius to search for valid input pixels for nearest neighbor resampling (–method "nearest"). Value is in geocentric meters regardless of input or output projection. By default this will be estimated based on input and output projection and pixel size.

CHAPTER

SIX

WRITERS

Writers are the final step in the Polar2Grid processing chain. They take gridded data, scale it to fit in an output format, and write the data to one or more output files. These files can then be provided to a visualization tool that is optimized for viewing the data.

6.1 AWIPS Tiled Writer

The AWIPS Tiled writer is used to create AWIPS compatible tiled NetCDF4 files.

The Advanced Weather Interactive Processing System (AWIPS) is a program used by the United States National Weather Service (NWS) and others to display and analyze data relevant to meteorology. Sectorized Cloud and Moisture Imagery (SCMI) is a NetCDF4 format accepted by AWIPS to store one image broken up into one or more "tiles". This format has been used to support additional products over time and so this writer is now called "awips_tiled" to refer to the generic usse of these files. Once AWIPS is configured for specific products the AWIPS Tiled writer can be used to provide compatible products to the system. The files created by this writer are compatible with AWIPS II.

The writer takes remapped image data and creates an AWIPS-compatible NetCDF4 file. The writer and the AWIPS client may need to be configured to make things appear the way the user wants in the AWIPS client. The writer can only produce files for datasets mapped to areas with specific projections:

- Lambert Conformal Conic (+*proj=lcc*)
- Geostationary (+*proj=geos*)
- Mercator (+*proj=merc*)
- Polar Stereographic (+*proj=stere*)

This is a limitation of the AWIPS client and not of the writer.

6.1.1 Numbered versus Lettered Grids

By default the writer will save tiles by number starting with '1' representing the upper-left image tile. Tile numbers then increase along the column and then on to the next row.

By specifying `--letters` on the command line, tiles can be designated with a letter. Lettered grids or sectors are preconfigured in the writer configuration file (`awips_tiled.yaml` in SatPy). The lettered tile locations are static and will not change with the data being written to them. Each lettered tile is split in to a certain number of subtiles (*–letter-subtiles*), default 2 rows by 2 columns. Lettered tiles are meant to make it easier for receiving AWIPS clients/stations to filter what tiles they receive; saving time, bandwidth, and space.

Tiles (numbered or lettered) not containing any valid data are not created.

> **Warning:** The writer does not default to using any grid. Therefore, it is recommended to specify one or more grids for remapping by using the *-g* flag.

For more detailed information on templates and other options for this writer see the Satpy documentation here.

6.1.2 Command Line Arguments

```
usage: |script| -r <reader> -w awips_tiled [-h] [--compress]
                                           [--output-filename FILENAME]
                                           [--use-end-time]
                                           [--use-sector-reference]
                                           [--tiles TILE_COUNT TILE_COUNT]
                                           [--tile-size TILE_SIZE TILE_SIZE]
                                           [--letters]
                                           [--letter-subtiles LETTER_SUBTILES LETTER_
↪SUBTILES]
                                           [--source-name SOURCE_NAME]
                                           [--sector-id SECTOR_ID]
                                           [--template TEMPLATE]
                                           [--environment-prefix ENVIRONMENT_PREFIX]
                                           [--check-categories]
```

AWIPS Tiled Writer

--compress zlib compress each netcdf file

Default: False

--output-filename custom file pattern to save dataset to

--use-end-time use end_time metadata inplace of start_time (useful for multi-day composites)

Default: False

--use-sector-reference use the lettered sector location as reference and shift data to match tile pixel locations. Useful when tiles will be updated in future executions. By default the sector tiles are shifted to match the data location. Maximum shift is 0.5 pixels.

Default: False

--tiles Number of tiles to produce in Y (rows) and X (cols) direction respectively

Default: [1, 1]

--tile-size Specify how many pixels are in each tile (overrides '–tiles')

--letters Create tiles from a static letter-based grid based on the product projection

Default: False

--letter-subtiles Specify number of subtiles in each lettered tile: 'row col'

Default: (2, 2)

--source-name specify processing source name used in attributes and filename

Default: "SSEC"

--sector-id specify name for sector/region used in attributes and filename (example 'LCC')

--template specify name for pre-configured template used to determine output file structure and formatting.

Default: "polar"

--environment-prefix Force value for templates that support an 'environment_prefix' in the filename (default 'OR').

Default: "DR"

--check-categories, --no-check-categories Specify whether category/flag products should be checked for empty tiles. By default (True), tiles consisting entirely of 'missing' pixels will not be written. For category products this may not be desired as the missing or invalid value is something users are interested in. (default: True)

Default: True

6.1.3 Lettered Sectors

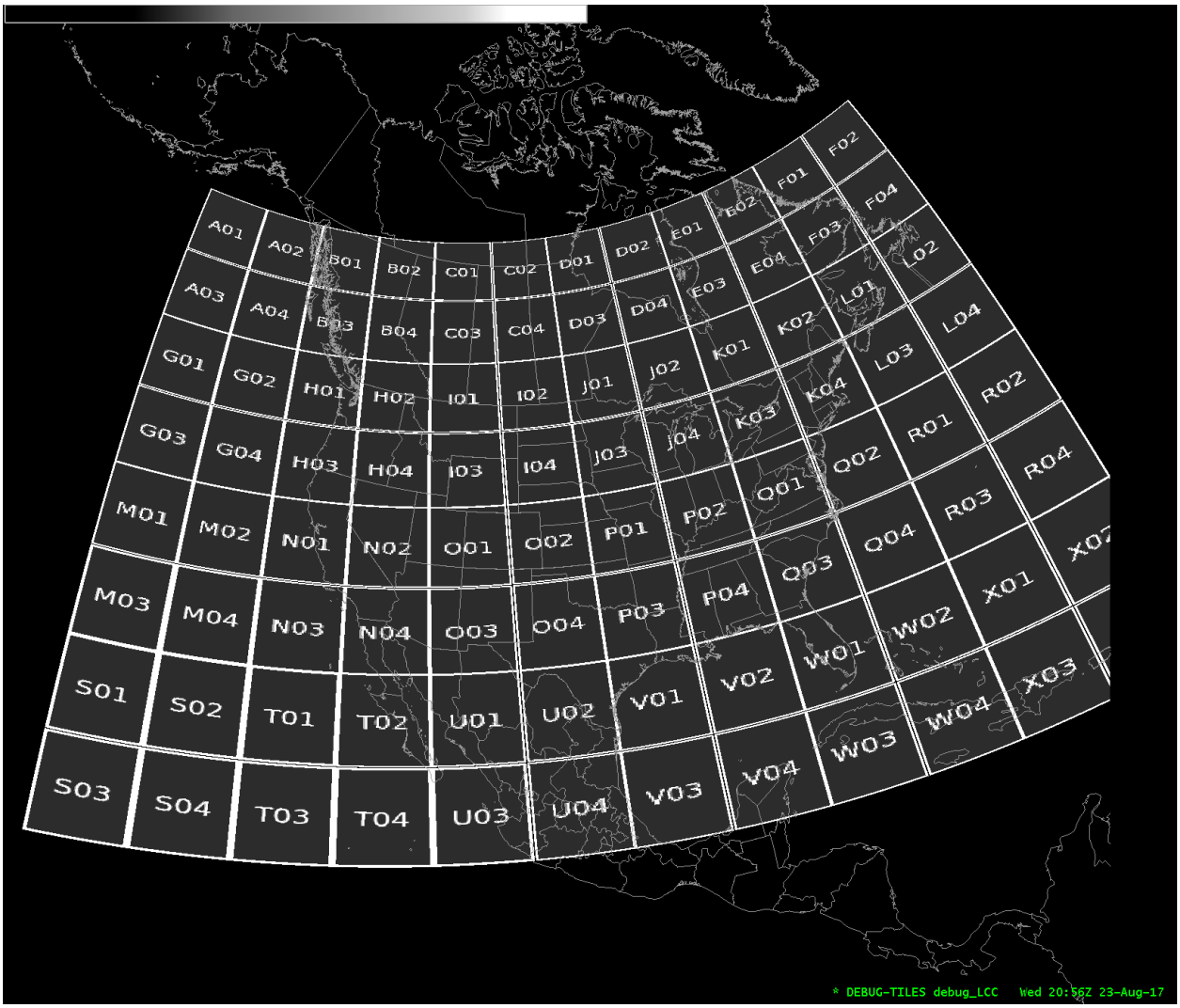


Fig. 6.1: "LCC" Sector Lettered Grid

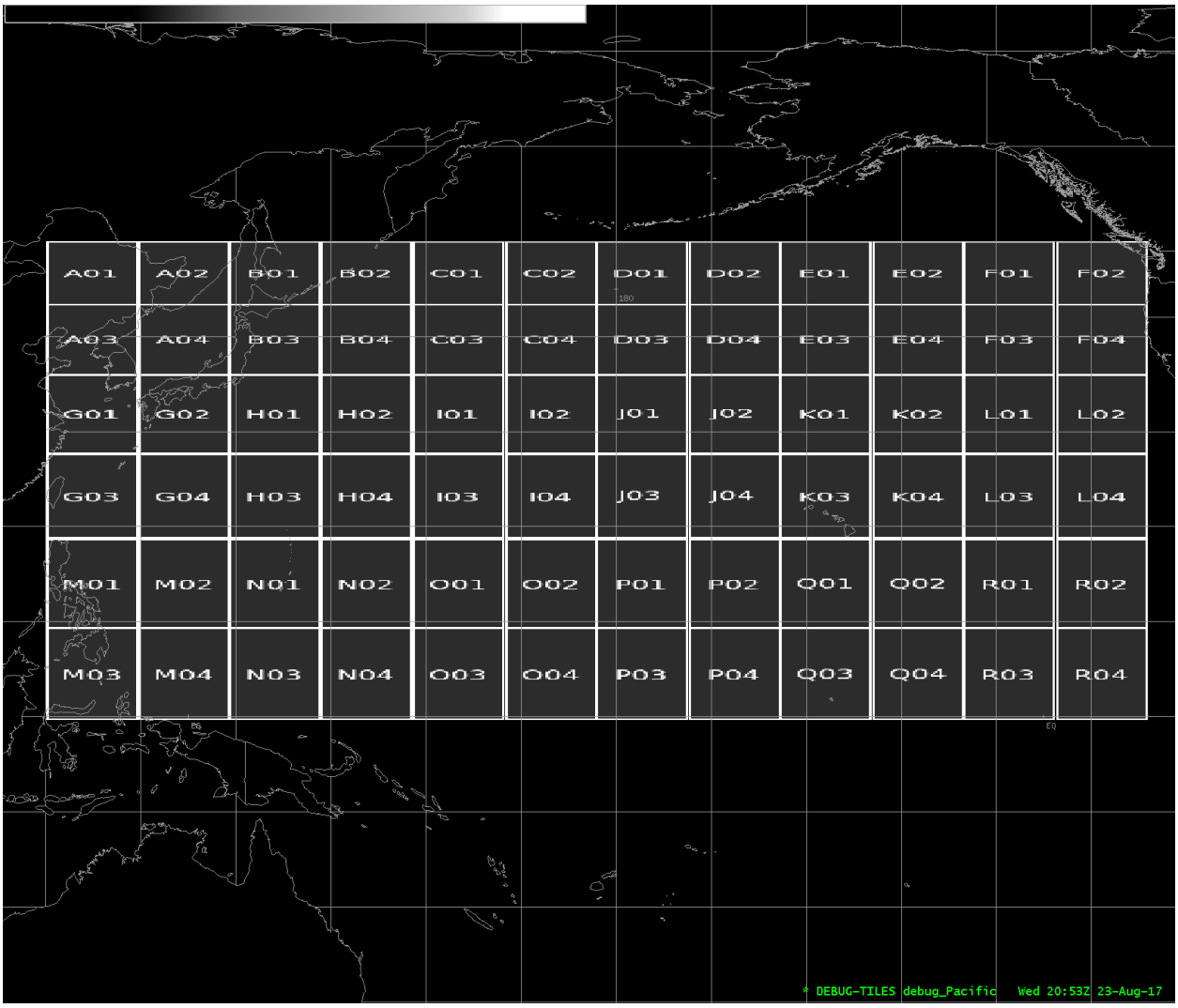


Fig. 6.2: "Pacific" Sector Lettered Grid

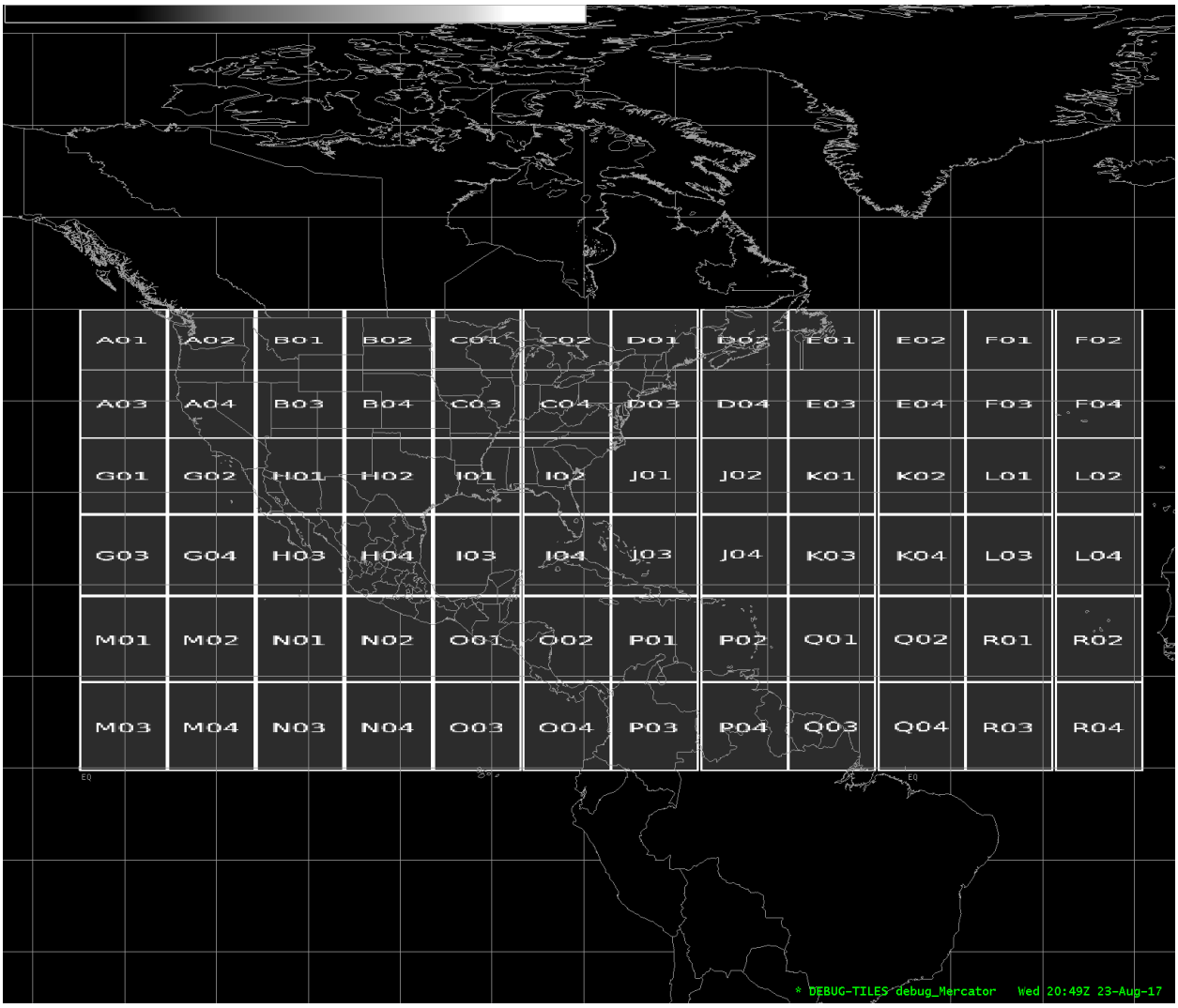


Fig. 6.3: "Mercator" Sector Lettered Grid

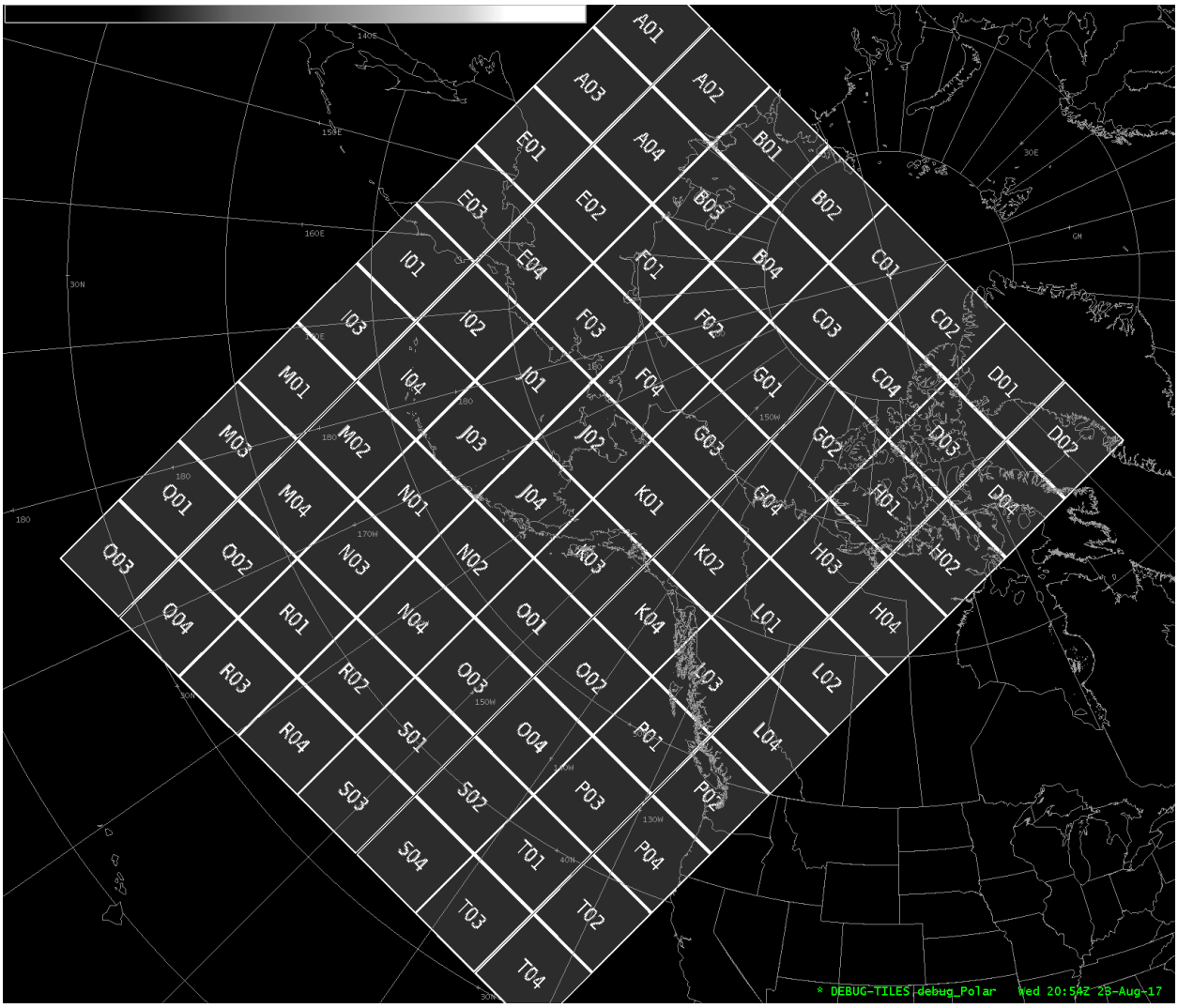


Fig. 6.4: "Polar" Sector Lettered Grid

Examples:

```
polar2grid.sh -r viirs_sdr -w awips_tiled --awips-true-color --awips-false-color --
↪num-workers 8 -g lcc_conus_300 --sector-id LCC --letters --compress -f viirs/*.h5

polar2grid.sh -r modis_l1b -w awips_tiled -p vis01 bt31 -g lcc_conus_1km --sector-id␣
↪LCC --letters --compress -f a1.22261.1857.250m.hdf a1.22261.1857.1000m.hdf a1.22261.
↪1857.geo.hdf

polar2grid.sh -r amsr2_l1b -w awips_tiled --num-workers 4 -grid-coverage 0.002 -g␣
↪polar_alaska_1km --sector-id Polar --letters --compress -f $data_dir/GW1AM2_
↪202209102335_181A_L1DLBTBR_1110110.h5

polar2grid.sh -r amsr2_l1b -w awips_tiled --grid-coverage 0 -g merc_pacific_1km --
↪sector-id Pacific --letters --compress -f GW1AM2_202209120018_188A_L1DLBTBR_1110110.
↪h5

 polar2grid.sh -r clavrx -w awips_tiled -p cld_height_acha cloud_type cld_temp_acha -
↪g lcc_conus_300 --sector-id LCC --letters --compress --grid-coverage 0.002 -f␣
↪clavrx_a1.22245.0759.1000m.level2.hdf

polar2grid.sh -r acspo -w awips_tiled --num-workers 8 --grid-coverage 0 -g lcc_conus_
↪750 --sector-id LCC --letters --compress --method ewa --weight-delta-max 40.0 --
↪weight-distance-max 1.0 -f $data_dir/202*VIIRS_NPP-ACSPO_V2.80*.nc
```

6.2 Binary Writer

The Binary writer writes band data to a flat binary file.

By default it enhances the data based on the enhancement configuration file and then saves the data to a flat binary file. The output data type will match the input data for integer types (ex. uint8 -> uint8), but floating point types will always be forced to 32-bit floats for consistency between readers and changes in the low-level Satpy library. A different output type can be specified using the `--dtype` flag. To turn off scaling of the data (a.k.a. enhancements) the `--no-enhance` command line flag can be specified to write the "raw" band data.

6.2.1 Command Line Arguments

```
usage: polar2grid.sh -r <reader> -w binary [-h] [--output-filename FILENAME]
                                           [--dtype {uint8,uint16,uint32,uint64,int8,
↪int16,int32,int64,float32,float64}]
                                           [--no-enhance]
                                           [--fill-value FILL_VALUE]
```

Binary Writer

--output-filename	Custom file pattern to save dataset to
--dtype	Possible choices: uint8, uint16, uint32, uint64, int8, int16, int32, int64, float32, float64 Data type of the output file (8-bit unsigned integer by default - uint8)
--no-enhance	Don't enhance the data before saving it Default: True
--fill-value	Replace invalid values with the specified value. Floating-point products typically use NaN while integer fields will use 0 or the max value for that data type.

6.3 GeoTIFF Writer

The GeoTIFF writer puts gridded image data into a standard GeoTIFF file.

It uses the GDAL python API and rasterio python package to create the GeoTIFF files. It can handle any grid that can be described by PROJ.4 and understood by the GeoTIFF format.

By default the 'geotiff" writer will add an "Alpha" band to the file to mark any invalid or missing data pixels. This results in invalid pixels showing up as transparent in most image viewers.

6.3.1 Command Line Arguments

```
usage: |script| -r <reader> -w geotiff [-h] [--output-filename FILENAME]
                                       [--dtype {uint8,uint16,uint32,uint64,int8,
↪int16,int32,int64,float32,float64}]
                                       [--no-enhance]
                                       [--fill-value FILL_VALUE]
                                       [--compress COMPRESS] [--keep-palette]
                                       [--tiled] [--blockxsize BLOCKXSIZE]
                                       [--blockysize BLOCKYSIZE]
                                       [--overviews OVERVIEWS]
                                       [--overviews-resampling {nearest,average,
↪bilinear,cubic,cubicspline,lanczos}]
                                       [--gdal-driver DRIVER]
```

Geotiff Writer

--output-filename	Custom file pattern to save dataset to
--dtype	Possible choices: uint8, uint16, uint32, uint64, int8, int16, int32, int64, float32, float64 Data type of the output file (8-bit unsigned integer by default - uint8)
--no-enhance	Don't try to enhance the data before saving it
--fill-value	Instead of an alpha channel fill invalid values with this value. Turns LA or RGBA images in to L or RGB images respectively.

--compress File compression algorithm (DEFLATE, LZW, NONE, etc)

Default: "LZW"

--keep-palette When saving 'palettized' enhanced images, save the colormap as a geotiff color table instead of converting the image to RGB/A

--tiled, --no-tiled Tile geotiffs internally (default: True) (default: True)

Default: True

--blockxsize Set tile block X size

--blockysize Set tile block Y size

--overviews Build lower resolution versions of your image for better performance in some clients. Specified as a space separate list of numbers, typically as powers of 2. Example: '2 4 8 16'

--overviews-resampling Possible choices: nearest, average, bilinear, cubic, cubicspline, lanczos

Specify resampling used when generating overviews

Default: "nearest"

--gdal-driver Name of the GDAL driver to use when writing the geotiff. By default the 'geotiff' driver is used. If '–driver COG' is used then the GDAL 'COG' driver will be used and will create a tiled COG-compatible geotiff.

6.4 HDF5 Writer

The HDF5 writer creates HDF5 files with groups for each gridded area.

All selected products are in one file. Products are subgrouped together under a parent HDF5 data group based on the data product projection/remapping (parent projection group). Each parent projection group contains attributes describing the projection. Product subgroups contain attributes of the data including timestamps, sensor and platform information. See the command line arguments for HDF5 compression options, the flag to include longitude and latitude data in the file, instructions for output-filename patterns, and product selection.

6.4.1 Command Line Arguments

```
usage: polar2grid.sh -r <reader> -w hdf5 [-h] [--output-filename FILENAME]
                                         [--dtype {uint8,uint16,uint32,uint64,int8,
↪int16,int32,int64,float32,float64}]
                                         [--compress {none,gzip,lzf}]
                                         [--add-geolocation] [--no-append]
```

HDF5 Writer

--output-filename	Custom file pattern to save dataset to
--dtype	Possible choices: uint8, uint16, uint32, uint64, int8, int16, int32, int64, float32, float64
	Data type of the output file (8-bit unsigned integer by default - uint8)
--compress	Possible choices: none, gzip, lzf
	Dataset compression algorithm. Defaults to no compression.
	Default: "none"
--add-geolocation	Add 'longitude' and 'latitude' datasets for each grid
	Default: False
--no-append	Don't append to the HDF5 file if it already exists (otherwise may overwrite data)
	Default: True

CHAPTER

SEVEN

UTILITY SCRIPTS

The following are scripts that can be used to aid in the creation of customized Polar2Grid products. All utility scripts are stored in the bin directory:

```
$POLAR2GRID_HOME/bin/<script>.sh ...
```

For simplicity, the sections below will specify the script directly, but note the scripts exist in the bin directory above.

7.1 Defining Your Own Grids (Grid Configuration Helper)

This script is meant to help those unfamiliar with PROJ.4 and projections in general. By providing a few grid parameters this script will provide a grid configuration line that can be added to a user's custom grid configuration. Based on a center longitude and latitude, the script will choose an appropriate projection.

```
usage: p2g_grid_helper.sh [-h] [-p PROJ_STR] [--legacy-format]
                          grid_name center_longitude center_latitude
                          pixel_size_x pixel_size_y grid_width grid_height
```

7.1.1 Positional Arguments

grid_name	Unique grid name
center_longitude	Decimal longitude value for center of grid (-180 to 180)
center_latitude	Decimal latitude value for center of grid (-90 to 90)
pixel_size_x	Size of each pixel in the X direction in grid units, meters for default projections.
pixel_size_y	Size of each pixel in the Y direction in grid units, meters for default projections.
grid_width	Grid width in number of pixels
grid_height	Grid height in number of pixels

7.1.2 Named Arguments

-p	PROJ.4 projection string to override the default
--legacy-format	Produce a legacy '.conf' format grid definition.

Example:

```
p2g_grid_helper.sh my_grid_name -150.1 56.3 250 -250 1000 1000
```

Will result in:

```
my_grid_name:
  projection:
    proj: lcc
    lat_1: 56.3
    lat_0: 56.3
    lon_0: -150.1
    datum: WGS84
    units: m
    no_defs: null
    type: crs
  shape:
    height: 1000
    width: 1000
  center:
    x: -150.1
    y: 56.3
    units: degrees
  resolution:
    dx: 250.0
    dy: 250.0
```

The above example creates a YAML formatted block of text for the grid named 'my_grid_name'. It is defined to have a pixel resolution of 250m, have 1000 rows and 1000 columns, and be centered at -150.1 degrees longitude and 56.3 degrees latitude. The projection is a lambert conic conformal projection which was chosen based on the center longitude and latitude.

Once this text has been output, it can be added to a text file ending in `.yaml` and referenced in the `polar2grid.sh` command line. For instance, if I save the output text to a file named `/home/user/my_grids.yaml`, I can create a GeoTIFF from satellite data by executing a command like this:

```
polar2grid.sh -r viirs_sdr -w geotiff --grid-configs /home/p2g/my_grids.yaml -g my_
↪grid_name -f <path_to_files>
```

7.2 Add Overlays (Borders, Coastlines, Grids Lines, Rivers)

Add overlays to a GeoTIFF file and save as a PNG file.

```
usage: add_coastlines.sh [-h] [--add-coastlines]
                         [--coastlines-resolution {c,l,i,h,f}]
                         [--coastlines-level {1,2,3,4,5,6}]
                         [--coastlines-outline [COASTLINES_OUTLINE [COASTLINES_
↪OUTLINE ...]]]
```

(continues on next page)

(continued from previous page)

```
                        [--coastlines-fill [COASTLINES_FILL [COASTLINES_FILL ...]]]
                        [--coastlines-width COASTLINES_WIDTH] [--add-rivers]
                        [--rivers-resolution {c,l,i,h,f}]
                        [--rivers-level {0,1,2,3,4,5,6,7,8,9,10}]
                        [--rivers-outline [RIVERS_OUTLINE [RIVERS_OUTLINE ...]]]
                        [--rivers-width RIVERS_WIDTH] [--add-grid]
                        [--grid-no-text] [--grid-text-size GRID_TEXT_SIZE]
                        [--grid-font GRID_FONT]
                        [--grid-fill [GRID_FILL [GRID_FILL ...]]]
                        [--grid-outline [GRID_OUTLINE [GRID_OUTLINE ...]]]
                        [--grid-minor-outline [GRID_MINOR_OUTLINE [GRID_MINOR_
↪OUTLINE ...]]]
                        [--grid-D GRID_D GRID_D] [--grid-d GRID_D GRID_D]
                        [--grid-lon-placement {tl,lr,lc,cc}]
                        [--grid-lat-placement {tl,lr,lc,cc}]
                        [--grid-width GRID_WIDTH] [--add-borders]
                        [--borders-resolution {c,l,i,h,f}]
                        [--borders-level {1,2,3}]
                        [--borders-outline [BORDERS_OUTLINE [BORDERS_OUTLINE ...]]]
                        [--borders-width BORDERS_WIDTH] [--add-colorbar]
                        [--colorbar-width COLORBAR_WIDTH]
                        [--colorbar-height COLORBAR_HEIGHT]
                        [--colorbar-extend]
                        [--colorbar-tick-marks COLORBAR_TICK_MARKS]
                        [--colorbar-minor-tick-marks COLORBAR_MINOR_TICK_MARKS]
                        [--colorbar-text-size COLORBAR_TEXT_SIZE]
                        [--colorbar-text-color [COLORBAR_TEXT_COLOR [COLORBAR_TEXT_
↪COLOR ...]]]
                        [--colorbar-font COLORBAR_FONT]
                        [--colorbar-align {left,top,right,bottom}]
                        [--colorbar-vertical] [--colorbar-min COLORBAR_MIN]
                        [--colorbar-max COLORBAR_MAX]
                        [--colorbar-units COLORBAR_UNITS]
                        [--colorbar-title COLORBAR_TITLE]
                        [--shapes-dir SHAPES_DIR] [--cache-dir CACHE_DIR]
                        [--cache-regenerate]
                        [-o OUTPUT_FILENAME [OUTPUT_FILENAME ...]] [-v]
                        input_tiff [input_tiff ...]
```

7.2.1 Positional Arguments

input_tiff Input geotiff(s) to process

7.2.2 Named Arguments

--shapes-dir Specify alternative directory for coastline shape files (default: GSHSS_DATA_ROOT)

--cache-dir Specify directory where cached coastline output can be stored and accessed in later executions. The cache will never be cleared by this script. Caching depends on the grid of the image and the decorations added to the image.

--cache-regenerate Force regeneration of any cached overlays. Requires '–cache-dir'.

-o, --output Specify the output filename (default replace '.tif' with '.png')

-v, --verbose each occurrence increases verbosity 1 level through ERROR-WARNING-INFO-DEBUG (default INFO)

Default: 0

7.2.3 coastlines

--add-coastlines Add coastlines

--coastlines-resolution Possible choices: c, l, i, h, f

Resolution of coastlines to add (crude, low, intermediate, high, full)

Default: "i"

--coastlines-level Possible choices: 1, 2, 3, 4, 5, 6

Level of detail from the selected resolution dataset

Default: 4

--coastlines-outline Color of coastline lines (color name or 3 RGB integers)

Default: ['yellow']

--coastlines-fill Color of land

--coastlines-width Width of coastline lines

Default: 1.0

7.2.4 rivers

--add-rivers Add rivers grid

--rivers-resolution Possible choices: c, l, i, h, f

Resolution of rivers to add (crude, low, intermediate, high, full)

Default: "c"

--rivers-level Possible choices: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Level of detail for river lines

Default: 5

--rivers-outline Color of river lines (color name or 3 RGB integers)

Default: ['blue']

--rivers-width Width of rivers lines

Default: 1.0

7.2.5 grid

--add-grid Add lat/lon grid

--grid-no-text Add labels to lat/lon grid

--grid-text-size Lat/lon grid text font size

Default: 32

--grid-font Path to TTF font (package provided or custom path)

Default: "Vera.ttf"

--grid-fill Color of grid text (color name or 3 RGB integers)

Default: ['cyan']

--grid-outline Color of grid lines (color name or 3 RGB integers)

Default: ['cyan']

--grid-minor-outline Color of tick lines (color name or 3 RGB integers)

Default: ['cyan']

--grid-D Degrees between grid lines (lon, lat)

Default: (10.0, 10.0)

--grid-d Degrees between tick lines (lon, lat)

Default: (2.0, 2.0)

--grid-lon-placement Possible choices: tl, lr, lc, cc

Longitude label placement

Default: "tb"

--grid-lat-placement Possible choices: tl, lr, lc, cc

Latitude label placement

Default: "lr"

--grid-width Width of grid lines

Default: 1.0

7.2.6 borders

--add-borders Add country and/or region borders

--borders-resolution Possible choices: c, l, i, h, f

Resolution of borders to add (crude, low, intermediate, high, full)

Default: "i"

--borders-level Possible choices: 1, 2, 3

Level of detail for border lines

Default: 2

--borders-outline Color of border lines (color name or 3 RGB integers)

Default: ['white']

--borders-width Width of border lines

Default: 1.0

7.2.7 colorbar

--add-colorbar Add colorbar on top of image

--colorbar-width Number of pixels wide

--colorbar-height Number of pixels high

--colorbar-extend Extend colorbar to full width/height of the image

--colorbar-tick-marks Major tick and tick label interval in data units

Default: 5.0

--colorbar-minor-tick-marks Minor tick interval in data units

Default: 1.0

--colorbar-text-size Tick label font size

Default: 32

--colorbar-text-color Color of tick text (color name or 3 RGB integers)

Default: ['black']

--colorbar-font Path to TTF font (package provided or custom path)

Default: "Vera.ttf"

--colorbar-align Possible choices: left, top, right, bottom

Which side of the image to place the colorbar

Default: "bottom"

--colorbar-vertical DEPRECATED

--colorbar-min Minimum data value of the colorbar. Defaults to 'min_in' of input metadata or minimum value of the data otherwise.

--colorbar-max Maximum data value of the colorbar. Defaults to 'max_in' of input metadata or maximum value of the data otherwise.

--colorbar-units Units marker to include in the colorbar text

--colorbar-title Title shown with the colorbar

Examples:

```
add_coastlines.sh noaa20_viirs_true_color_20221011_174112_wgs84_fit.tif --add-
↪coastlines --coastlines-outline yellow --coastlines-level 1 --coastlines-
↪resolution=i --add-borders --borders-level 2 --borders-outline gray --add-grid --
↪grid-text-size 16 --grid-fill white --grid-D 5 5 --grid-d 5 5 --grid-outline white
```

7.3 Add Colormap

Add a GeoTIFF colortable to an existing single-band GeoTIFF.

```
usage: add_colormap.sh [-h] ct_file geotiffs [geotiffs ...]
```

7.3.1 Positional Arguments

ct_file Color table file to apply (CSV of (int, R, G, B, A)

geotiffs Geotiff files to apply the color table to

Colormap files are comma-separated 'integer,R,G,B,A' text files.

A basic greyscale example for an 8-bit GeoTIFF would be:

```
0,0,0,0,255
1,1,1,1,255
...
254,254,254,254,255
255,255,255,255,255
```

Where the … represents the lines in between, meaning every input GeoTIFF value has a corresponding RGBA value specified. The first value is the input GeoTIFF value, followed by R (red), G (green), B (blue), and A (alpha).

This script will also linearly interpolate between two values. So the above colormap file could also be written in just two lines:

```
0,0,0,0,255
255,255,255,255,255
```

Often times you may want to have the 0 value as a transparent 'fill' value and continue the colormap after that. This can be done by doing the following:

```
# 0 is a fill value
0,0,0,0,0
# 1 starts at bright red
1,255,0,0,255
# and we end with black at the end
255,0,0,0,255
```

Note: Not all image viewers will obey the transparent (alpha) settings

Blank lines are allowed as well as spaces between line elements.

Note this script is no longer needed in modern versions of Polar2Grid if the original geotiff (no color) is not needed. The colormap can be specified directly in the enhancement YAML file for a product. For example, for the AMSR-2 L1B product "btemp_36.5h" we could add the following to a `etc/enhancements/amsr2.yaml` (or `generic.yaml`):

```
yaml

amsr2_btemp_365h:
  name: btemp_36.5h
```

(continues on next page)

(continued from previous page)

```
  sensor: amsr2
  operations:
    - name: add_colormap
      method: !!python/name:polar2grid.enhancements.palettize
      kwargs:
        palettes:
          - filename: $POLAR2GRID_HOME/colormaps/amsr2_36h.cmap
            min_value: 180
            max_value: 280
```

When saved using the 'geotiff' writer this will be converted to an RGB/RGBA image. Optionally you can provide the `--keep-palette` flag to your `polar2grid.sh` call which will add the colormap as a geotiff color table.

7.4 GeoTIFF to KMZ Conversion

The `gtiff2kmz.sh` script converts a single GeoTIFF file into a Google Earth compatible Keyhole Markup language Zipped (KMZ) file. It is a wrapper around the GDAL tool `gdal2tiles.py`. The script can be executed with:

```
gtiff2kmz.sh input.tif [output.kmz]
```

Where `output.kmz` is an optional parameter specifying the name of the output KMZ file. If it isn't specified it defaults to the input filename with the extension changed to `.kmz`.

Example:

```
gtiff2kmz.sh noaa20_viirs_false_color_20221011_174112_wgs84_fit.tif
```

7.5 Overlay GeoTIFF Images

The `overlay.sh` script can be used to overlay one GeoTIFF image (ex. VIIRS EDR Active Fires) on top of another image (ex. VIIRS Adaptive DNB or True Color). This script uses GDAL's `gdal_merge.py` utility underneath, but converts everything to RGBA format first for better consistency in output images.

```
usage: overlay.sh background.tif foreground.tif out.tif
```

Example: The following example shows how you would overlay the VIIRS Active Fire AFMOD resolution Fire Confidence Percentage GeoTIFF image on top of a VIIRS Day/Night Band GeoTIFF image.

```
overlay.sh noaa20_viirs_dynamic_dnb_20191120_151043_wgs84_fit.tif noaa20_viirs_
↪confidence_pct_20191120_151043_wgs84_fit.tif afmod_overlay_confidence_cat.tif
```

7.6 Convert GeoTIFFs to MP4 Video

The `gtiff2mp4.sh` script converts a series of GeoTIFF files in to a single MP4 video file. This script uses default video creation settings to support most video players. If an image is too large for the video creation they will be automatically scaled to a smaller size.

```
gtiff2mp4.sh out.mp4 in1.tif in2.tif ...
```

This will create a MP4 video file called `out.mp4` with 24 images (frames) per second.

Example:

```
gtiff2mp4.sh my_natural_color_animation.mp4  *natural_color*.tif
```

7.7 Remap GOES GeoTIFFs

The projection of the GOES-East and GOES-West satellites uses special parameters that are not always supported by older visualization tools. While new versions of GDAL and PROJ.4 libraries can often fix these issues, this is not always an option. Polar2Grid provides the `reproject_goes.sh` script to remap GOES GeoTIFFs to a nearly identical projection that is more compatible with older visualization tools. The script can be called by executing:

```
reproject_goes.sh in1.tif in2.tif in3.tif
```

The script will take the original name and add a `-y` to the end. So in the above example the results would be `in1-y.tif`, `in2-y.tif`, and `in3-y.tif`. The `y` refers to the sweep angle axis projection parameter that differs between the input geotiff (`x`) and the output geotiff (`y`).

7.8 Convert legacy grids.conf to grids.yaml format

Convert legacy grids.conf format to Pyresample YAML format.

```
usage:
To write to a file:
    convert_grids_conf_to_yaml.sh input_file.conf > output_file.yaml
```

7.8.1 Positional Arguments

grids_filename Input grids.conf-style file to convert to YAML.

Example:

```
convert_grids_conf_to_yaml.sh old_file.conf > new_file.yaml
```

CHAPTER

EIGHT

VERIFYING YOUR POLAR2GRID INSTALLATION

8.1 Executing the VIIRS Polar2Grid Test Case

To run the VIIRS GeoTIFF test case, unpack the test data as shown in Section 2.2 and execute the following commands:

```
cd polar2grid_test/viirs
mkdir work
cd work
polar2grid.sh -r viirs_sdr -w geotiff -p true_color false_color --grid-configs \
  ${POLAR2GRID_HOME}/grid_configs/grid_example.yaml -g miami --weight-distance-max 1 -
↪f ../input
```

The test case consists of 6 input direct broadcast HDF 5 SDR granules for a selection of VIIRS bands from a pass acquired on 19 March 2017 at 18:32 UTC. In this test, the Polar2Grid software is using the example configuration file (${POLAR2GRID_HOME}/grid_configs/grid_example.yaml) and the lambert conformal conic (lcc) miami grid definition entry located within it. We will create one true and one false color image at 300 m resolution, 750 lines x 1000 elements centered on the US city of Miami in the state of Florida.

The creation of the true and false color images includes the Atmospheric Rayleigh Scattering Correction, and sharpening of the image to the spatial resolution of the VIIRS I-Bands. We are using a `--weight-distance-max` option of `1` to inform the elliptical weight averaging (EWA) technique how to weight the effect of the input pixel to an output pixel based upon its location in the scan line and other calculated coefficients. Although this may result in the "sharpest" output resolution image, the user should be aware that with reprojecting VIIRS terrain corrected imagery this may lead to black missing data sections in regions of varying terrains, especially at higher view angles. That is why the default `--weight-distance-max` value is `2`.

The processing should run in less than 2 minutes and create 2 atmospherically corrected and sharpened output VIIRS GeoTIFF true and false color images.

If the VIIRS Polar2Grid processing script runs normally, it will return a status code equal to zero. If the VIIRS Polar2Grid processing script encounters a fatal error, it will return a non-zero status code.

To verify your output files against the output files created at UW/SSEC, execute the following commands:

```
cd ..
p2g_compare.sh output work
```

This script compares the values of all bands in the GeoTIFF file for the true and false color high resolution images. The verification text string from our test system is shown below.

```
p2g_compare.sh output work

Comparing work/npp_viirs_false_color_20170319_183246_miami.tif to known valid file
INFO:__main__:Comparing 'work/npp_viirs_false_color_20170319_183246_miami.tif' to␣
↪known valid file 'output/npp_viirs_false_color_20170319_183246_miami.tif'...
```

(continues on next page)

(continued from previous page)

```
INFO:__main__:0 pixels out of 3000000 pixels are different
INFO:__main__:Comparing 'work/npp_viirs_true_color_20170319_183246_miami.tif' to␣
↪known valid file 'output/npp_viirs_true_color_20170319_183246_miami.tif'.
INFO:__main__:0 pixels out of 3000000 pixels are different
All files passed
SUCCESS
```

The VIIRS true color GeoTIFF image created from the test data is displayed below:

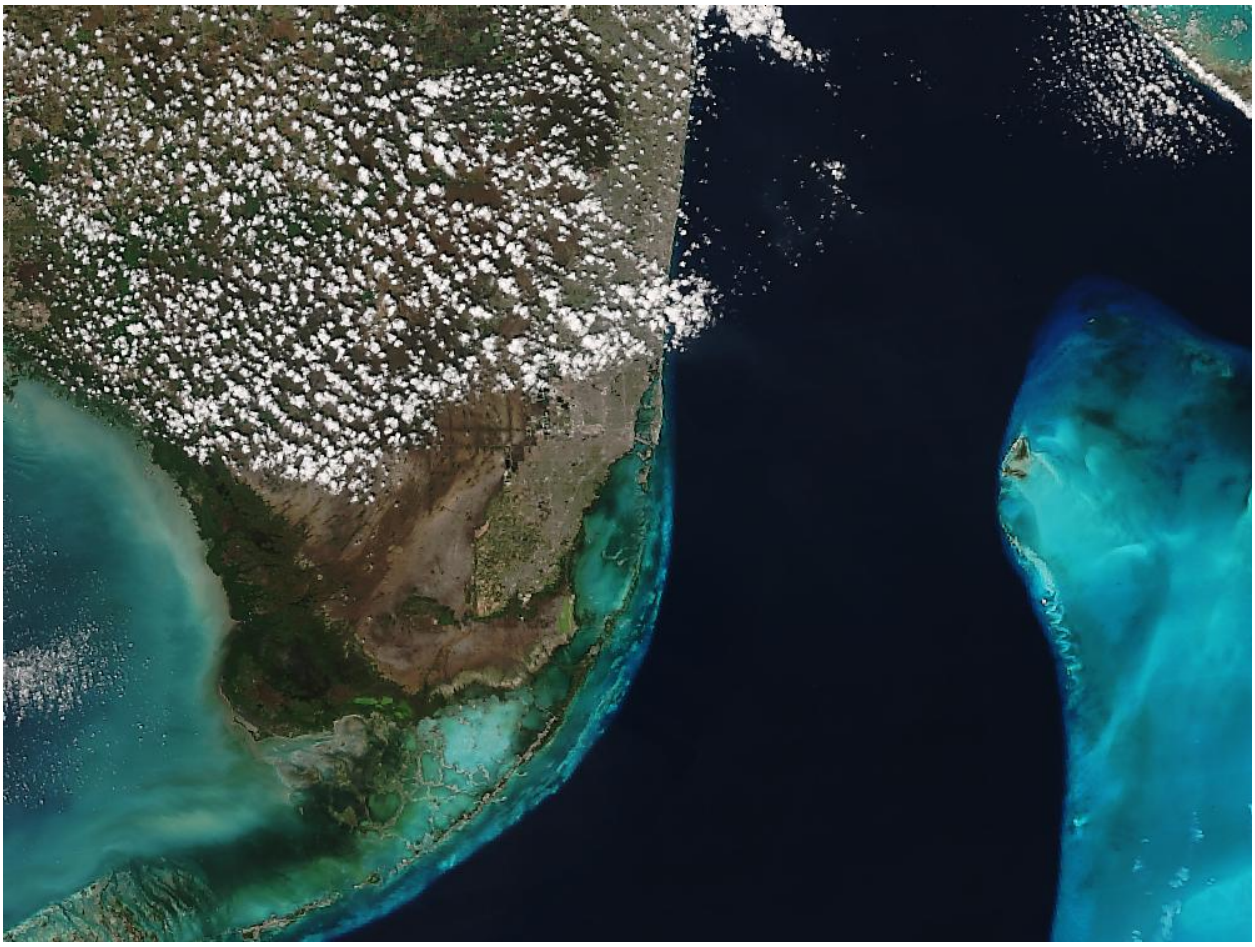


Fig. 8.1: GeoTIFF true color image created from the 19 March 2017 VIIRS test data centered on Miami, Florida.

8.2 Executing the MODIS Polar2Grid Test Case

To run the MODIS GeoTIFF test case, unpack the test data as shown in Section 2.2 and execute the following commands:

```
cd polar2grid_test/modis
mkdir work
cd work
polar2grid.sh -r modis -w geotiff -p true_color false_color --fill-value 0 \
  --grid-configs ${POLAR2GRID_HOME}/grid_configs/grid_example.yaml -g miami -f ../
↪input/
```

The test case consists of a set of MODIS archived 5 minute HDF 4 Level 1B granule files (1KM, HKM, QKM and Geolocation) for a Terra MODIS pass observed on 19 March 2017 at 16:30 UTC. In this test, the Polar2Grid software is using the example configuration file (${POLAR2GRID_HOME}/grid_configs/grid_example.yaml) and the lambert conformal conic (lcc) "miami" grid definition entry located within it. The software goes through a number of steps to produce the true and false color imagery, including the removal of the atmospheric Rayleigh Scattering, creation of reflectances from the normalized radiances, sharpening the image to full resolution and combining the 3 bands into 24-bit output GeoTIFF files. The end result is one true and one false color image at 300 m resolution, 750 lines x 1000 elements centered on the US city of Miami in the state of Florida. The processing should run in less than 2 minutes.

If the MODIS Polar2Grid processing script runs normally, it will return a status code equal to zero. If the MODIS Polar2Grid processing script encounters a fatal error, it will return a non-zero status code.

To verify your output files against the output files created at UW/SSEC, execute the following commands:

```
cd ..
p2g_compare.sh output work
```

This script compares the values of all bands in the GeoTIFF file for the true and false color high resolution images. The verification text string from our test system is shown below.

```
p2g_compare.sh output work

INFO:__main__:Comparing 'work/terra_modis_false_color_20170319_163000_miami.tif' to␣
↪known valid file 'output/terra_modis_false_color_20170319_163000_miami.tif'.
INFO:__main__:0 pixels out of 2250000 pixels are different
INFO:__main__:Comparing 'work/terra_modis_true_color_20170319_163000_miami.tif' to␣
↪known valid file 'output/terra_modis_true_color_20170319_163000_miami.tif'.
INFO:__main__:0 pixels out of 2250000 pixels are different
All files passed
SUCCESS
```

The Terra MODIS false color GeoTIFF image created from the test data is displayed below:

Fig. 8.2: GeoTIFF false color image created from the 19 March 2017 Terra MODIS test data centered on Miami, Florida.

CHAPTER

NINE

EXAMPLES

9.1 Creating VIIRS SDR GeoTIFF Files

This example walks through the creation of VIIRS GeoTIFF output files and adding overlays.

9.1.1 Basic VIIRS SDR GeoTIFF file creation

Find the options available when creating VIIRS SDR GeoTIFFs:

    polar2grid.sh -r viirs_sdr -w geotiff -h

List the supported products that can be created from your VIIRS SDR dataset:

```
polar2grid.sh -r viirs_sdr -w geotiff --list-products -f <path_to_sdr_files>
```

This will provide a list of standard products that can be created from the files that are provided to polar2grid.sh.

To create VIIRS GeoTIFF files of all default products (including true and false color) found in your data set and reprojected in default Platte Carrée projection using the default 4 workers, execute the following command:

    polar2grid.sh -r viirs_sdr -w geotiff -f <path_to_sdr_files>

Create a subset of VIIRS I- and M-Band reprojected GeoTIFFs using 8 workers:

```
polar2grid.sh -r viirs_sdr -w geotiff -p i01 i05 m09 m14 --num-workers 8 -f <path_to_
↪sdr_files>
```

Create only true color and false color GeoTIFFs with a black background (no alpha channel):

```
polar2grid.sh -r viirs_sdr -w geotiff -p true_color false_color --fill-value 0 -f
↪<path_to_sdr_files>
```

Create a true color image from a S-NPP VIIRS pass acquired on 19 September 2022, 17:53 UTC, in a US Centric Lambert Conformal Conic (LCC) projection:

```
polar2grid.sh -r viirs_sdr -w geotiff -p true_color -g lcc_fit -f /data/viirs_sdr
```

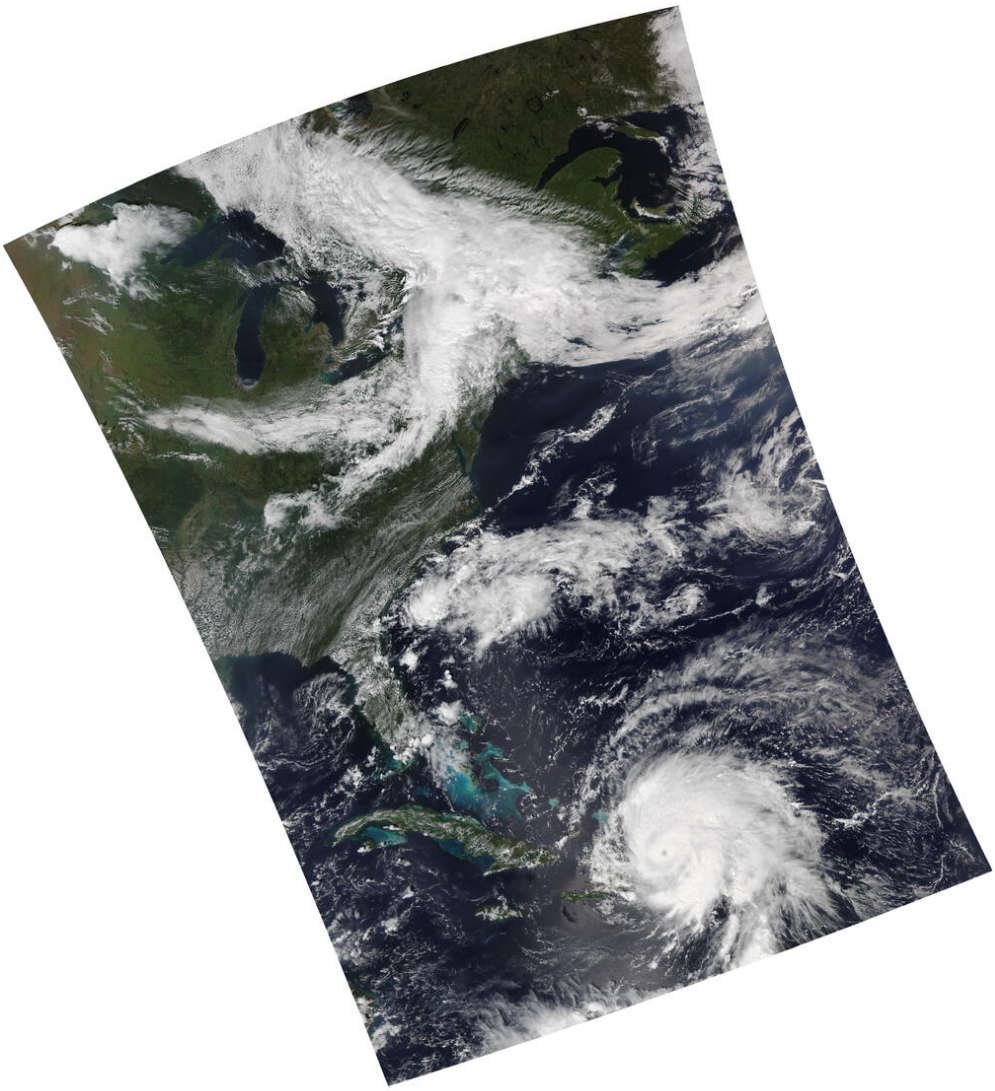


Fig. 9.1: VIIRS True color image in Lambert Conformal Conic (LCC) projection (noaa20_viirs_true_color_20220919_175331_lcc_fit.tif).

Add coastlines, borders and latitude/longitude grid lines to the true color image, and write the output to the file "myfile.png".

```
add_coastlines.sh --add-coastlines --add-borders --borders-resolution=h --borders-
↪outline='red' --add-grid noaa20_viirs_true_color_20220919_175331_lcc_fit.tif -o␣
↪myfile.png
```

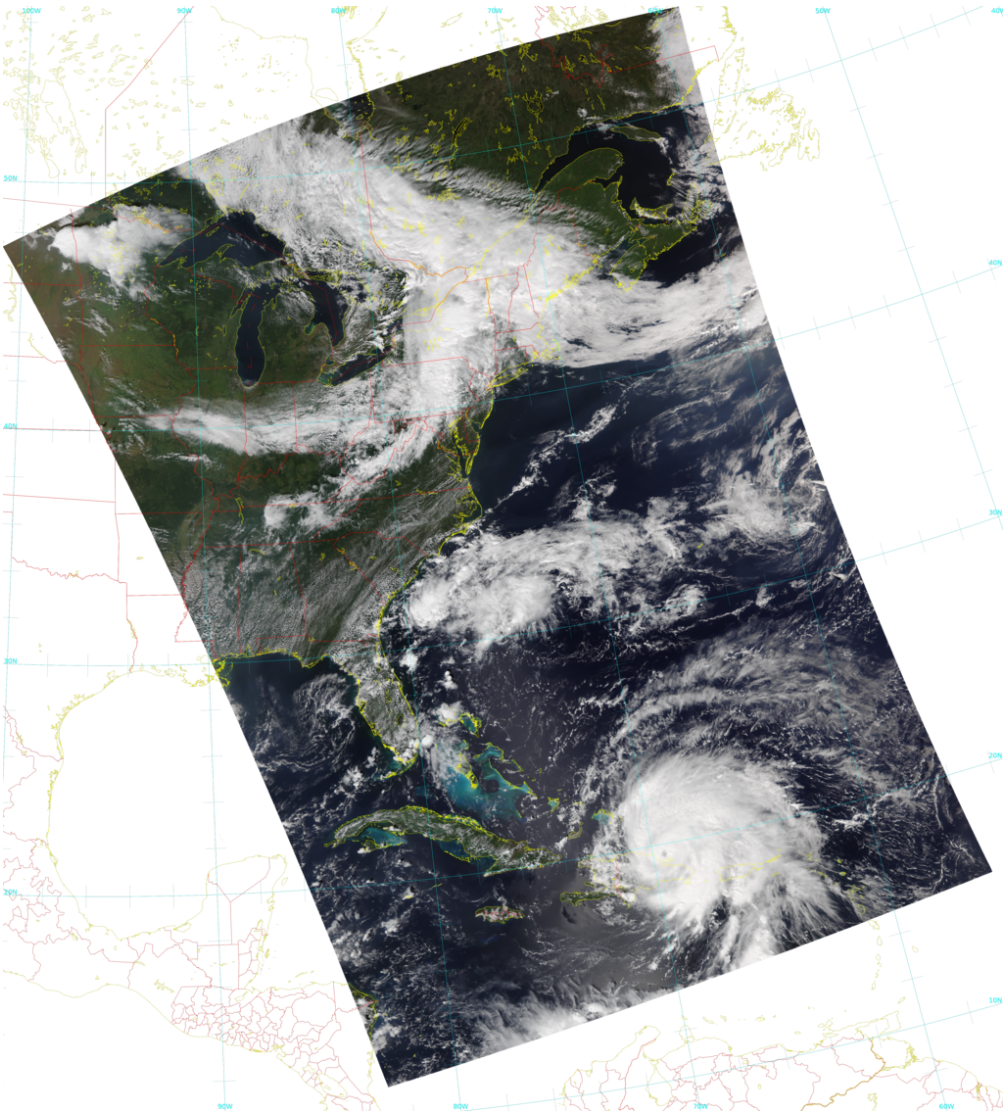


Fig. 9.2: VIIRS True color image with overlays (myfile.png).

Convert the true color GeoTIFF file into a Google Earth compatible Keyhole Markup language Zipped (KMZ) file.

```
gtiff2kmz.sh noaa20_viirs_true_color_20220919_175331_lcc_fit.tif
```

which creates the *noaa20_viirs_true_color_20220919_175331_lcc_fit.kmz* file. When displayed in Google Earth this image appears as:

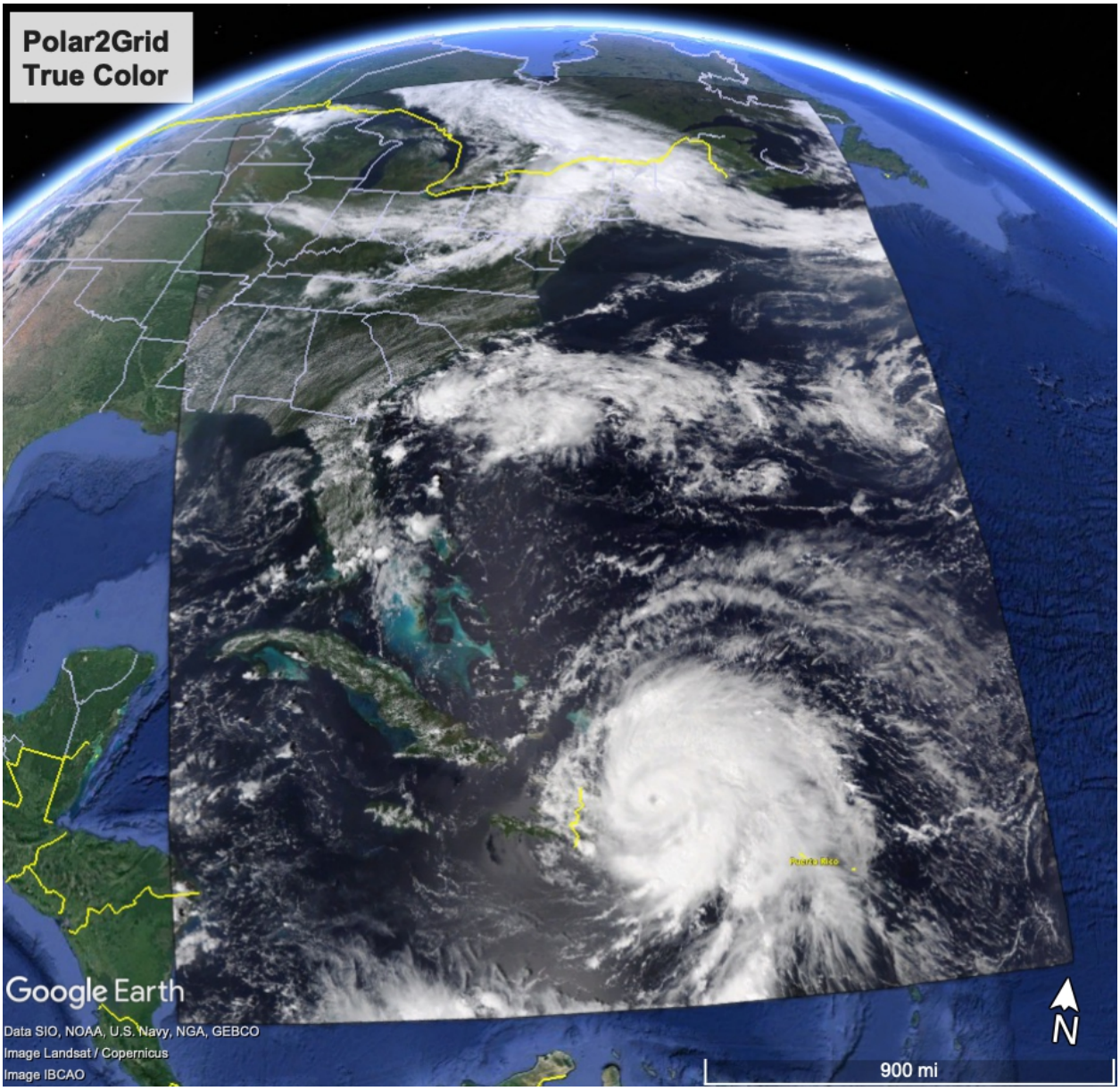


Fig. 9.3: VIIRS True color KMZ image displayed in the Google Earth Geobrowser.

9.2 Creating MODIS AWIPS Compatible Files

This example walks through the creation of MODIS NetCDF files for display in AWIPS.

9.2.1 Basic MODIS Level 1B AWIPS compatible file creation

Find the options available when creating MODIS AWIPS files:

```
polar2grid.sh -r modis -w awips_tiled -h
```

List the products that can be created from your MODIS L1B dataset. For the new Sectorized Cloud and Moisture Imagery (SCMI) AWIPS writer, include the sector name (see Section 6.1.3) either Lambert Conformal Conic (LCC), Pacific, Mercator, or Polar:

```
polar2grid.sh -r modis -w awips_tiled --sector-id LCC --list-products
-f <path_to_files>
```

Follow the command below to create MODIS AWIPS NetCDF files of all Level 1B products found in your data set for your sector. When using the `awips_tiled` scmi server, it is advised that a specific grid be chosen, and that the `--letters` and `--compress` options are used. In our LCC example, we will use the 1km grid:

```
polar2grid.sh -r modis -w awips_tiled --sector-id LCC --letters
--compress -g lcc_conus_1km -f <path_to_files>
```

Create a subset of MODIS reprojected AWIPS products for a specfic AWIPS grid:

```
polar2grid.sh -r modis -w awips_tiled -p bt27 vis02 --sector-id LCC --letters --
↪compress -g lcc_conus_1km -f <path_to__files>
```

Create true color and false color Aqua MODIS AWIPS NetCDF files from the 1000m, 500m, 250m and geolocation pass files acquired on 16 October 2022 at 20:52 UTC, reprojected onto the LCC 300m lettered grid.

```
polar2grid.sh -r modis -w awips_tiled --awips-true-color --awips-false-color --sector-
↪id LCC --letters --compress -g lcc_conus_300 -f l1b/a1.22289.2052.1000m.hdf  l1b/a1.
↪22289.2052.250m.hdf  l1b/a1.22289.2052.500m.hdf  l1b/a1.22289.2052.geo.hdf
```

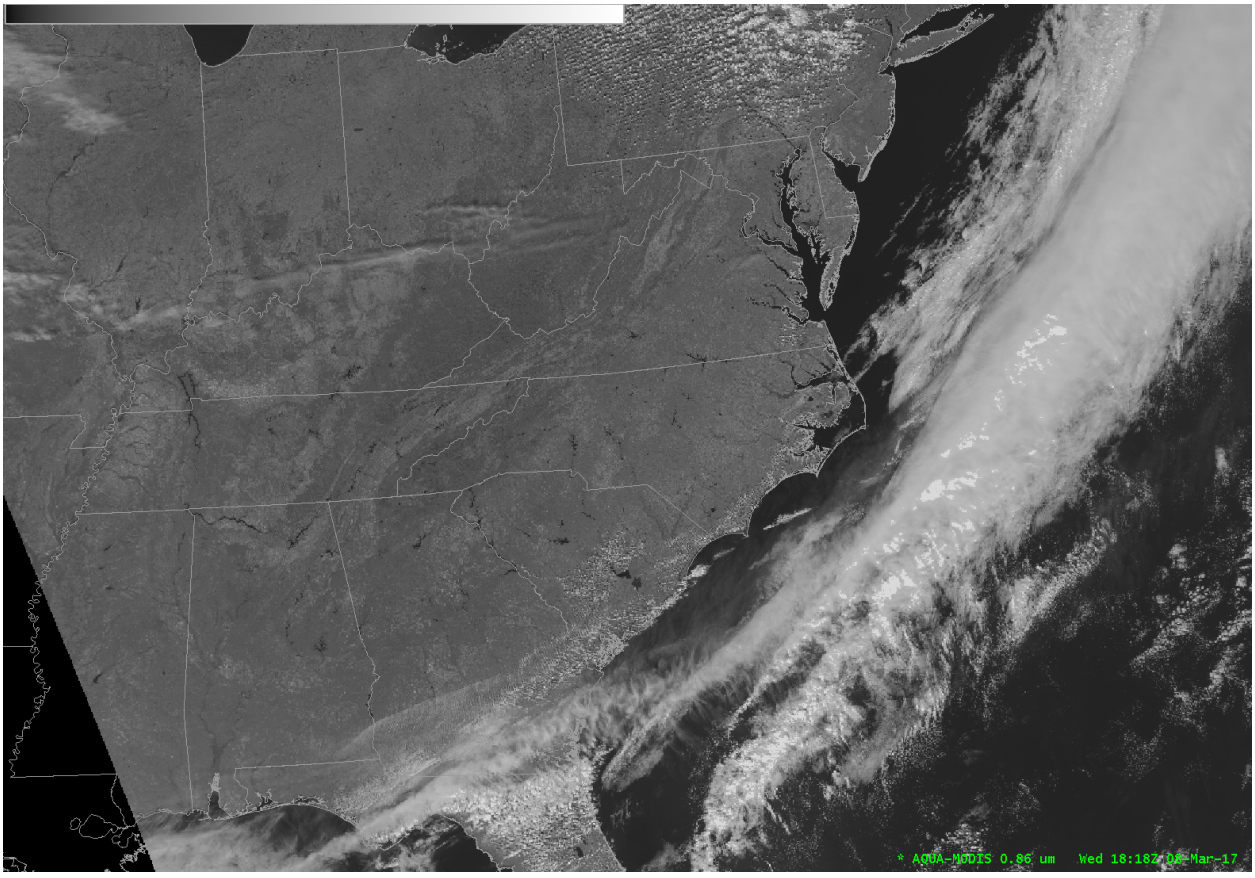


Fig. 9.4: AWIPS display of Aqua MODIS Band 2 (.86 micron) reflectances from 20:52 UTC, 16 October 2022.

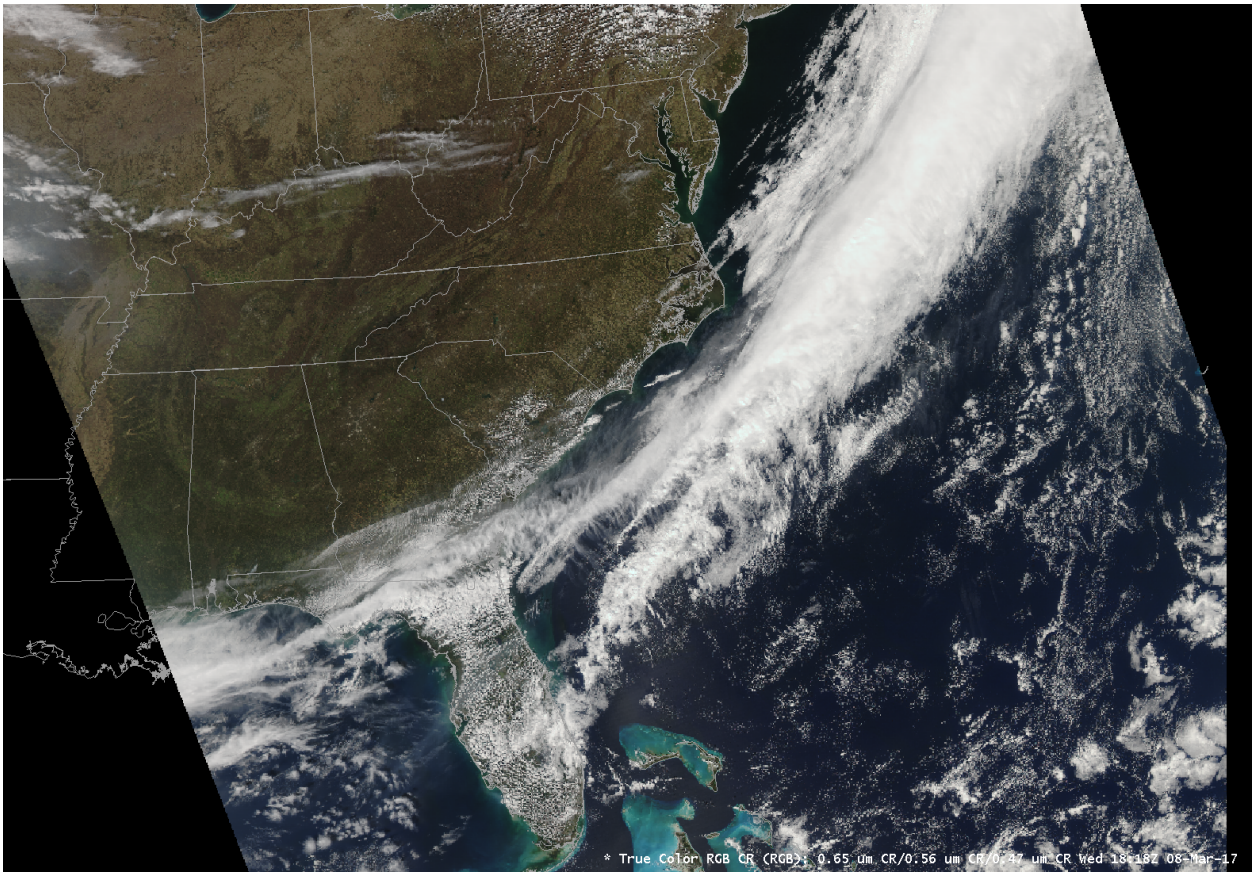


Fig. 9.5: AWIPS display of Polar2Grid MODIS corrected reflectances combined to create a 24 bit true color image. Data was collected from a Aqua MODIS pass at 20:52 UTC, 16 October 2022.

9.3 Creating ACSPO SST Reprojections

This set of examples demonstrates how you can create high quality Sea Surface Temperature (SST) color enhanced images using a number of the functionalities available in Polar2Grid.

9.3.1 Creating ACSPO GeoTIFF files

Find the options available for creating ACSPO VIIRS SST GeoTIFFs:

```
polar2grid.sh -r acspo -w geotiff -h
```

List the products that can be created from your ACSPO NetCDF dataset:

```
polar2grid.sh -r acspo -w geotiff --list-products -f <path_to_l1b_file>
```

To create the default product image, which is *sst* taken from the *sea_surface_temperature* array in the ACSPO file, use the following command. The example data set is the NOAA-20 VIIRS direct broadcast overpass from 18:43 UTC, 10 August 2022. Since there is often cloud cover and land in your datasets, use the `--grid-coverage` option to bypass the requirement for 10% coverage of data in the output image. Also note that we are using `--fill-value 0` to make a one band output file with a black background.

```
polar2grid.sh -r acspo -w geotiff acspo gtiff --grid-coverage 0 --fill-value 0 \
  -f viirs/20220810184327-CSPP-L2P_GHRSST-SSTskin-VIIRS_N20-ACSPO_V2.80-v02.0-fv01.0.
↪nc
```

The data set is re-projected into the WGS84 (Platte Carrée) projection by default. The image scaling is defined in the `generic.yaml` located in the `$POLAR2GRID_HOME/libexec/python_runtime/etc/polar2grid/enhancements` directory. This file contains product scaling information for all data parameters supported by Polar2Grid. It replaces the `rescale.ini` file that was used in previous versions of Polar2Grid.

The default scaling used for the ACSPO Version 2.80 SST files can be found under *sea_surface_temperature4 sea_surface_subskin_temperature* which is taken from the ACSPO array *standard_name* attribute. The section of the generic.yaml file that references our SST product is listed below.

```
395   sea_surface_temperature4:
396     standard_name: sea_surface_subskin_temperature
397     operations:
398       - name: linear_stretch
399         method: !!python/name:satpy.enhancements.stretch
400         kwargs: {stretch: 'crude', min_stretch: 267.317, max_stretch: 309.816}
```

This is used in the Polar2Grid software to define the range of brightness values in the output GeoTIFF file (0-255) to the temperatures they represent - in this case 267.317 K to 309.816 K. There are a number of different sea surface temperature arrays defined in the the generic.yaml file that allow Polar2Grid to support previous versions of the ACSPO SST files.

The scaling is done linearly. The output greyscale image below shows the VIIRS I-Band 2 (.86 micron) Reflectances on the left, and the ACSPO SST VIIRS image on the right.

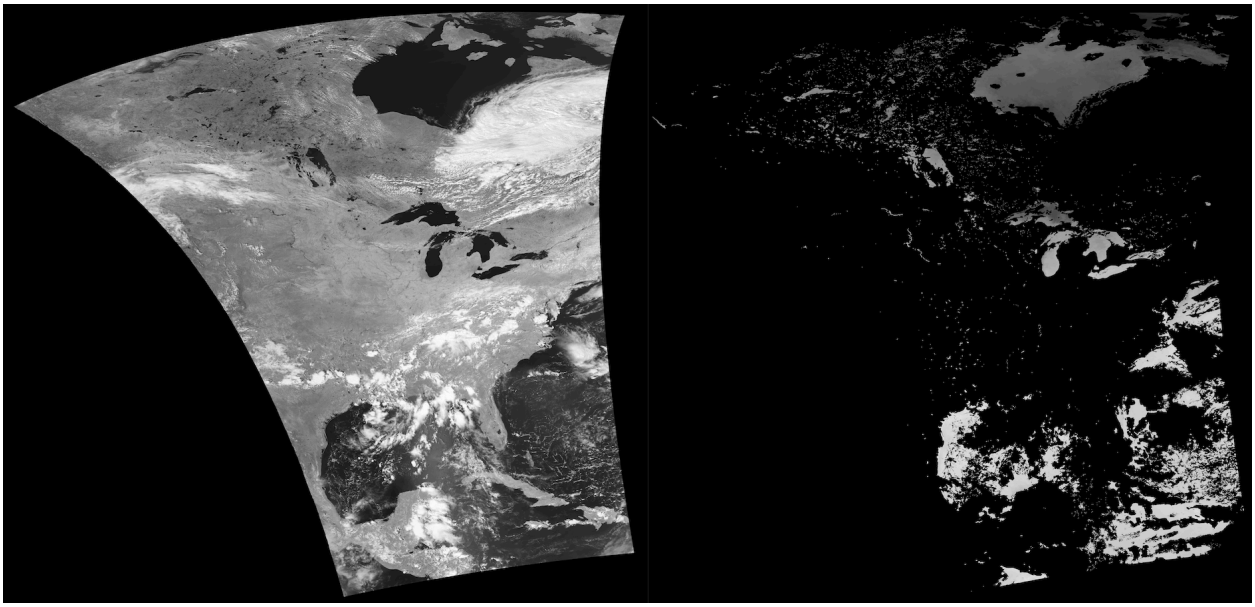


Fig. 9.6: NOAA-20 VIIRS I-Band 02 Reflectance image (Left panel) and ACSPO Sea Surface Temperature image (Right Panel) from an entire direct broadcast pass acquired on 10 August 2022 over North America. The default projection is WGS84 (Platte Carrée) and the default scaling is greyscale brightness values 0-255.

Now I would like to create an image cutting out a subset of this pass over the Great Lakes in the Northern United States. To do this, I need to create a new grid. I will use the *Defining Your Own Grids (Grid Configuration Helper)* script to do this.

```
p2g_grid_helper.sh great_lakes -83.5 45.1 750 750 1800 1200
```

I named my grid *great_lakes*, centered it on *-83.5 E Longitude* and *45.1 N Latitude*, with *750 m* spatial resolution in the X and Y directions, and defined the output grid to be *1800 x 1200* elements and lines.

Executing this command results in the following grid definition:

```
great_lakes:
  projection:
    proj: lcc
    lat_1: 45.1
    lat_0: 45.1
    lon_0: -83.5
    datum: WGS84
    units: m
    no_defs: null
    type: crs
  shape:
    height: 1200
    width: 1800
  center:
    x: -83.5
    y: 45.1
    units: degrees
  resolution:
    dx: 750.0
    dy: 750.0
```

I store this grid in an ASCII text file named *my_grid.yaml*, which I can provide to *polar2grid.sh* to create an image over my subset region by executing this command:

```
polar2grid.sh -r acspo -w geotiff --grid-coverage 0 --grid-configs my_grid.yaml \
  -g great_lakes --fill-value 0 -f viirs/*.nc
```

Note that you need to provide the full path to the *my_grid.yaml* if it is not located in the execution directory. The subset image that is created from executing this command is shown below.

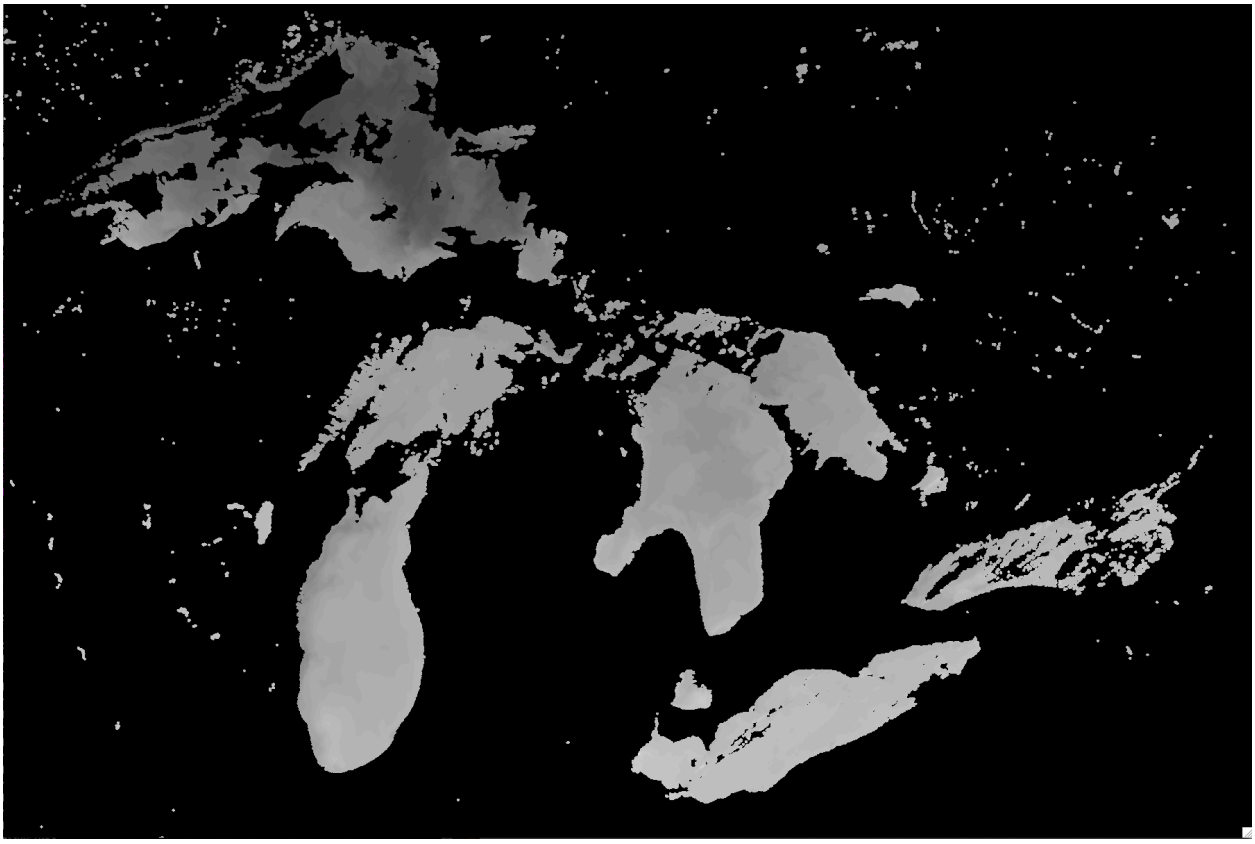


Fig. 9.7: NOAA-20 VIIRS ACSPO SST subset image for our defined grid over the great lakes.

To add a color enhancement to this image, I use the *add_colormap.sh* utility script and a rainbow color table *p2g_sst_palette.txt* that is included as part of the Polar2Grid package. This table is formatted as described in the *Add Colormap* section. You can view the file online.

This colormap will assign a color value to each of the 0-255 brightness range in the GeoTIFF image. Again, the default brightness range is associated with a temperature range of 267.317 K to 309.816 K.

```
add_colormap.sh          $POLAR2GRID_HOME/colormaps/p2g_sst_palette.txt
noaa20_viirs_sst_20220810_184327_great_lakes.tif
```

The filename will not change, but a color enhancement will be added to the image as shown below.

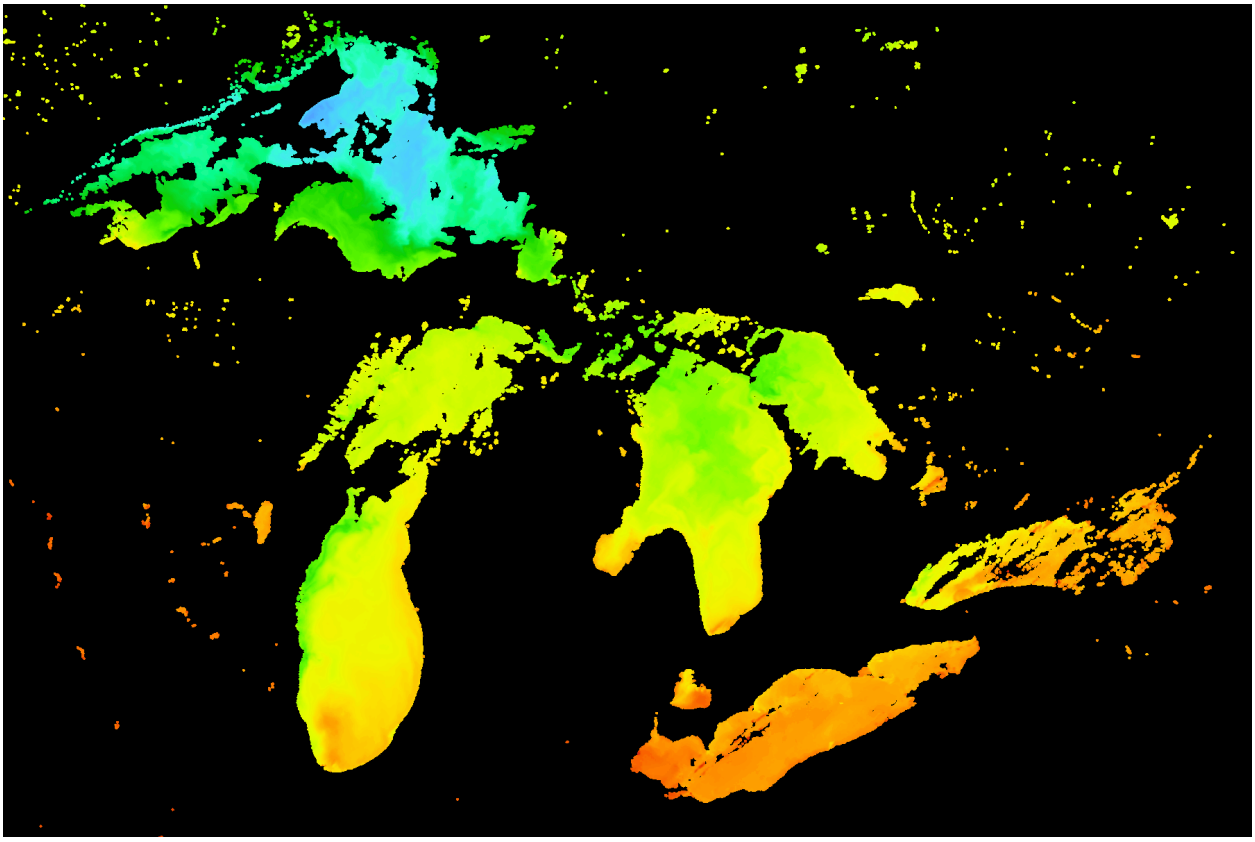


Fig. 9.8: NOAA-20 VIIRS ACSPO SST color enhanced image for our defined grid over the great lakes.

We can tighten the temperature range that is displayed in our region of interest by creating our own local rescaling. This allows us to use the full range of brightness values. In order to do this, I need to create a new rescaling *yaml* file that I will then provide to polar2grid.sh.

I chose an enhancment name of *great_lakes_sst* and will use the same standard_name of *sea_surface_subskin_temperature* and then redefined the relationship between the brightness values and the data. I tighten the temperature range to be between 275.0 K and 305.0 K. The contents of my new rescale yaml file is shown below (*my_rescale.yaml*).

```
enhancements:
  great_lakes_sst:
    standard_name: sea_surface_subskin_temperature
    operations:
      - name: linear_stretch
        method: !!python/name:satpy.enhancements.stretch
        kwargs: {stretch: 'crude', min_stretch: 275.0, max_stretch: 305.0}
```

I can then apply this new rescaling by referencing the file in the *polar2grid.sh* execution. In the example below, *my_rescale.yaml* file is located in the execution directory. If it is not, you will need to provide the full path to the file. The original *noaa20_viirs_sst_20220810_184327_great_lakes.tif* file will be overwritten by using this command.

```
polar2grid.sh -r acspo -w geotiff --extra-config-path my_rescale.yaml \
  --grid-coverage 0 --grid-configs my_grid.yaml -g great_lakes \
  --fill-value 0 -f  viirs/*.nc
```

The result of applying this rescaling to my image and applying my colormap is shown below.

To further enhance this ACSPO SST image I can add a color bar using the *add_coastlines.sh* script. There are many

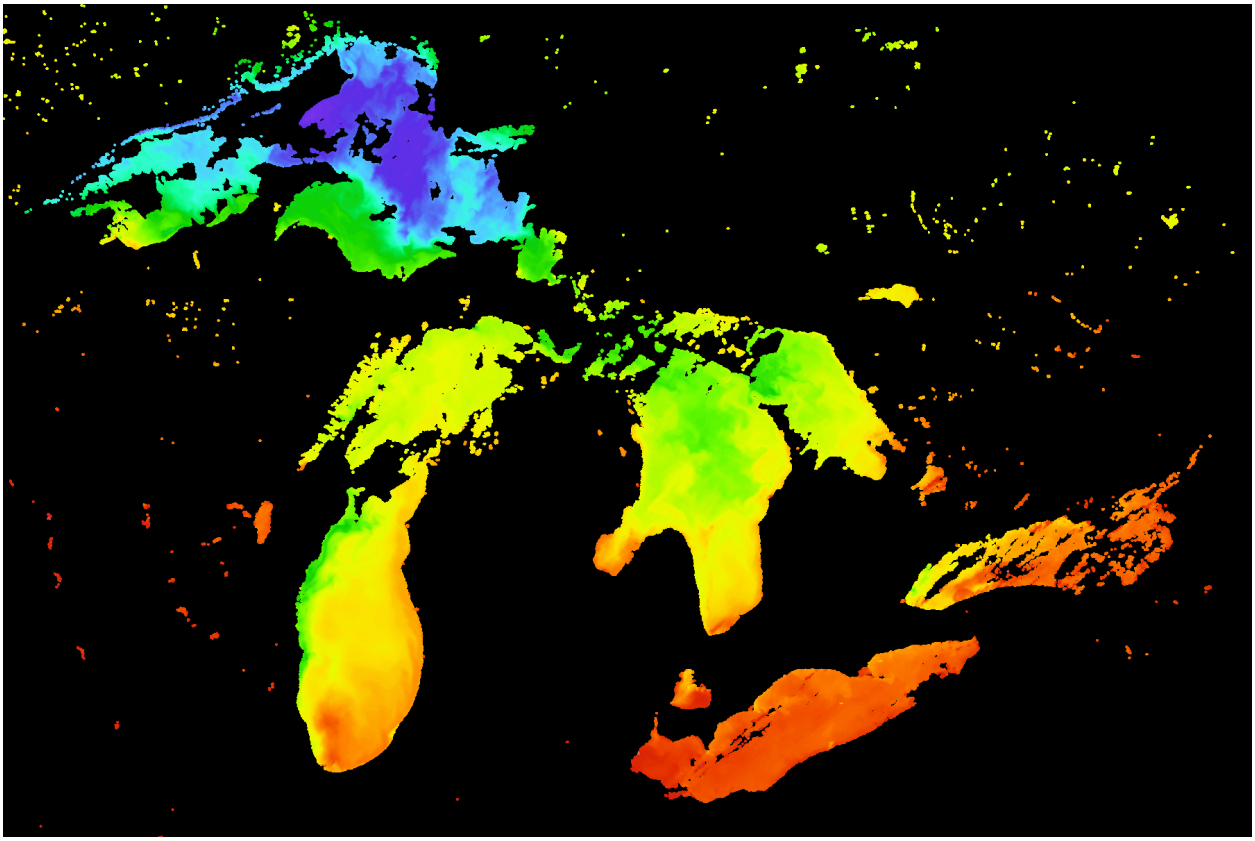


Fig. 9.9: S-NPP VIIRS ACSPO SST color enhanced subset image over our area of interest using a customized rescaling that linearly maps brightness values of 0-255 to a temperature range of 275.0 K to 305.0 K.

options to this script all of which are listed in the *Add Overlays (Borders, Coastlines, Grids Lines, Rivers)* section. Users can control the location and size of the color bar, a color bar title, fonts, etc. The script overlays the color bar and text onto the image storing it as a .png file.

For example, executing the following command:

```
add_coastlines.sh noaa20_viirs_sst_20220810_184327_great_lakes.tif \
  --add-colorbar --colorbar-text-color "white" \
  --colorbar-units "°K" --colorbar-align bottom --colorbar-text-size=20 \
  --colorbar-title "VIIRS ACSPO SST 10 August 2022 18:43 UTC" \
  --colorbar-height 35 --colorbar-tick-marks 4
```

results in the creation of the file *noaa20_viirs_sst_20220810_184327_great_lakes.png* as displayed below.

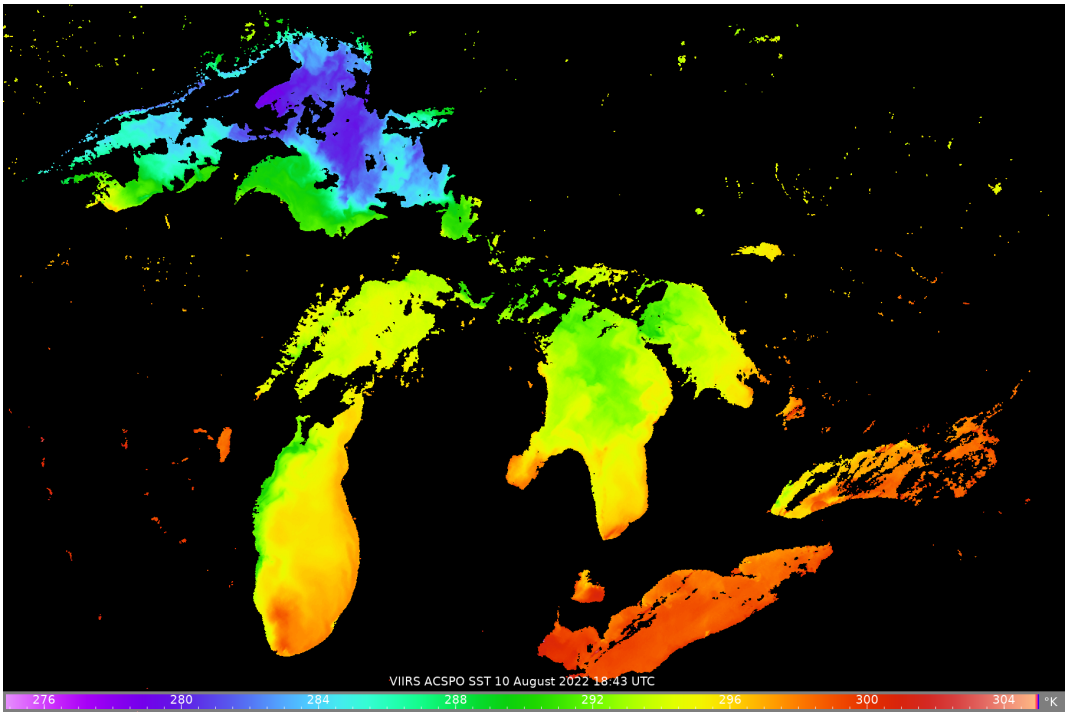


Fig. 9.10: S-NPP VIIRS ACSPO SST color enhanced subset image over the great lakes using a customized rescaling that linearly maps brightness values of 0-255 to a temperature range of 275.0 K to 305.0 K including a overlaid color table.

If you wanted to create a display using a more familiar SST temperature scale such as Celsius or Fahrenheit, you can do that by using the `--colorbar-min` and `--colorbar-max` options to *add_coastlines.sh*. This will not change any data values in the file, but it will change the color table display. For example, I have set the dataset range in my file to be 275.0 K to 305.0 K. This is equivalent to a range in Degrees Celsius of 1.85 C to 31.85 C. So by executing the following command, I can display the image with a color bar in Degrees Celsius.

```
add_coastlines.sh noaa20_viirs_sst_20220810_184327_great_lakes.tif \
  --add-colorbar --colorbar-text-color "white" \
  --colorbar-units "°C" --colorbar-align bottom --colorbar-min 1.85 \
  --colorbar-max 31.85 --colorbar-tick-marks 5 --colorbar-text-size=20 \
  --colorbar-title "VIIRS ACSPO SST 10 August 2022 18:43 UTC" \
  --colorbar-height 35
```

I can perform a similar conversion of the temperature range to Degrees Fahrenheit and create an image with a color bar labeled in those units.

```
add_coastlines.sh noaa20_viirs_sst_20220810_184327_great_lakes.tif \
  --add-colorbar --colorbar-text-color "white" \
  --colorbar-units "°F" --colorbar-align bottom --colorbar-min 35.33 \
  --colorbar-max 89.33 --colorbar-tick-marks 5 --colorbar-text-size=20 \
  --colorbar-title "VIIRS ACSPO SST 10 August 2022 18:43 UTC" \
  --colorbar-height 35
```

I can also use the same add_coastlines.sh command to overlay maps including borders and latitiude longitude grids. For example, if I execute the command,

```
add_coastlines.sh noaa20_viirs_sst_20220810_184327_great_lakes.tif \
  --add-borders --borders-level 3 --borders-outline gray --borders-width 2 \
  --borders-resolution h --add-colorbar --colorbar-text-color "white" \
  --colorbar-units "°C" --colorbar-align bottom --colorbar-min 1.85 \
  --colorbar-max 31.85 --colorbar-tick-marks 5 --colorbar-text-size=20 \
  --colorbar-title "VIIRS ACSPO SST 10 August 2022 18:43 UTC" \
  --colorbar-height 35
```

it will result in the creation of the final image product that is a re-gridded, re-scaled, color enhanced image with a color bar labeled in Degrees Celsius and border overlays.

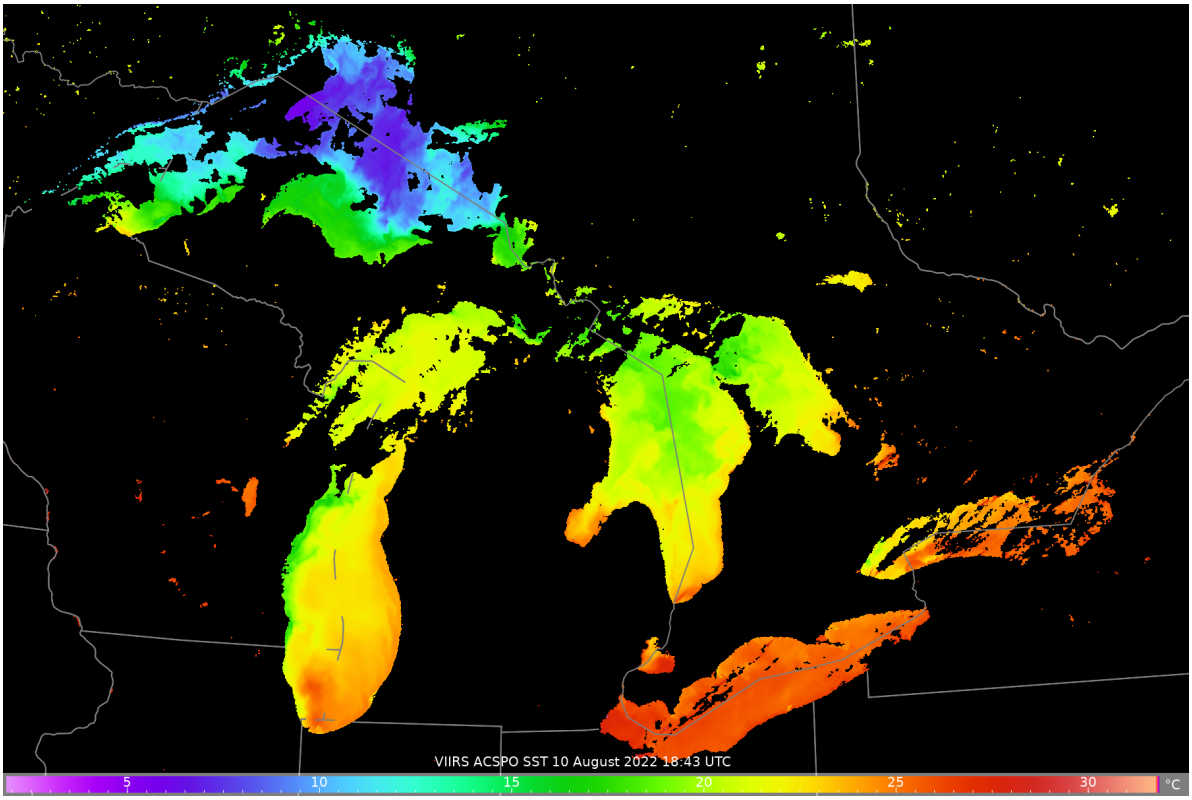


Fig. 9.11: Final S-NPP VIIRS ACSPO SST image created from data acquired by direct broadcast on 10 August 2022 beginning at 18: UTC. The image creation includes re-gridding, re-scaling, color enhanced with color table and map overlays.

9.4 Creating AMSR2 Reprojections

This example walks through some common tasks when working with GCOM-W1 AMSR2 Level 1B data.

9.4.1 Creating AMSR2 GeoTIFF files

Find the options available for creating AMSR2 Level 1B GeoTIFFs:

```
polar2grid.sh -r amsr2_l1b -w geotiff -h
```

List all of the products that can be created from your AMSR2 HDF5 dataset:

```
polar2grid.sh -r amsr2_l1b -w geotiff --list-products-all -f
<path_to_l1b_file>
```

To create AMSR2 GeoTIFF files of all default bands found in your data set and reprojected to the default Platte Carrée projection:

```
polar2grid.sh -r amsr2_l1b -w geotiff -f <path_to_l1b_file>
```

Create the default set of GeoTIFF images for the AMSR2 Level 1B file acquired on 10 September 2022, at 23:35 UTC:

```
polar2grid.sh -r amsr2_l1b -w geotiff --fill-value 0 -f GW1AM2_202209102335_181A_
↪L1DLBTBR_1110110.h5
```

Executing this command produces these files in WGS84 (Platte Carrée) projection:

```
gcom-w1_amsr2_btemp_36.5h_20220910_233500_wgs84_fit.tif
gcom-w1_amsr2_btemp_36.5v_20220910_233500_wgs84_fit.tif
gcom-w1_amsr2_btemp_89.0ah_20220910_233500_wgs84_fit.tif
gcom-w1_amsr2_btemp_89.0av_20220910_233500_wgs84_fit.tif
gcom-w1_amsr2_btemp_89.0bh_20220910_233500_wgs84_fit.tif
gcom-w1_amsr2_btemp_89.0bv_20220910_233500_wgs84_fit.tif
```

The GeoTIFF image for AMSR2 89.0ah GHz band is displayed below.

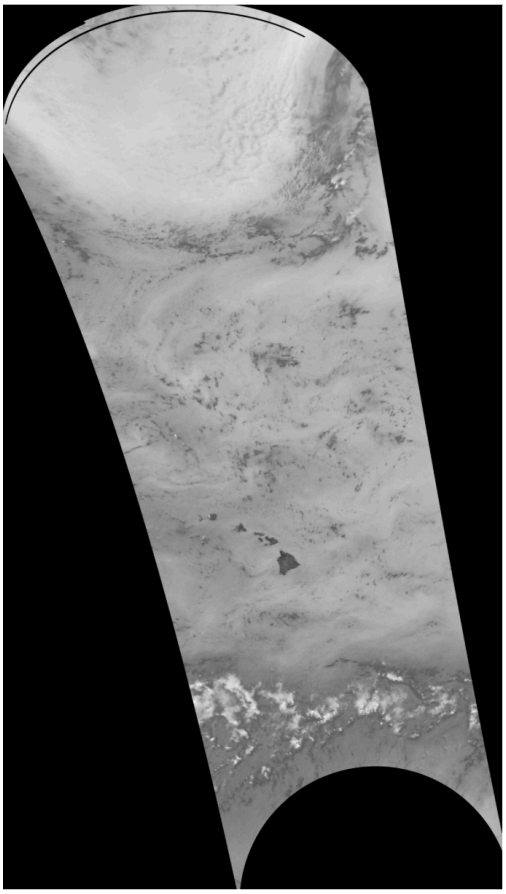


Fig. 9.12: GCOMW-1 AMSR2 L1B 89.0ah GHz brightness temperatures using default scaling. Data set was observed at 23:35 UTC on 10 Setember 2022.

9.4.2 Naval Research Lab (NRL) Image Reproductions

Polar2Grid inclues the capability to reproduce the AMSR2 color enhanced images staged on the the Naval Research Lab (NRL) tropical cyclone page: http://www.nrlmry.navy.mil/TC.html

First, create a reprojected GeoTIFF in Lambert Conic Conformal (LCC) projection and rescale the data. The data in this example is from 10 September 2022. We are pointing to the rescale information that is stored in the *$POLAR2GRID_HOME/example_enhancements/amsr2_png/enhancements/generic.yaml* file. This will produce a linear scaled output of data rangeing from 180.0 K to 280.0 K brightness temperatures for our default products.

```
polar2grid.sh -r amsr2_l1b -w geotiff --extra-config-path $POLAR2GRID_HOME/example_
↪enhancements/amsr2_png -g lcc_fit --fill-value 0 -f GW1AM2_202209102335_181A_
↪L1DLBTBR_1110110.h5
```

Executing this command produces these AMSR2 LCC GeoTIFF files:

```
gcom-w1_amsr2_btemp_36.5h_20220910_233500_lcc_fit.tif
gcom-w1_amsr2_btemp_36.5v_20220910_233500_lcc_fit.tif
gcom-w1_amsr2_btemp_89.0ah_20220910_233500_lcc_fit.tif
gcom-w1_amsr2_btemp_89.0av_20220910_233500_lcc_fit.tif
gcom-w1_amsr2_btemp_89.0bh_20220910_233500_lcc_fit.tif
gcom-w1_amsr2_btemp_89.0bv_20220910_233500_lcc_fit.tif
```

Once the data has been rescaled, you are ready to apply the NRL colormaps to the data. In this example we are using the 89A/H GHz file.

```
add_colormap.sh $POLAR2GRID_HOME/libexec/python_runtime/etc/polar2grid/colormaps/
↪amsr2_89h.cmap gcom-w1_amsr2_btemp_89.0ah_20220910_233500_lcc_fit.tif
```

This command adds the enhancement to the original GeoTIFF. The rescaled and final color enhanced product are shown below:

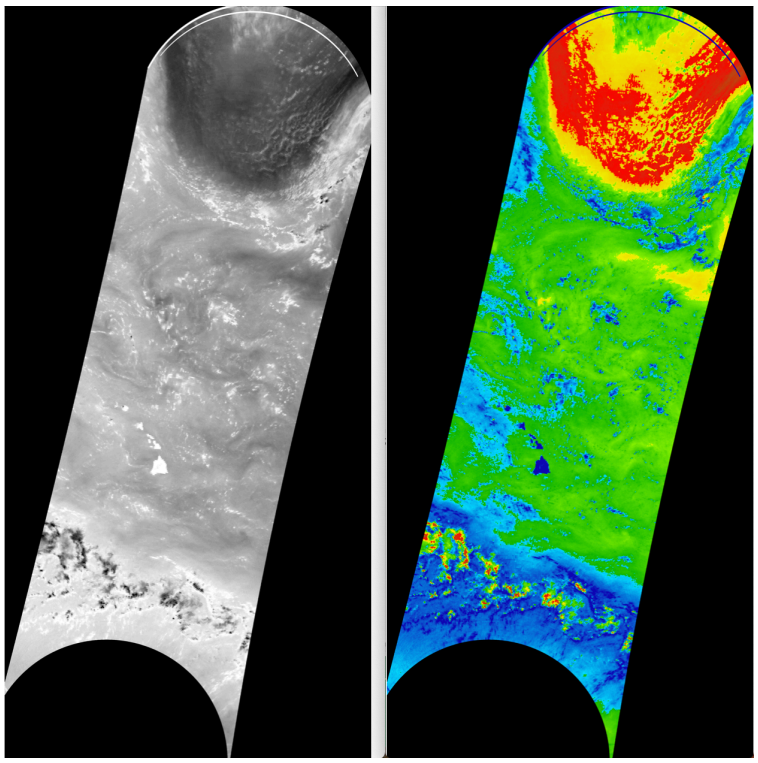


Fig. 9.13: GCOMW-1 AMSR2 L1B 89.0A/H GHz brightness temperatures reprojected in Lambert Conic Conformal Projection and rescaled (left), and with a color table applied (right) using the Naval Research Lab color ehancement. The data set was collected at 23:35 UTC on 10 September 2022.

CHAPTER

TEN

GRIDS

Polar2Grid allows users to remap to one or more projected grids. A grid defines the uniform geographic area that an output image covers. Polar2Grid comes with various grids to choose from that should suit most users and their use cases. Some grids are provided for specific writers (like Tiled AWIPS), but can be used for other writers as well. Users can also specify their own custom grids. See the *Custom Grids* documentation for help with this.

10.1 Provided Grids

Below are descriptions for a few of the grids provided with Polar2Grid. For information on all of the grids provided by Polar2Grid see the Grids Configuration YAML File.

The grids' projections are defined using PROJ.4. Go to the PROJ documentation for more information on what each projection parameter means.

Note: If the grid does not have a parameter specified it will be derived from the data during remapping. This allows for grids that fit to the data (dynamic grids).

10.1.1 WGS84 Dynamic Fit

Grid Name
wgs84_fit

Description
Longitude/Latitude WGS84 Grid

Projection
EPSG:4326

Resolution
0.0057 degrees

10.1.2 WGS84 Dynamic Fit 250m

Grid Name
wgs84_fit_250

Description
Longitude/Latitude WGS84 Grid at ~250m resolution

Projection
EPSG:4326

Resolution
0.00225 degrees

10.1.3 Lambert Conic Conformal Dynamic Fit

Grid Name
lcc_fit

Description
1km East CONUS centered lcc grid (alias: lcc_na)

Projection
+proj=lcc +lat_1=25 +lat_0=25 +lon_0=-95 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
1000.0 meters

10.1.4 Lambert Conic Conformal - South America Centered

Grid Name
lcc_sa

Description
1km South America centered LCC grid

Projection
+proj=lcc +lat_1=-25 +lat_0=-25 +lon_0=-55 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
1000.0 meters

10.1.5 Lambert Conic Conformal - Europe Centered

Grid Name
lcc_eu

Description
1km Europe centered LCC grid

Projection
+proj=lcc +lat_1=25 +lat_0=25 +lon_0=15 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
1000.0 meters

10.1.6 Lambert Conic Conformal - South Africa Centered

Grid Name
lcc_south_africa

Description
1km South Africa centered LCC grid

Projection
+proj=lcc +lat_1=-25 +lat_0=-25 +lon_0=25 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
1000.0 meters

10.1.7 Lambert Conic Conformal - Australia Centered

Grid Name
lcc_aus

Description
1km Australia centered LCC grid

Projection
+proj=lcc +lat_1=-25 +lat_0=-25 +lon_0=135 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
1000.0 meters

10.1.8 Lambert Conic Conformal - Asia Centered

Grid Name
lcc_asia

Description
1km Asia centered LCC grid

Projection
+proj=lcc +lat_1=25 +lat_0=25 +lon_0=105 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
1000.0 meters

10.1.9 High Resolution Lambert Conic Conformal Dynamic Fit

Grid Name
lcc_fit_hr

Description
400m East CONUS centered LCC grid

Projection
+proj=lcc +lat_1=25 +lat_0=25 +lon_0=-95 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
400.0 meters

10.1.10 Equirectangular Fit

Grid Name
eqc_fit

Description
250m Equirectangular grid

Projection
+proj=eqc +lat_ts=0 +lat_0=0 +lon_0=0 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
250.0 meters

10.1.11 Polar-Stereographic Canada

Grid Name
polar_canada

Description
1km Polar-stereographic Canada centered grid

Projection
+proj=stere +lat_0=90 +lat_ts=45 +lon_0=-150 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
1000.0 meters

10.1.12 Polar-Stereographic North Pacific

Grid Name
polar_north_pacific

Description
400m Polar-stereographic North Pacific centered grid

Projection
+proj=stere +lat_0=90 +lat_ts=45 +lon_0=-170 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
400.0 meters

10.1.13 Polar-Stereographic South Pacific

Grid Name
polar_south_pacific

Description
400m Polar-stereographic South Pacific centered grid

Projection
+proj=stere +lat_0=-90 +lat_ts=-45 +lon_0=-170 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
400.0 meters

10.1.14 Polar-Stereographic Russia

Grid Name
polar_russia

Description
400m Polar-stereographic Russia centered grid

Projection
+proj=stere +lat_0=90 +lat_ts=45 +lon_0=50 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
400.0 meters

10.1.15 Polar-Stereographic Alaska

Grid Name
polar_alaska

Description
400m Polar-stereographic Alaska centered grid

Projection
+proj=stere +lat_0=90 +lat_ts=60 +lon_0=-150 +datum=WGS84 +units=m +no_defs +type=crs

Resolution
400.0 meters

10.1.16 GOES-East 1km

Grid Name
goes_east_1km

Description
GOES-East 1km Full Disk Grid

Projection
+proj=geos +sweep=x +lon_0=-75 +h=35786023 +ellps=GRS80 +units=m +no_defs +type=crs

Extent
[-5434894.885056, -5434894.885056, 5434894.885056, 5434894.885056]

10.1.17 GOES-East 4km

Grid Name
goes_east_4km

Description
GOES-East 4km Full Disk Grid

Projection
+proj=geos +sweep=x +lon_0=-75 +h=35786023 +ellps=GRS80 +units=m +no_defs +type=crs

Extent
[-5434894.885056, -5434894.885056, 5434894.885056, 5434894.885056]

10.1.18 GOES-East 8km

Grid Name
goes_east_8km

Description
GOES-East 8km Full Disk Grid

Projection
+proj=geos +sweep=x +lon_0=-75 +h=35786023 +ellps=GRS80 +units=m +no_defs +type=crs

Extent
[-5434894.885056, -5434894.885056, 5434894.885056, 5434894.885056]

10.1.19 GOES-East 10km

Grid Name
goes_east_10km

Description
GOES-East 10km Full Disk Grid

Projection
+proj=geos +sweep=x +lon_0=-75 +h=35786023 +ellps=GRS80 +units=m +no_defs +type=crs

Extent
[-5434894.885056, -5434894.885056, 5434894.885056, 5434894.885056]

10.1.20 GOES-West 1km

Grid Name
goes_west_1km

Description
GOES-West 1km Full Disk Grid

Projection
+proj=geos +sweep=x +lon_0=-137 +h=35786023 +ellps=GRS80 +units=m +no_defs +type=crs

Extent
[-5434894.885056, -5434894.885056, 5434894.885056, 5434894.885056]

10.1.21 GOES-West 4km

Grid Name
goes_west_4km

Description
GOES-West 4km Full Disk Grid

Projection
+proj=geos +sweep=x +lon_0=-137 +h=35786023 +ellps=GRS80 +units=m +no_defs +type=crs

Extent
[-5434894.885056, -5434894.885056, 5434894.885056, 5434894.885056]

10.1.22 GOES-West 8km

Grid Name
goes_west_8km

Description
GOES-West 8km Full Disk Grid

Projection
+proj=geos +sweep=x +lon_0=-137 +h=35786023 +ellps=GRS80 +units=m +no_defs +type=crs

Extent
[-5434894.885056, -5434894.885056, 5434894.885056, 5434894.885056]

10.1.23 GOES-West 10km

Grid Name
goes_west_10km

Description
GOES-West 10km Full Disk Grid

Projection
+proj=geos +sweep=x +lon_0=-137 +h=35786023 +ellps=GRS80 +units=m +no_defs +type=crs

Extent
[-5434894.885056, -5434894.885056, 5434894.885056, 5434894.885056]

CHAPTER

ELEVEN

CUSTOM GRIDS

Polar2Grid provides a set of grids to suit most use cases, but sometimes these grids are not enough. This is why Polar2Grid allows users to create their own custom grids.

Grids can be static, meaning the grid definition specifies the projection, pixel size, origin, and grid size. Grids can also be dynamic, meaning that only some grid defining parameters are specified. An example of a dynamic grid is the *WGS84 Dynamic Fit* grid. This grid does not have an origin or grid size specified, which tells the remapping components of Polar2Grid to calculate these values from the data.

11.1 Adding your own grid

If you wish to add your own grids as a replacement for or in addition to the provided set you'll have to make your own grid configuration file. The instructions below describe how to create your own configuration file and how it can be provided to `polar2grid.sh`:

1. Create a text file named something ending in ".yaml" (ex. "my_grids.yaml"). Open it for editing. The package includes a `grid_configs` directory where user configuration files can be stored.
2. Add an entry to this file for each grid you would like to add Polar2Grid. Follow the *Grid Configuration File Format* section below. The grid file is in the YAML text format.
3. Call the `polar2grid.sh` script and add the command line option `--grid-configs grids.conf <your-file.yaml>`. The builtin grids in Polar2Grid are included when "grids.conf" is provided. If you would like only your grids and not the Polar2Grid provided grids don't include the "grids.conf" in the command line option.

Polar2Grid also includes a simple script that can generate the required YAML text when provided with general information about the grid you wish to create. See the *Defining Your Own Grids (Grid Configuration Helper)* section.

Note: Configuration files are loaded in the order specified. If a grid name is used more than once, the last one loaded is used.

11.2 Grid Configuration File Format

Note: The legacy ".conf" format is still supported for backwards compatibility, but should not be used for new grid definition files.

Example Grid Configuration File: `grid_example.yaml`

Grid configuration files follow the format used by the Satpy and Pyresample Python libraries in their areas.yaml files and are in the YAML text format. Comments can be added by prefixing lines with a # character. There is an example file provided in the Polar2Grid bundle at:

`$POLAR2GRID_HOME/grid_configs/grid_example.yaml`

Grids can be dynamic or static. Dynamic grids have some amount of information unspecified that will be filled in later at runtime using the provided input geolocation data. The most common case for a dynamic grid is specifying only "resolution", but not "shape" or any extent information. If enough information is provided in the definition then a static grid is created which will always be in the same location at the same resolution, but will process faster as the other grid parameters don't need to be computed.

If you are unfamiliar with projections, try the *Defining Your Own Grids (Grid Configuration Helper)* script. One example of a grid is shown below.

```
my_211e:
  description: 'My LCC grid'
  projection:
    proj: lcc
    lat_1: 25
    lat_0: 25
    lon_0: -95
    R: 6371200
    units: m
    no_defs: null
    type: crs
  shape:
    height: 5120
    width: 5120
  resolution:
    dy: 1015.9
    dx: 1015.9
  upper_left_extent:
    x: -122.9485839789149
    y: 59.86281930852158
    units: degrees
```

This static grid is named `my_211e` and has the following parameters:

1. **description**: Optional human-readable description of the grid. This is not currently used by Polar2Grid.
2. **projection**: PROJ.4 parameters of the projection of the grid. Can also be specified as a string. Or as an EPSG code integer. In addition to the example grids file linked above, for more information on possible parameters see the PROJ documentation.
3. **shape**: Number of pixels in each dimension.
4. **resolution**: Resolution of each pixel in projection units (usually meters). This can also be specified in degrees by adding a `units: degrees` in this section.

5. **upper_left_extent**: Location of the upper-left corner of the upper-left pixel of the grid. By default this is in projection units (usually meters), but is specified in degrees here with the extra `units:` parameter. Note this differs from the legacy `.conf` format which used the center of the upper-left pixel.

See the example grids file linked above for more examples and other available parameters like **center** or **area_extent**.

CHAPTER

TWELVE

IMAGE PROCESSING TECHNIQUES

Many composites in Polar2Grid take advantage of various corrections or adjustments to produce the best looking imagery possible. The below sections describe the corrections and other related topics used in Polar2Grid. See the various *Readers* documentation for more information on what products are available and descriptions of what corrections are used.

12.1 RGB Images

Satellite imagers can simultaneously observe the Earth in multiple spectral channels, while the human eye is sensitive to only the visible channels. By mapping data from imager channels to the visible red, green, and blue channels in different ways, we can produce "RGB" images that show the Earth as a human would see it from space ("true color"), or that emphasize certain features that can be detected using combinations of different channels ("false color").

Luminance (L), or single band, images are also used when displaying a single imager channel in grayscale. Another popular way of showing single imager channels is to apply a "colormap" to the data. In these cases, each data value of a single satellite imager channel is represented by a color. This is different than the RGB composites described above where multiple channels go into making a single color image.

Depending on the configuration and writer used, Polar2Grid may also add an additional "Alpha" channel (ex. RGBA) to an image. This Alpha channel is used to determine the opaqueness or transparency of an image. This is typically used in Polar2Grid to make invalid or missing data values transparent (completely opaque or completely transparent).

12.2 Solar Zenith Angle Modification

Reflectance is defined as the reflected radiation as a fraction of the incident radiation. To calculate reflectance, the solar zenith angle is needed, in addition to the radiance measured by the sensor. This modification, used by some RGB recipes, involves dividing the channel data by the cosine of the solar zenith angle.

12.3 Rayleigh Scattering Correction - CREFL

Due to the size of molecules that make up our atmosphere, some visible channel light is preferentially scattered more than others, especially at larger viewing angles. The Corrected Reflectance algorithm performs a simple atmospheric correction with MODIS visible, near-infrared, and short-wave infrared bands (1 to 16). Later versions of the software were adapted to work with VIIRS data. Both implementations have been merged and made available as a "modifier" in the Satpy Python library and used by Polar2Grid. This algorithm was originally developed by the MODIS Rapid Response Team (http://rapidfire.sci.gsfc.nasa.gov/) and made available by cooperative agreement, with subsequent additions by the University of South Florida (USF) and the NASA Direct Readout Laboratory (DRL).

The algorithm corrects for molecular (Rayleigh) scattering and gaseous absorption (water vapor, ozone) using climatological values for gas contents. It requires no real-time input of ancillary data. The algorithm performs no aerosol correction. The Corrected Reflectance products are very similar to the MODIS Land Surface Reflectance product (MOD09) in clear atmospheric conditions, since the algorithms used to derive both are based on the 6S Radiative Transfer Model (Vermote et al.1994). The products show differences in the presence of aerosols, however, because the MODIS Land Surface Reflectance product uses a more complex atmospheric correction algorithm that includes a correction for aerosols.

12.4 Rayleigh Scattering Correction - Pyspectral

Due to the size of molecules that make up our atmosphere, some visible channel light is preferentially scattered more than others, especially at larger viewing angles. One method to correct for this is implemented in the Pyspectral Python library. A detailed description of the algorithm used by Pyspectral and other features of the library can be found in the official Pyspectral documentation:

https://pyspectral.readthedocs.io/en/latest/rayleigh_correction.html

12.5 Ratio Sharpening

Some sensors include channels that measure radiance at the same wavelength, but at different spatial resolutions. When making an RGB image that uses one of these multi-resolution wavelengths combined with other channels that are only available at lower resolutions, we can use the multi-resolution channels to sharpen the other channels. For example, if the high-resolution channel is used for R, and lower resolution channels for G and B, we can do:

```
R_ratio = R_hi / R_lo
new_R = R_hi
new_G = G * R_ratio
new_B = B * R_ratio
```

By upsampling the lower resolution G and B channels and multiplying by the ratio of high and low resolution R channels, we can produce a sharper looking final image. That is, the lower resolution channels appear to have a better spatial resolution than they did originally.

12.6 Self Ratio Sharpening

Similar to the *Ratio Sharpening* described above, it is possible to apply a similar sharpening when one of the channels of the RGB is only provided in a high resolution. In this case, we can downsample the high resolution channel to the resolution of the other channels (averaging the pixels), then upsample the result again. By taking the ratio of the original high resolution and this averaged version, we can produce a ratio similar to that in the above ratio sharpening technique.

12.7 Non-linear True Color Scaling

As a final step for some RGB images, Polar2Grid scales the image values using a series of linear interpolation ranges to bring out certain regions of the image and lessen the effect of others. For lack of a better name, these multiple linear stretches make up an overall non-linear scaling. A typical scaling where reflectance data (0 - 1) has been multiplied by 255 (8-bit unsigned integer) would be:

Input Range	Output Range
0 - 25	0 - 90
25 - 55	90 - 140
55 - 100	140 - 175
100 - 255	175 - 255

APPENDIX

A

VERSION 2.3 TO VERSION 3.0 COMMAND CHANGES

A.1 Important Changes

- New basic implementation: `polar2grid.sh -r <reader> -w <writer>`.
  - For example: `polar2grid.sh -r viirs_sdr -w geotiff -f <path to files>`.
- Some reader and writer names have been changed.
  - For example: GeoTIFF writer is now `geotiff`.
- Improved execution speeds using the xarray and Dask python library.
- Option now available to choose how many worker threads to use: `--num-workers`. Default is 4.
- NOAA20 output file names standardized to "noaa20" prefix. For instance, `n20_viirs_sst*` is now `noaa20_viirs_sst*`.
- The crefl reader is no longer supported. Use `-p true_color false_color` with the MODIS or VIIRS readers.
- The scmi writer is replaced by `awips_tiled`.
- AWIPS true and false color tiles are created using `--awips-true-color --awips-false-color`.
- GeoTIFF output files now include an "Alpha" channel by default. To reproduce output from previous versions, use `--fill-value 0` command line option.
- `--list-products` and `list-product-all` now available. `--list-products-all` includes Satpy products that can be created in addition to standard Polar2Grid products. Please note that the additional products have not been tested.
- Rescale .ini files have been replaced by the .yaml format. If you required help in converting your custom .ini files, please contact us.
- Grid definition `.conf` files are transitioning to `.yaml` file definitions. A script which converts the grid files to the new format is now available.
- Output files with the same name will now be overwritten.
- Reflectances are now stored in GeoTIFF files (dtype float32) as 0-100 (v2.3 values were stored as 0-1).
- The dtype real4 format is replaced with dtype float32.
- The way that day/night product filtering is implemented has changed. The use of `--sza-threshold` is no longer used. Use the `--filter-day-products`.
- AWIPS output files are written in a new way. We have found some problems with the current version of AWIPS not displaying fill values correctly, and the units `Micron` is not recognized in active version of AWIPS at the time of writing. These bugs have been fixed and will eventually be resolved in future AWIPS versions.

- VIIRS false color image creating now uses the .86 micron band for sharpening. Previous versions used the .68 micron for sharpening.
- Different option for nearest neighbor resampling (`--method nearest`).
  - `--radius-of-influence` in meters replaces `--distance-upper-bound` in units of grid cell.
- Different option names for elliptical weighted averaging resampling (`--method ewa`)
  - `--weight-delta-max` (replaces `--fornav-D` option)
  - `--weight-distance-max` (replaces `--fornav-d` option)
- Reflectances in HD5 files and binary files are now stored as 0-100%?
- The standard convention for grid configuration is now .yaml file formatting. The legacy .conf files can still be used. A script was made to convert .conf style to .yaml style grid configuration.

A.2 Examples

The following show a few Polar2Grid Version 2.3 commands and the new command structure to produce the same or similar files in Polar2Grid Version 3.0.

Create VIIRS GeoTIFF default output files. The default GeoTIFF now includes an Alpha Band which will make the background transparent along with along with the creation of true and false color images. Version 3 also provides the user with the option to choose how many computer worker threads to use. The default is 4.

v2.3: `polar2grid.sh viirs gtiff -f <path to files>`

v3.0: `polar2grid.sh -r viirs_sdr -w geotiff --num-workers 8 -f <path to files>`

Create VIIRS SDR true and false color GeoTIFF output files. Note the version 3.0 execution will by default create an Alpha band that makes the background transparent. Using `--fill-value 0` will create an image with a black background.

v2.3: `polar2grid.sh crefl gtiff --true-color --false-color -f <path to files>`

v3.0: `polar2grid.sh viirs_sdr geotiff -p true_color false_color --fill-value 0 -f <path to files>`

Create VIIRS SDR I-Band GeoTIFFs with 32 bit floating point output, and customize the product output filenames. The reflectance output in version 3.0 is stored as 0-100% values as opposed to 0.0-1.0 in previous versions.

v2.3: `polar2grid.sh viirs gtiff --grid-coverage 0.002 -p i01 i02 i03 i04 i05 -g polar_300 --dtype real4 --output-pattern {satellite}{instrument}{product_name}{begin_time}{grid_name}.float.tif -f <path to files>`

v3.0 `polar2grid.sh -r viirs_sdr -w geotiff --num-workers 4 --grid-coverage 0.002 -g polar_300 -p i01 i02 i03 i04 i05 --fill-value 0 --dtype float32 --no-enhance --output-filename {satellite}{instrument}{product_name}{begin_time}{grid_name}.float.tif -f <path to files>`

Create true and false color MODIS AWIPS tiles for the United States CONUS Sector.

v2.3: `polar2grid.sh crefl scmi --true-color --false-color --sector-id LCC --letters --compress -g lcc_conus_300 -f <path to files>`

v3.0: `polar2grid.sh -r viirs_sdr -w awips_tiled --awips-true-color --awips-false-color --num-workers 8 -g lcc_conus_300 --sector-id LCC --letters --compress -f <path to files>`

Create MiRS GeoTIFF product files.

v2.3: `polar2grid.sh mirs gtiff -p rain_rate sea_ice snow_cover swe tpw sfr btemp_57h1 btemp_23v btemp_165h btemp_183h1 btemp_88v -g lcc_fit -f <path to files>`

v3.0: `polar2grid.sh -r mirs -w geotiff --num-workers 6 --grid-coverage 0 --fill-value 0 -p rain_rate sea_ice snow_cover swe tpw sfr btemp_57h1 btemp_23v btemp_165h btemp_183h1 btemp_88v -g lcc_fit -f <path to files>`

Create MODIS AWIPS tiles files for the Alaska Region.

v2.3: `polar2grid.sh modis scmi -p vis01 vis02 --sector-id Polar --letters --compress --grid-coverage 0.00001 -g polar_alaska_300 -f <path to files>`

v3.0: `polar2grid.sh -r modis_l1b -w awips_tiled --grid-coverage 0.00001 -p vis01 vis02 -g polar_alaska_300 --sector-id Polar --letters --compress --num-workers 8 -f <path to files>`

Create rescaled AMSR2 GeoTIFF output files.

v2.3: `polar2grid.sh amsr2_l1b gtiff --rescale-configs $POLAR2GRID_HOME/rescale_configs/amsr2_png.ini -g lcc_fit -f <path to file>`

v3.0: `polar2grid.sh -r amsr2_l1b -w geotiff --extra-config-path $POLAR2GRID_HOME/example_enhancements/amsr2_png --fill-value 0 -f <path to file>`

APPENDIX

B

THIRD-PARTY RECIPES

Third-party tools like those provided by the Geospatial Data Abstraction Library (GDAL) can be found in the `libexec/python_runtime/bin` directory alongside the Python executable used by Polar2Grid.

B.1 Combining GeoTIFF Images

When working with polar orbiter satellite data, it is often useful to stitch images of neighboring passes together. The GDAL merge tool can do this easily using Polar2Grid GeoTIFF output files.

Suppose we have two VIIRS GeoTIFF files created from two sequential Suomi NPP overpasses. The GeoTIFF files we use in this example are false color images from data acquired at 20:43 and 22:22 UTC on 23 March 2017 created in a WGS84 projection. The individual images are displayed side by side below.

Fig. 2.1: Suomi-NPP VIIRS False Color Images from two separate passes (Red:VIIRS M-Band 11 (2.25 μm), Green:VIIRS M-Band 7 (.87 μm) and Blue:VIIRS M-Band 5 (.67μm)) observed on 23 March 2017.

To combine these images into a single output GeoTIFF image I can use the *gdal_merge.py* command that is packaged as part of Polar2Grid:

```
gdal_merge.py -n 0 -o my_false_color.tif npp_viirs_false_color_20170323_204320_wgs84_
↪fit.tif npp_viirs_false_color_20170323_222255_wgs84_fit.tif
```

The *-n 0* is used to set the background data value so it will not be included in the merge. This is required because without it, the black regions that border the second WGS84 GeoTIFF will be overlaid on top of the first image.

The resulting image is displayed below.

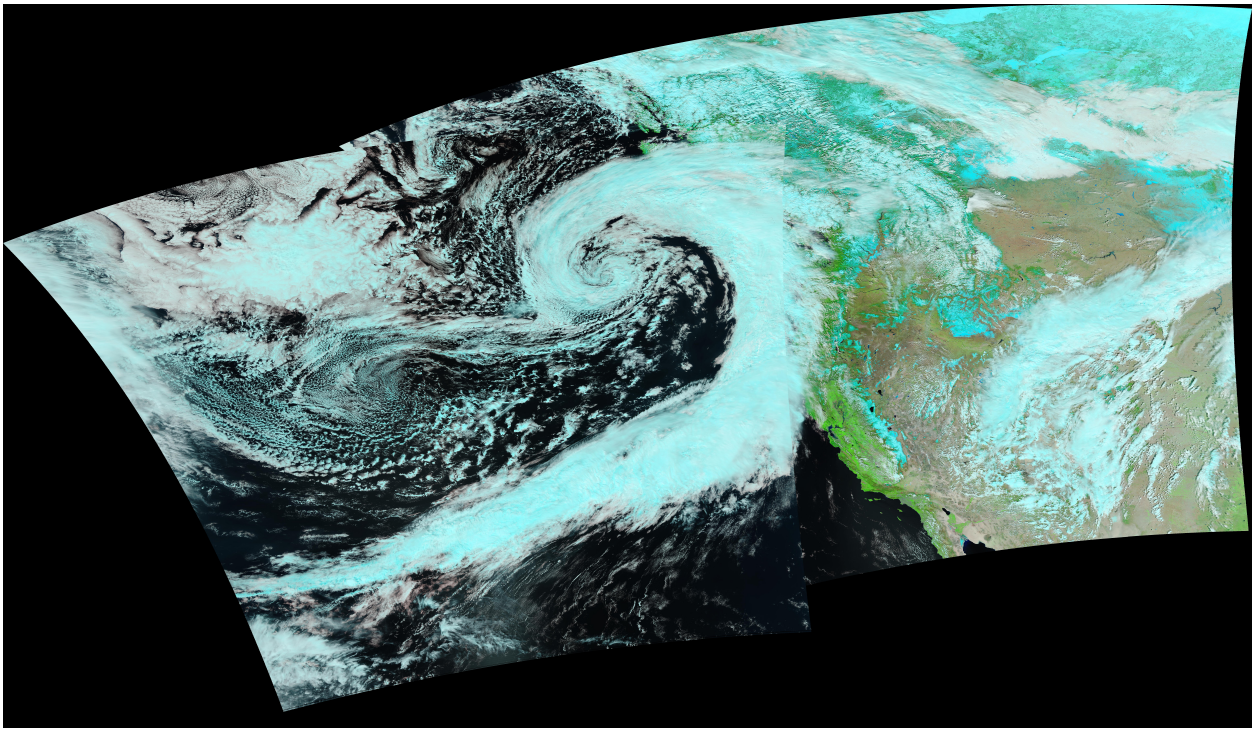


Fig. 2.2: Merged S-NPP VIIRS False Color Images created from a pair of images acquired and processed from two different orbits.

More than one image can be combined. There are more options available to *gdal_merge.py*. Execute

```
gdal_merge.py -h
```

for a complete list of options.

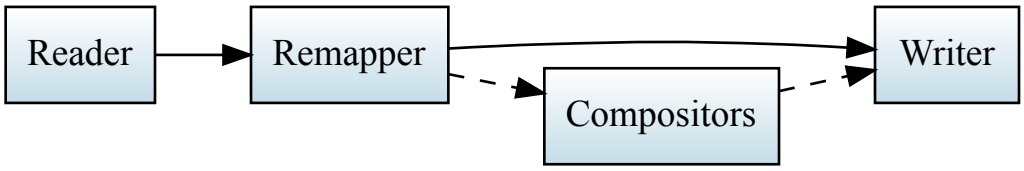
APPENDIX

C

SOFTWARE DESIGN OVERVIEW

The primary goal of Polar2Grid is to allow scientists to convert satellite imager data into a format that they can view using the forecasting tools with which they are most comfortable. Due to the way most satellite instruments operate, raw satellite data comes in many different forms. It often comes in multiple resolutions that can be difficult to combine or compare. Data can also be represented as a non-uniform swath of pixels where each pixel has a corresponding longitude and latitude. This type of sparse data can not be easily shown on viewing programs so it must be resampled to a uniform grid. Resampling is only one of many difficulties involved with processing satellite data and while their solutions can be summarized in a few sentences, there is a lot to consider to get a good looking image suitable for viewing.

Polar2Grid has a modular design to ease development of features added in the future. It operates on the idea of satellite "products"; data observed by a satellite instrument. These products can be any type of raster data, such as temperatures, reflectances, radiances, or any other value that may be recorded by or calculated from an instrument. As shown below there are 4 main steps of Polar2Grid used to work with these products: the Reader, Writer, Compositor, and Remapper. Typically these components are "`glued`" together to create gridded versions of the user provided products. Depending on the input data and what the user wants these steps may be optional or appear in a different order.



In Polar2Grid a majority of this functionality is provided by the open source SatPy library created by the Pytroll group. More information on SatPy and the capabilities it provides to python users can be found in the SatPy documentation. For more on the Pytroll group and their work see the Pytroll home page.

C.1 Data Container

Polar2Grid, and the SatPy library it depends on, use DataArray objects provided by the XArray library. Additionally, these `DataArray` objects use dask arrays underneath. These libraries and their data structures provide community-supported containers for scientific data and easy multithreaded processing.

C.2 Readers

The Reader is responsible for reading provided data files to create Polar2Grid products. In the simplest case, this means reading data from the file and placing it in `DataArray`. In more advanced cases a Reader may choose to provide products that require extra processing; from masking bad values to creating a new product from the combination of others. The *readers documentation* has more details on the current readers available.

C.3 Compositors

Compositors are an optional component of Polar2Grid that may not be needed by most users. The role of a compositor is to create new products that can not be created by the Reader. Usually this means combining multiple products to create a new one. The most common case is creating color (RGB) images like true color or false colors images which are the combination of 3 or more products. Depending on what a composite needs as input, resampling may be needed before the composite can actually be generated.

Customizing the behavior of Compositors is considered an advanced topic and is covered in the SatPy documentation.

C.4 Remapping

Remapping is the process of putting satellite data pixels into an equidistant grid for easier viewing, manipulation, and storage. Polar2Grid currently offers multiple different algorithms for achieving this gridding. See the *remapping documentation* for more information.

C.5 Writers

The Writer's responsibility is to write gridded data to a file format that can be used for viewing and/or analyzing in another program. This usually involves scaling the data to fit the data type used by the file format being written. For example, most satellite temperature data is best represented as floating-point numbers (200.0K - 320.0K), but many file formats like NetCDF or GeoTIFF prefer unsigned 8-bit integers (0 - 255). To best represent the data in the file, the Writer must scale the real-world value to a value that can be written to the output file(s), whether that be with a simple linear transformation or something more complex. For more information, see the *Writers documentation*.

APPENDIX

D

SCALING OF THE VIIRS DAY/NIGHT BAND IN POLAR2GRID

Scaling of the Day/Night Band (DNB) is complicated due to the huge range of values that can exist across a given scene. The Day/Night Band is centered on .7 microns with a wide spectral response function (half width .505 to .890 microns). Polar2Grid offers the user four different options for enhancing the final image product. If no specific DNB enhancement is provided to the viirs readers (for example, polar2grid.sh -r viirs_sdr -w geotiff), three different output products will be created for the given scene by default. The three options are:

- Adaptive Day/Night Band scaling - option `-p adaptive_dnb`
- Dynamic Day/Night Band scaling - option `-p dynamic_dnb`
- Simplified HNCC Day/Night scaling - option `-p hncc_dnb`

In addition, a fourth enhancement option is available by explicitly requesting it in the command line (using the `-p` option).

- Histogram Day/Night Band scaling - option `-p histogram_dnb`

The Histogram and Adaptive enhancements work by breaking up the radiance values and scale them based upon three regimes:

- Day – Solar zenith angles less than 88 degrees,
- Twilight or Terminator Region – Solar Zenith angles between 88 and 100 degrees, and
- Night – Solar Zenith Angles less than 100 degrees.

For each of these regions, a histogram equalization is calculated, excluding data that falls beyond 4 standard deviations of the mean. Then a histogram equalization is calculated across all the data in all of the regions. Then the data are scaled from 0-1, remapped to the requested projection and then finally rescaled to 0-255. This allows us to display day and night data together in one image, and make the maximum use of all of the data no matter how many regimes are included in a swath.

Figure 8.1 below shows a Polar2Grid VIIRS Day/Night band image created using data that includes the transition region between day and night regimes (left panel). This data set was acquired on 22 June 2015.

The Adaptive Scaling Option (center panel) is an alternative that attempts to provide better contrast across the Terminator region of the Day/Night band. This algorithm cuts each region into tiles and calculates a histogram equalization for each tile. Once the histogram equalization functions have been calculated for each tile, each tile is processed separately. The "current tile" is equalized using the histogram equalization calculated from itself and it is also separately equalized using the surrounding tiles. These resulting equalized versions of the tiles are combined using bi-linear interpolation, so that each pixel uses a weighted amount in inverse relation to it's distance from the centers of the nearest 4 tiles. An example of the result of applying this technique to the same data set can be seen in the following image. Please note that some image artifacts (wave patterns) are introduced when applying this technique over the Terminator region.

The Dynamic option (right panel) implements an error function to scale the VIIRS Day/Night band data. This algorithm was provided by Dr. Curtis Seaman, NOAA Cooperative Institute for Research in the Atmosphere (CIRA), Colorado State University. For detailed information on this technique, please see:

> Curtis J. Seaman, Steven D. Miller, 2015: A dynamic scaling algorithm for the optimized digital display of VIIRS Day/Night Band imagery. International Journal of Remote Sensing, Vol. 36, Iss. 7, pp. 1839-1854. DOI: 10.1080/01431161.2015.1029100.

And finally, Figure 8.2 provides an example of a technique that utilizes a simplified high and near-constant contrast approach. This approach was created by Stephan Zinke of the European Organisation for the Exploitation of Meteorological Satellites (EUMETSAT). His technique supports the display of VIIRS Day/Night Band granules using consistent settings for all granules. In this way, it provides consistent results whether applied to single granules whose images are then stitched together or if it is applied to a concatenation of granules.

For more information about this technique, and for more details about the example dataset shown in Figure 8.2 please see :

> Zinke, Stephan, 2017: A simplified high and near-constant contrast approach for the display of VIIRS day/night band imagery. International Journal of Remote Sensing, Vol. 38 Iss. 19, pp.5374-5387. DOI: 10.1080/01431161.2017.1338838

Fig. 4.1: Example of three options for scaling the VIIRS Day/Night band in Polar2Grid for a S-NPP pass collected on 22 June 2015. The left panel applies a histogram equalization technique (histogram_dnb), center panel utilizes an adaptive histogram equalization technique (adaptive_dnb), and the third option (right panel) implements an dynamic error function scaling technique (dynamic_dnb).

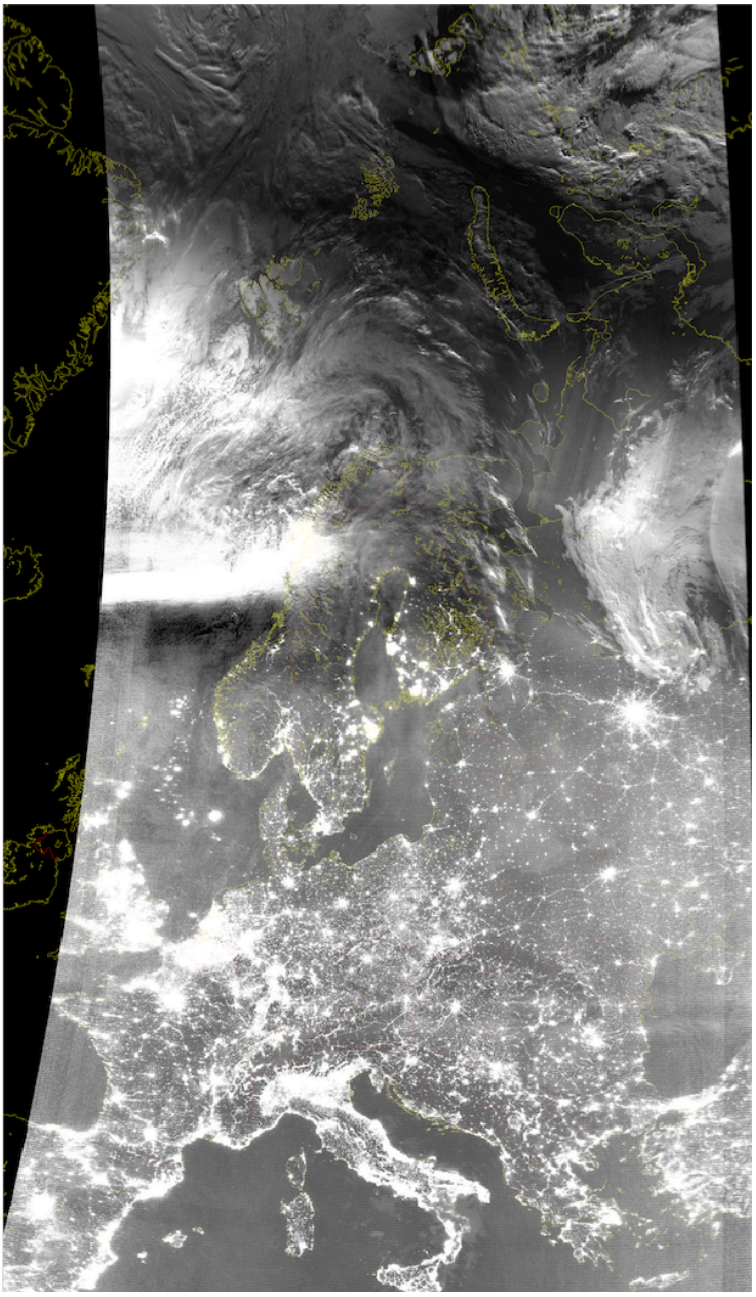


Fig. 4.2: Example of the high and near-constant contrast VIIRS Day/Night band scaling option (-p hncc_dnb) image created from a S-NPP pass collected on 1 September 2016. For more information about the lunar illumination regimes of this data, please see Zinke, 2017.