Geo2Grid Documentation

Release 1.1

NOAA Community Satellite Processing Package for Geostationary Satellites (CSPP Geo)

Dec 08, 2022

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Overview

Geo2Grid 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. Geo2Grid was created by scientists and software developers at the SSEC. It is distributed as part of the CSPP Geo project for processing of data received via direct broadcast antennas. Although Geo2Grid was created to serve the direct broadcast community, it can be used on most archived data files.

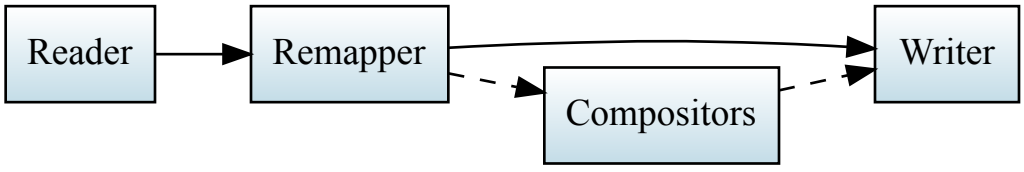
The features provided by Geo2Grid are accessible via bash scripts and binary command line tools. This is meant to give scientists an easy way to use and access features that typically involve complicated programming interfaces. Linux terminal commands included in these instructions assume the bash shell is used.

Documentation Website

GitHub Repository

CSPP Geo2Grid Forum

1.2 Software Design



Geo2Grid 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 Geo2Grid: reading, writing, compositing, and remapping. Geo2Grid makes it possible to access and configure these steps with a simple bash script called `geo2grid.sh` and other helper scripts. More information on accessing Geo2Grid's features and running its scripts can be found in the *Geo2Grid 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 Geo2Grid 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?

Included in this release:

- GOES-18 ABI reader support added
- ABI Level 2 (abi_l2_nc) reader added
- Gridded GLM (glm_l2) reader added
- GEO-KOMPSAT AMI (ami_l1b) reader added
- FY-4A AGRI (agri_fy4a_l1) reader added
- FY-4B AGRI (agri_fy4b_l1) reader added
- Various optimizations
- Support for additional RGBs
- Use of yaml files for grid definitions
- Various bug fixes

For more details on what's new in this version and past versions see the Geo2Grid Release Notes in the github repository.

1.4 System Requirements

For minimal processing requirements (i.e. not realtime) the following system specifications are required:

- Intel or AMD CPU with 64-bit instruction support (2+ cores - 2.4GHz)
- 16 GB RAM (minimum)
- CentOS 7.9 64-bit Linux (or other compatible 64-bit Linux distribution)
- 10 GB disk space (minimum)

For a more demanding processing load, like realtime generation of all GOES-16 ABI channels, true color, and natural color RGB images at full resolution, a system should have at least:

- Intel or AMD CPU with 64-bit instruction support (20+ cores - 2.4GHz)
- 64 GB RAM (minimum)
- CentOS 7.9 64-bit Linux (or other compatible 64-bit Linux distribution)
- 1 TB disk space (minimum for ~1 week of images, does not include long-term storage)

Local storage (i.e. not network file systems) are preferred to limit any effect that network congestion may have. If additional satellites are included in the processing requirements then the above system requirements will need to be adjusted accordingly.

1.4.1 Execution Times

The following table provides execution time averages for creating all default GeoTIFF images at full spatial resolution for the given instrument and sector. Eight computer threads were used. The times are provided for the higher end system defined above. Execution times decrease when fewer bands and smaller regions are processed.

Table of Execution Times for Creating GeoTIFF default Images (All bands plus true and natural color images)

Instrument	Full Disk Sector	CONUS Sector	1000x1000 pixel subset
GOES ABI	2m50s	29s	14s
AHI HSD	5m40s	N/A	23s
AHI HimawariCast	23s	N/A	11s
GEO-KOMPSAT AMI	2m48s	N/A	12s
FY4 AGRI	5m45s	N/A	29s

1.5 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 Geo 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 Geo software.

CHAPTER

TWO

INSTALLATION

Geo2Grid is released as an all-in-one tarball for Enterprise Linux systems. The tarball, or software bundle, provided by the CSPP for Geostationary Satellites (CSPP Geo) team includes a python runtime and all of the necessary third-party software to run the features provided by Geo2Grid. 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 Geo2Grid 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 Geo2Grid Software

The Geo2Grid tarball can be downloaded from the CSPP Geo website: http://cimss.ssec.wisc.edu/csppgeo/

To install the software, unpack the tarball:

```
tar xf CSPP_GEO2GRID_V1.1.tar.gz
```

This will create a Geo2Grid software bundle directory, `geo2grid_v_1_1`. To simplify calling scripts included in the bundle the following line should be added to your `.bash_profile`:

```
export GEO2GRID_HOME=/path/to/geo2grid_v_1_1
```

All other environment information needed to run is automatically loaded by the scripts provided by Geo2Grid. Scripts are typically invoked using:

```
$GEO2GRID_HOME/bin/<g2g_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=$PATH:$GEO2GRID_HOME/bin
```

The example invocations in this document assume you have added this to your PATH.

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 `geo2grid.sh`).

See *Geo2Grid Basics* for more information on running Geo2Grid.

2.2 Geo2Grid Test Data

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

```
CSPP_GEO2GRID_V1.1_TEST_DATA.tar.gz
```

The test data should be unpacked in a directory separate from the Geo2Grid installation:

```
cd $HOME
tar xf CSPP_GEO2GRID_V1.1_TEST_DATA.tar.gz
```

This will create a `geo2grid_test` directory containing the test input, output, and verification scripts for the ABI instrument.

See *Verifying your Geo2Grid Installation* for instructions on using the verification test data.

CHAPTER

THREE

GEO2GRID BASICS

All of the tools provided by Geo2Grid 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 purpose of Geo2Grid is to convert satellite data files into high quality gridded image files. The main run script is `geo2grid.sh` and requires users to choose an input reader (-r what instrument data would you like to use) and an output writer (-w what output format would you like to create). The only other required input is the list of files or a directory pointing to the location of the input files (-f). Each instrument data reader by default will create single band output image GeoTIFF files for whatever bands are provided, along with true and natural color images. Only one time step can be processed with each script execution.

For example, executing the following command will create 8-bit GeoTIFF files of all 16 ABI imager channels, a true color RGB, and natural color RGB in the native resolution of the instrument channel (500m for RGB composites). This can be customized with command line arguments.

```
$GEO2GRID_HOME/bin/geo2grid.sh -r abi_l1b -w geotiff -f <path to files>
```

This script takes advantage of the modular design of Geo2Grid; a user only needs to decide on a *Reader* and a *Writer* and provide them to `geo2grid.sh`.

If processing errors occur Geo2Grid will attempt to continue processing to make as many products as it can.

3.2 Common Script Options

Additional command line arguments for the `geo2grid.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 Geo2Grid 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 (Currently only option is geotiff).
-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 geo2grid 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 native instrument projection. See *Grids* and *Custom Grids* for information on other possible values.

--cache-dir <dir> Directory to store resampling intermediate results between executions. Not used with 'native' resampling method.

--num-workers NUM_WORKERS Specify number of worker threads to use (Default: 4).

--progress Show processing progress bar (Not recommended for logged output).

--ll-bbox <lonmin latmin lonmax latmax> Subset input data to the bounding coordinates specified.

-v Print detailed log information.

Examples:

```
geo2grid.sh -r abi_l1b -w geotiff --list-products -f <path to files>/<list of files>

geo2grid.sh -r abi_l1b -w geotiff -p C01 natural_color -v -f <path to files>

geo2grid.sh -r abi_l1b -w geotiff --ll-bbox -95.0 40.0 -85.0 50.0 -f /abi/OR_ABI-L1b-
↪RadF-*.nc

geo2grid.sh -r ahi_hsd -w geotiff -p B03 B04 B05 B14 -f /ahi/*FLDK*.DAT

geo2grid.sh -r ahi_hrit -w geotiff -f /ahi/IMG_DK01*

geo2grid.sh -r ami_l1b -w geotiff -p IR112 VI006 --num-workers 12 -f /ami/gk2a_ami_
↪l31b*.nc

geo2grid.sh -r agri_fy4a_l1 -w geotiff -p C07 natural_color --progress -f /fy4a/FY4A-_
↪AGRI--*.HDF

geo2grid.sh -r abi_l2_nc -w geotiff -f /cloud_products/CG_ABI-L2-ACH?C-M6_G16*.nc

geo2grid.sh -r glm_l2 -w geotiff -p flash_extent_density -f CG_GLM-L2-GLMF-M3_G18*.nc
```

For information on other scripts and features provided by Geo2Grid 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 `geo2grid.sh`. To access these features provide the "reader" and "writer" names to the `geo2grid.sh` script followed by other script options:

```
$GEO2GRID_HOME/bin/geo2grid.sh -r <reader> -w <writer> --list-products <options> -f /
↪path/to/files
```

Table 3.1: Geo2Grid Reader Summary Table

Reader Name	Input Source	Input Filename Pattern
abi_l1b	ABI L1B	`OR_ABI-L1b-Rad?-M?C??_G??_s<YYYYDDDHHMMSSS_eYYYYDDDHHMMSSS_c*.nc` **Ex: OR_ABI-L1b-RadF-M3C03_G16_s20183161830345_e20183161841112_c20183161841154.nc**
ahi_hsd	AHI HSD	`HS_H08_YYYYMMDD_HHMM_B??_*_*_S*.DAT` **Ex: HS_H08_20181112_1230_B08_FLDK_R20_S0910.DAT**
ahi_hrit	AHI HRIT (HimawariCast) from JMA	`IMG_DK01B??_YYYYMMDDHHMM` **Ex: IMG_DK01B05_201811121230**
ami_l1b	GEO-KOMPSAT AMI from KMA	`gk2a_ami_le1b_?????_fd0??ge_YYYYMMDDHHMM.nc` **Ex: gk2a_ami_le1b_vi006_fd005ge_201909300300.nc**

agri_fy4a_l1
agri_fy4b_l1
	FY4A, FY4B AGRI from CMA	`FY4A-_AGRI--_N_DISK_1047E_L1-_???-MULT_NOM_YYYYMMDDHHMMSS_*_????M_V0001.HDF`
		`FY4B-_AGRI--_N_DISK_1330E_L1-_???-MULT_NOM_YYYYMMDDHHMMSS_*_????M_V0001.HDF`
		Ex: FY4A-_AGRI–_N_DISK_1047E_L1-_FDI-_MULT_NOM_20220120040000_*_1000M_V0001.HDF
		Ex: FY4B-_AGRI–_N_DISK_1330E_L1-_FDI-_MULT_NOM_20221115030000_*_0500M_V0001.HDF

abi_l2_nc	ABI Product Cloud Height and Cloud Top Temperature	`CG_ABI-L2-{PROD}F-M6_G??_sYYYYDDDHHMMSSS_*.nc` **Ex: CG_ABI-L2-ACHAF-M6_G17_s20223271830316_e20223271839394_c20223271842100.nc** **Ex: CG_ABI-L2-ACHTF-M6_G17_s20223271830316_e20223271839394_c20223271842100.nc**
glm_l2	CSPP Gridded GLM Product	`CG_GLM-L2-GLM?-M3_G??_sYYYYDDDHHMMSSS_*.nc` **Ex: CG_GLM-L2-GLMF-M3_G18_s20223141900000_e20223141901000_c20223142002160.nc**

Table 3.2: Geo2Grid Writer Summary Table

Writer Name	Output Filename Pattern	Output Format
geotiff	`GOES-??_ABI_Rad?_C??_YYYYMMDD_HHMMSS_*.tif` **Ex: GOES-16_ABI_RadF_C03_20181112_183034_GOES-East.tif**	8-bit single band GeoTIFF
geotiff	`HIMAWARI-8_AHI_B??_YYYYMMDD_HHMMSS_*.tif` **Ex: HIMAWARI-8_AHI_B03_20181112_183020_FLDK.tif**	8-bit single band GeoTIFF
geotiff	`GEO-KOMPSAT-2A_AMI_?????_YYYYMMDD_HHMMSS_ami_geos_fd.tif` **Ex: GEO-KOMPSAT-2A_AMI_VI006_20190930_030031_ami_geos_fd.tif**	8-bit single band GeoTIFF
geotiff	`FY-4A_AGRI_C??_YYYYMMDD_HHMMSS_DISK_????m.tif` **Ex: FY-4A_AGRI_C07_20220120_040004_DISK_2000m.tif** **Ex: FY-4B_AGRI_C11_20221115_030002_DISK_4000m.tif**	8-bit single band GeoTIFF
geotiff	`GOES-17_ABI_????_YYYYMMDD_HHMMSS_GOES-West.tif` **Ex: GOES-17_ABI_TEMP_20221123_183031_GOES-West.tif** **Ex: GOES-17_ABI_HT_20221123_183031_GOES-West.tif**	8-bit single band color enhanced GeoTIFF
geotiff	`GOES-18_GLM_{PRODUCT}_YYYYMMDD_HHMMSS_GOES-West.tif` **Ex: GOES-18_GLM_group_extent_density_20221110_190000_GOES-West.tif**	8-bit single band GeoTIFF

3.3.1 Creating Red Green Blue (RGB) Composite Imagery

The list of supported products includes true and natural color 24-bit RGB imagery. The software uses the number of specified CPU threads to create high quality reprojections in the lowest latency possible thanks to the dask python library. Dask splits data arrays in to multiple "chunks" and processes them in parallel. The creation of these RGBs includes the following steps, which are performed by default with each execution:

- Check for required spectral bands used in RGB creation among input files.
- Upsample and sharpen composite bands to the highest spatial resolution (500m).
- Creation of pseudo "green" band for the ABI and AGRI instruments.
- Reflectance adjustment (dividing by cosine of the solar zenith angle).
- Removal of atmospheric Rayleigh scattering (atmospheric correction).
- Nonlinear scaling before writing data to disk.

Geo2Grid also supports the creation of other RGBs (this varies depending on the instrument), however these files are not produced by default. The recipes for creating these RGBs come from historical EUMETSAT recipes that have been adjusted to work with the data being used in Geo2Grid.

3.4 Creating Your Own Custom Grids

The Geo2Grid software bundle comes with a script for *Custom Grid Utility* that allows users to easily create Geo2Grid custom grid definitions over a user determined longitude and latitude region. Once these definitions have been created, they can be provided to Geo2Grid. To run the utility script from the software bundle wrapper run:

```
$GEO2GRID_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 Geo2Grid 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 Geo2Grid 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 ABI L1B Reader

The ABI Level 1B Reader operates on NOAA Level 1B (L1B) NetCDF files from the GOES-16 (GOES-East) and GOES-17/18 (GOES-West) Advanced Baseline Imager (ABI) instrument. The ABI L1B reader works off of the input filenames to determine if a file is supported by Geo2Grid. Files usually have the following naming scheme:

```
OR_ABI-L1b-RadF-M6C02_G18_s20223191830206_e20223191839514_c20223191839547.nc
```

These are the mission compliant radiance file naming conventions used by the NOAA Comprehensive Large Array-data Stewardship System (CLASS) archive and the CSPP GOES Rebroadcast (GRB) software. The ABI L1B reader supports all instrument spectral bands, identified in Geo2Grid as the products shown in the table below. The ABI L1B reader can be provided to the main geo2grid.sh script using the `-r` option and the reader name `abi_l1b`.

The list of supported products includes true and natural color imagery. These are created by means of a python based atmospheric Rayleigh scattering correction algorithm that is executed as part of the Geo2Grid ABI L1B reader, along with sharpening to the highest spatial resolution. For more information on the creation of RGBs, please see the *RGB section*.

Product Name	Description
C01	Channel 1 (0.47um) Reflectance Band
C02	Channel 2 (0.64um) Reflectance Band
C03	Channel 3 (0.86um) Reflectance Band
C04	Channel 4 (1.37um) Reflectance Band
C05	Channel 5 (1.6um) Reflectance Band
C06	Channel 6 (2.2um) Reflectance Band
C07	Channel 7 (3.9um) Brightness Temperature Band
C08	Channel 8 (6.2um) Brightness Temperature Band
C09	Channel 9 (6.9um) Brightness Temperature Band
C10	Channel 10 (7.3um) Brightness Temperature Band
C11	Channel 11 (8.4um) Brightness Temperature Band
C12	Channel 12 (9.6um) Brightness Temperature Band
C13	Channel 13 (10.3um) Brightness Temperature Band
C14	Channel 14 (11.2um) Brightness Temperature Band
C15	Channel 15 (12.3um) Brightness Temperature Band
C16	Channel 16 (13.3um) Brightness Temperature Band
true_color	Ratio sharpened rayleigh corrected true color
natural_color	Ratio sharpened corrected natural color
airmass	Air mass RGB
ash	Ash RGB
dust	Dust RGB
fog	Fog RGB
night_microphysics	Night Microphysics RGB

4.1.1 Command Line Usage

```
usage: geo2grid.sh -r abi_l1b -w <writer> [-h]
```

Examples:

```
geo2grid.sh -r abi_l1b -h

geo2grid.sh -r abi_l1b -w geotiff --list-products -f /data/goes16/abi

geo2grid.sh -r abi_l1b -w geotiff --num-workers 8 -f /data/goes18/abi

geo2grid.sh -r abi_l1b -w geotiff -p C02 C03 C04 C05 C06 true_color -f OR_ABI-L1b-
↪RadC*.nc

geo2grid.sh -r abi_l1b -w geotiff --ll-bbox -95.0 40.0 -85.0 50.0 -f OR_ABI-L1b-RadC*.
↪nc

geo2grid.sh -r abi_l1b -w geotiff -p airmass dust --num-workers 4 --grid-configs=/
↪home/g2g/my_grid.conf -g madison --method nearest -f /data/goes17/abi/
```

4.1.2 Product Explanation

True Color

The ABI L1b "true_color" composite produced by Geo2Grid 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 image is an *RGB (Red, Green, Blue) image* consisting of the ABI C02 reflectance channel, a pseudo-green reflectance channel, and the ABI C01 reflectance channel. Each channel goes through a series of modifications to produce the final high quality image output by Geo2Grid.

The pseudo-green channel is a simple combination of 46.5% of C01, 46.5% of C02, and 7% of C03. While it is impossible to completely reproduce what a dedicated "green" channel on the ABI instrument would see, this method provides a good approximation while also limiting the computational requirements to produce it.

All bands involved in the true color composite have the *Solar Zenith Angle Modification* and *Rayleigh Scattering Correction* applied, except for C03 in the pseudo-green band where rayleigh correction is not applied due to the minimal effect it would have at that wavelength.

To improve the general spatial quality of the image, a *Self Ratio Sharpening* is also applied. Lastly, a *Non-linear enhancement* is applied.

C01 through C06

The Channel 1 through Channel 6 products are all reflectance channels on the ABI 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 brightening dark regions of the image.

For a more detailed explanation of these bands on the instrument, see the ABI Bands Quick Information Guides.

C07 and C11 through C16

These channels 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.

For a more detailed explanation of these bands on the instrument, see the ABI Bands Quick Information Guides.

C08 through C10

These channels are also brightness temperature (infrared/IR) channels, but are scaled differently than those above. These are scaled linearly between 180K and 280K and then inverted. The inversion has the same effect of making land dark and clouds white.

For a more detailed explanation of these bands on the instrument, see the ABI Bands Quick Information Guides.

4.2 ABI L2 NetCDF Reader

The ABI Level 2 Reader operates on NOAA Level 2 (L2) NetCDF files from the GOES-16 (GOES-East) and GOES-17 (GOES-West) Advanced Baseline Imager (ABI) instrument. The ABI L2 NetCDF reader works off of the input filenames to determine if a file is supported by Geo2Grid. Files usually have the following naming scheme:

```
OR_ABI-L2-{PROD}F-M3_G16_s20182531700311_e20182531711090_c20182531711149.nc
```

and:

```
CG_ABI-L2-{PROD}F-M6_G17_s20223271830316_e20223271839394_c20223271842100.nc
```

These are the mission compliant radiance file naming conventions used by the NOAA Comprehensive Large Array-data Stewardship System (CLASS) archive and the CSPP Geo AIT Framework Level 2 software. The ABI L2 NetCDF reader supports most L2 products, but Geo2Grid is tested with a limited subset of these. See the table below for more information. The ABI L2 NetCDF reader can be provided to the main geo2grid.sh script using the `-r` option and the reader name `abi_l2_nc`.

Product Name	Description
HT	Cloud Top Height
TEMP	Cloud Top Temperature

4.2.1 Command Line Usage

```
usage: geo2grid.sh -r abi_l2_nc -w <writer> [-h]
```

Examples:

```
geo2grid.sh -r abi_l2_nc -h

geo2grid.sh -r abi_l2_nc -w geotiff --list-products -f abi/full_disk/CG_ABI-L2-ACHAF-
↪M6_G17_*.nc

geo2grid.sh -r abi_l2_nc -w geotiff -p TEMP -f CG_ABI-L2-ACHTF-M6_G17_s20223271830316_
↪e20223271839394_c20223271842100.nc

geo2grid.sh -r abi_l2_nc -w geotiff -f /data/conus/CG_ABI-L2-ACH?C-M6_G16*.nc

geo2grid.sh -r abi_l2_nc -w geotiff -f /abi/meso1/CG_ABI-L2-ACHAM1-M6_G17_
↪s20223271830588_e20223271831056_c20223271831440.nc
```

4.3 AGRI FY-4A L1 Reader

The AGRI FY-4A L1 Reader operates on Level 1 (L1) HDF5 files for the Advanced Geostationary Radiation Imager (AGRI) instrument on board the Feng-Yun - 4 (FY-4A) satellite. Geo2Grid determines if it supports files based on the input filename. Files for AGRI supported by Geo2Grid usually have the following naming scheme:

```
FY4A-_AGRI--_N_DISK_1047E_L1-_FDI-_MULT_NOM_20220117000000_20220117001459_0500M_V0001.
↪HDF
```

The AGRI L1 reader supports all instrument spectral bands, identified in Geo2Grid as the products shown in the table below. The AGRI L1 reader for FY-4A can be provided to the main geo2grid.sh script using the `-r` option and the reader name `agri_fy4a_l1`.

The list of supported products includes true and natural color imagery. These are created by means of a python based atmospheric Rayleigh scattering correction algorithm that is executed as part of the Geo2Grid AGRI L1 reader, along with sharpening to the highest spatial resolution. For more information on the creation of RGBs, please see the *RGB section*.

Product Name	Description
C01	Channel 1 (0.47um) Reflectance Band
C02	Channel 2 (0.65um) Reflectance Band
C03	Channel 3 (0.83um) Reflectance Band
C04	Channel 4 (1.37um) Reflectance Band
C05	Channel 5 (1.61um) Reflectance Band
C06	Channel 6 (2.25um) Reflectance Band
C07	Channel 7 (3.75um) Brightness Temperature Band
C08	Channel 8 (3.75um) Brightness Temperature Band
C09	Channel 9 (6.25um) Brightness Temperature Band
C10	Channel 10 (7.10um) Brightness Temperature Band
C11	Channel 11 (8.5um) Brightness Temperature Band
C12	Channel 12 (10.7um) Brightness Temperature Band
C13	Channel 13 (12.0um) Brightness Temperature Band
C14	Channel 14 (13.5um) Brightness Temperature Band
true_color	Ratio sharpened rayleigh corrected true color
natural_color	Ratio sharpened corrected natural color
ash	Ash RGB
dust	Dust RGB
fog	Fog RGB

4.3.1 Command Line Usage

```
usage: geo2grid.sh -r agri_fy4a_l1 -w <writer> [-h]
```

Examples:

```
geo2grid.sh -r agri_fy4a_l1 -h

geo2grid.sh -r agri_fy4a_l1 -w geotiff --list-products -f /data/fy4a/agri

geo2grid.sh -r agri_fy4a_l1 -w geotiff --num-workers 8 -f /data/fy4a/agri

geo2grid.sh -r agri_fy4a_l1 -w geotiff -p C02 C03 C08 C10 C12 -f FY4A-_AGRI--*.HDF

geo2grid.sh -r agri_fy4a_l1 -w geotiff --ll-bbox 100 20 120 40 -p true_color -f /data/
↪*.HDF
```

4.4 AGRI FY-4B L1 Reader

The AGRI FY-4B L1 Reader operates on Level 1 (L1) HDF5 files for the Advanced Geostationary Radiation Imager (AGRI) instrument on board the Feng-Yun - 4B (FY-4B) satellite. Geo2Grid determines if it supports files based on the input filename. Files for AGRI supported by Geo2Grid usually have the following naming scheme:

```
FY4B-_AGRI--_N_DISK_1047E_L1-_FDI-_MULT_NOM_20220117000000_20220117001459_0500M_V0001.
↪HDF
```

The AGRI L1 reader supports all instrument spectral bands, identified in Geo2Grid as the products shown in the table below. The AGRI L1 reader for FY-4B can be provided to the main geo2grid.sh script using the `-r` option and the reader name `agri_fy4b_l1`.

The list of supported products includes true and natural color imagery. These are created by means of a python based atmospheric Rayleigh scattering correction algorithm that is executed as part of the Geo2Grid AGRI L1 reader, along with sharpening to the highest spatial resolution. For more information on the creation of RGBs, please see the *RGB section*.

Product Name	Description
C01	Channel 1 (0.47um) Reflectance Band
C02	Channel 2 (0.65um) Reflectance Band
C03	Channel 3 (0.83um) Reflectance Band
C04	Channel 4 (1.37um) Reflectance Band
C05	Channel 5 (1.61um) Reflectance Band
C06	Channel 6 (2.25um) Reflectance Band
C07	Channel 7 (3.75um) Brightness Temperature Band
C08	Channel 8 (3.75um) Brightness Temperature Band
C09	Channel 9 (6.25um) Brightness Temperature Band
C10	Channel 10 (6.95um) Brightness Temperature Band
C11	Channel 11 (7.42um) Brightness Temperature Band
C12	Channel 12 (8.5um) Brightness Temperature Band
C13	Channel 13 (10.8um) Brightness Temperature Band
C14	Channel 14 (12.0um) Brightness Temperature Band
C15	Channel 15 (13.5um) Brightness Temperature Band
true_color	Ratio sharpened rayleigh corrected true color
natural_color	Ratio sharpened corrected natural color
ash	Ash RGB
dust	Dust RGB
fog	Fog RGB

4.4.1 Command Line Usage

```
usage: geo2grid.sh -r agri_fy4b_l1 -w <writer> [-h]
```

Examples:

```
geo2grid.sh -r agri_fy4b_l1 -h

geo2grid.sh -r agri_fy4b_l1 -w geotiff --list-products -f /data/fy4b/agri

geo2grid.sh -r agri_fy4b_l1 -w geotiff --num-workers 8 -f /data/fy4b/agri
```

(continues on next page)

(continued from previous page)

```
geo2grid.sh -r agri_fy4b_l1 -w geotiff -p C02 C03 C08 C10 C12 -f FY4B-_AGRI--*.HDF

geo2grid.sh -r agri_fy4b_l1 -w geotiff --ll-bbox 100 20 120 40 -p true_color -f /data/
↪*.HDF
```

4.5 AHI Himawari Standard Data (HSD) Reader

The AHI HSD Reader operates on standard files from the Japan Meteorological Agency (JMA) Himawari-8 and Himawari-9 Advanced Himawari Imager (AHI) instrument. The AHI Himawari Standard Data (HSD) reader works off of the input filenames to determine if a file is supported by Geo2Grid. Files usually have the following naming scheme:

HS_H08_20181022_0300_B09_FLDK_R20_S1010.DAT

These are the mission compliant radiance file naming conventions used by JMA. The AHI HSD reader supports all instrument spectral bands, identified in Geo2Grid as the products shown in the table below. The AHI HSD reader can be provided to the main geo2grid.sh script using the `-r` option and the reader name `ahi_hsd`.

The list of supported products includes true and natural color imagery. These are created by means of a python based atmospheric Rayleigh scattering correction algorithm that is executed as part of the Geo2Grid AHI HSD reader, along with sharpening to the highest spatial resolution. For more information on the creation of RGBs, please see the *RGB section*.

Product Name	Description
B01	Channel 1 (0.47um) Reflectance Band
B02	Channel 2 (0.51um) Reflectance Band
B03	Channel 3 (0.64um) Reflectance Band
B04	Channel 4 (0.86um) Reflectance Band
B05	Channel 5 (1.6um) Reflectance Band
B06	Channel 6 (2.3um) Reflectance Band
B07	Channel 7 (3.9um) Brightness Temperature Band
B08	Channel 8 (6.2um) Brightness Temperature Band
B09	Channel 9 (6.9um) Brightness Temperature Band
B10	Channel 10 (7.3um) Brightness Temperature Band
B11	Channel 11 (8.6um) Brightness Temperature Band
B12	Channel 12 (9.6um) Brightness Temperature Band
B13	Channel 13 (10.4um) Brightness Temperature Band
B14	Channel 14 (11.2um) Brightness Temperature Band
B15	Channel 15 (12.4um) Brightness Temperature Band
B16	Channel 16 (13.3um) Brightness Temperature Band
true_color	Ratio sharpened rayleigh corrected true color
natural_color	Ratio sharpened rayleigh corrected natural color
airmass	Air mass RGB
ash	Ash RGB
dust	Dust RGB
fog	Fog RGB
night_microphysics	Night Microphysics RGB

4.5.1 Command Line Usage

```
usage: geo2grid.sh -r ahi_hsd -w <writer> [-h] [--mask-space]
```

AHI HSD Reader

--mask-space, --no-mask-space Mask space pixels. (default: True)

Default: True

Examples:

```
geo2grid.sh -r ahi_hsd -h

geo2grid.sh -r ahi_hsd -w geotiff --list-products -f /ahi8/data/

geo2grid.sh -r ahi_hsd -w geotiff -f HS_H08_20181112_2330_B01_R301_R10_S0101.DAT

geo2grid.sh -r ahi_hsd -w geotiff -p B02 B03 B04 B05 B06 true_color --num-workers 8 -
↪f /data/hsd/2330/*FLDK*.DAT

geo2grid.sh -r ahi_hsd -w geotiff -p B09 B10 B11 B12 B13 natural_color -f /ahi8/HS_
↪H08_20181112_2330_*R303*.DAT

geo2grid.sh -r ahi_hsd -w geotiff --ll-bbox 125 -15 160 10 -f /data/hsd/*FLDK*.DAT

geo2grid.sh -r ahi_hsd -w geotiff -p true_color natural_color --num-workers 4 \
    --grid-configs=/geo/my_grid.conf -g perth --method nearest -f /data/hsd/*FLDK*.DAT
```

4.6 AHI HimawariCast (HRIT) Reader

The AHI High Rate Information Transmission (HRIT) Reader operates on standard Japan Meteorological Agency (JMA) Himawari-8 and Himawari-9 Advanced Himawari Imager (AHI) HRIT Digital Video Broadcast (DVB) HimawariCast files. This broadcast consists of a subset of 14 bands at reduced spatial resolution. The AHI HRIT reader works off of the input filenames to determine if a file is supported by Geo2Grid. Files usually have the following naming scheme:

IMG_DK01B04_201809100300

These are the mission compliant radiance file naming conventions used by JMA. The AHI HRIT reader supports a subset of instrument spectral bands, identified in Geo2Grid as the products shown in the table below. The AHI HRIT reader can be provided to the main geo2grid.sh script using the `-r` option and the reader name `ahi_hrit`.

The list of supported products includes natural color imagery. This is created by means of a python based atmospheric Rayleigh scattering correction algorithm that is executed as part of the Geo2Grid AHI HRIT reader, along with sharpening to the highest spatial resolution.

Please note that true color image RGB creation is not supported for HimawariCast because AHI Band 1 (Blue) and Band 2 (Green) are not included.

For more information on the creation of RGBs, please see the *RGB section*.

Product Name	Description
B03	Channel 3 (0.64um) Reflectance Band
B04	Channel 4 (0.86um) Reflectance Band
B05	Channel 5 (1.6um) Reflectance Band
B06	Channel 6 (2.3um) Reflectance Band
B07	Channel 7 (3.9um) Brightness Temperature Band
B08	Channel 8 (6.2um) Brightness Temperature Band
B09	Channel 9 (6.9um) Brightness Temperature Band
B10	Channel 10 (7.3um) Brightness Temperature Band
B11	Channel 11 (8.6um) Brightness Temperature Band
B12	Channel 12 (9.6um) Brightness Temperature Band
B13	Channel 13 (10.4um) Brightness Temperature Band
B14	Channel 14 (11.2um) Brightness Temperature Band
B15	Channel 15 (12.4um) Brightness Temperature Band
B16	Channel 16 (13.3um) Brightness Temperature Band
natural_color	Ratio sharpened rayleigh corrected natural color
airmass	Air mass RGB
ash	Ash RGB
dust	Dust RGB
fog	Fog RGB
night_microphysics	Night Microphysics RGB

4.6.1 Command Line Usage

```
usage: geo2grid.sh -r ahi_hrit -w <writer> [-h]
```

Examples:

```
geo2grid.sh -r ahi_hrit -h

geo2grid.sh -r ahi_hrit -w geotiff --list-products -f /ahi8/hcast/2330

geo2grid.sh -r ahi_hrit -w geotiff -f /ahi8/hcast/IMG_DK01IR4_201811122330

geo2grid.sh -r ahi_hrit -w geotiff -p B03 B04 B05 B06 B16 natural_color --num-workers␣
↪8 -f /hrit/ahi/

geo2grid.sh -r ahi_hrit -w geotiff --ll-bbox 125 -15 160 10 -f /data/IMG_DK01*
```

4.7 AMI L1B Reader

The AMI L1B Reader operates on Level 1B (L1B) NetCDF files for the Advanced Meteorological Imager (AMI) instrument on board the Korean Meteorological Administration (KMA) GEO-KOMPSAT-1 (GK-2A) satellite. Geo2Grid determines if it supports files based on the input filename. Files for AMI supported by Geo2Grid usually have the following naming scheme:

```
gk2a_ami_le1b_vi004_fd010ge_201909251200.nc
```

The AMI L1B reader supports all instrument spectral bands, identified in Geo2Grid as the products shown in the table below. The AMI L1B reader can be provided to the main geo2grid.sh script using the `-r` option and the reader name

`ami_l1b`.

The list of supported products includes true and natural color imagery. These are created by means of a python based atmospheric Rayleigh scattering correction algorithm that is executed as part of the Geo2Grid AMI L1B reader, along with sharpening to the highest spatial resolution. For more information on the creation of RGBs, please see the *RGB section*.

Product Name	Description
VI004	0.47um Reflectance Band
VI005	0.51um Reflectance Band
VI006	0.64um Reflectance Band
VI008	0.86um Reflectance Band
NR013	1.37um Reflectance Band
NR016	1.61um Reflectance Band
SW038	3.83um Brightness Temperature Band
WV063	6.21um Brightness Temperature Band
WV069	6.94um Brightness Temperature Band
WV073	7.33um Brightness Temperature Band
IR087	8.59um Brightness Temperature Band
IR096	9.62um Brightness Temperature Band
IR105	10.35um Brightness Temperature Band
IR112	11.23um Brightness Temperature Band
IR123	12.36um Brightness Temperature Band
IR133	13.29um Brightness Temperature Band
true_color	Ratio sharpened rayleigh corrected true color
natural_color	Ratio sharpened corrected natural color
airmass	Air mass RGB
ash	Ash RGB

4.7.1 Command Line Usage

```
usage: geo2grid.sh -r ami_l1b -w <writer> [-h]
```

Examples:

```
geo2grid.sh -r ami_l1b -h

geo2grid.sh -r ami_l1b -w geotiff --list-products -f /data/gk2a/ami

geo2grid.sh -r ami_l1b -w geotiff --num-workers 8 -f /data/gk2a/ami

geo2grid.sh -r ami_l1b -w geotiff -p VI005 VI006 VI008 NR013 NR016 true_color -f gk2a_
↪ami*.nc

geo2grid.sh -r ami_l1b -w geotiff --ll-bbox 125 -15 160 10 -f /ami_data
```

4.8 GLM L2 Reader

The GLM Level 2 Reader operates on Gridded GLM Level 2 NetCDF files from the GOES-16 (GOES-East) and GOES-17/18 (GOES-West) Geostationary Lightning Mapper (GLM). The GLM L2 reader works off of the input filenames to determine if a file is supported by Geo2Grid. Files usually have the following naming scheme:

```
OR_GLM-L2-GLMF-M3_G16_s20192701933000_e20192701934000_c20192701934330.nc
```

These files can be generated by the CSPP Gridded GLM software which depends on the python `glmtools` library and GLM L2 LCFA inputs. The point data from these inputs is gridded to the ABI full disk, CONUS, or mesoscale sectors. The GLM L2 reader supports the products shown in the table below, but should allow for other products available from the GLM files to be loaded. The GLM L2 reader can be provided to the main geo2grid.sh script using the `-r` option and the reader name `glm_l2`.

Product Name	Description
flash_extent_density	Flash Extent Density
group_extent_density	Group Extent Density
flash_centroid_density	Flash Centroid Density
group_centroid_density	Group Centroid Density
average_flash_area	Average Flash Area
minimum_flash_area	Minimum Flash Area
average_group_area	Average Group Area
total_energy	Total Energy

4.8.1 Command Line Usage

```
usage: geo2grid.sh -r glm_l2 -w <writer> [-h]
```

Examples:

```
geo2grid.sh -r glm_l2 -h

geo2grid.sh -r glm_l2 -w geotiff --list-products -f /data/goes16/glm

geo2grid.sh -r glm_l2 -w geotiff --num-workers 8 -f /data/goes16/glm

geo2grid.sh -r glm_l2 -w geotiff -p flash_extent_density minimum_flash_area -f OR_GLM-
↪L2-GLMC*.nc

geo2grid.sh -r glm_l2 -w geotiff --ll-bbox -175 -2 -155 18 -f CG_GLM-L2-GLMF-M3_G18*.
↪nc

geo2grid.sh -r glm_l2 -w geotiff -p average_flash_area total_energy --num-workers 4 --
↪grid-configs=/home/g2g/my_grid.conf -g madison --method nearest -f /data/goes16/glm
```

CHAPTER

FIVE

COMPOSITES

Compositing is the process in Geo2Grid of combining multiple products together to make a new one. Most often this is done to make RGB color images like `true_color` and `false_color`. The most common RGB recipes are already configured internally to Geo2Grid, but users can make their own combinations too. The following instructions will go over some basic examples of how to make your own composites.

One type of composite that you may want to make is an image that combines one type of product for the night side of the data and another on the day side. An example of this type of day/night composite can be found in:

```
$GEO2GRID_HOME/etc/polar2grid/composites/abi.yaml
```

This `abi.yaml` file is meant to hold all custom user composites for the ABI instrument. There are separate .yaml configuration files for each supported instrument available in the `$GEO2GRID_HOME/etc/polar2grid/composites` directory. This `abi.yaml` file contains the `true_color_night` composite recipe which combines the visible reflectance daytime `true_color` composite with the nighttime ABI Channel 14 `C14` infrared 11 micron brightness temperatures into one image. The `abi.yaml` file contents are displayed below for reference:

```
sensor_name: visir/abi

composites:
  true_color_night:
    compositor: !!python/name:satpy.composites.DayNightCompositor
    prerequisites:
      - true_color
      - C14
    standard_name: day_night_mix
```

A composite recipe consists of 4 main parts:

1. **Name:**
   The name of the composite which will be used to request the product on the command line with the `-p` flag. In this example it is `true_color_night`. The name for a composite should be unique within a single yaml file or it may be overwritten.

2. **Compositor:**
   The `compositor` is a pointer to the python code that does the work of combining the products together. In this case we are using the `DayNightCompositor` code from the SatPy package. Another common option is the `GenericCompositor` for joining three bands together in to an RGB.

3. **Inputs:**
   The prerequisites are the products that are passed as inputs to this compositor. In the case of the day/night compositor the first product listed will be used for day time observations and the second product listed will be used for night time data.

4. **Standard Name:**
   Used later in Geo2Grid processing to map a composite to a particular enhancement or scaling. For the

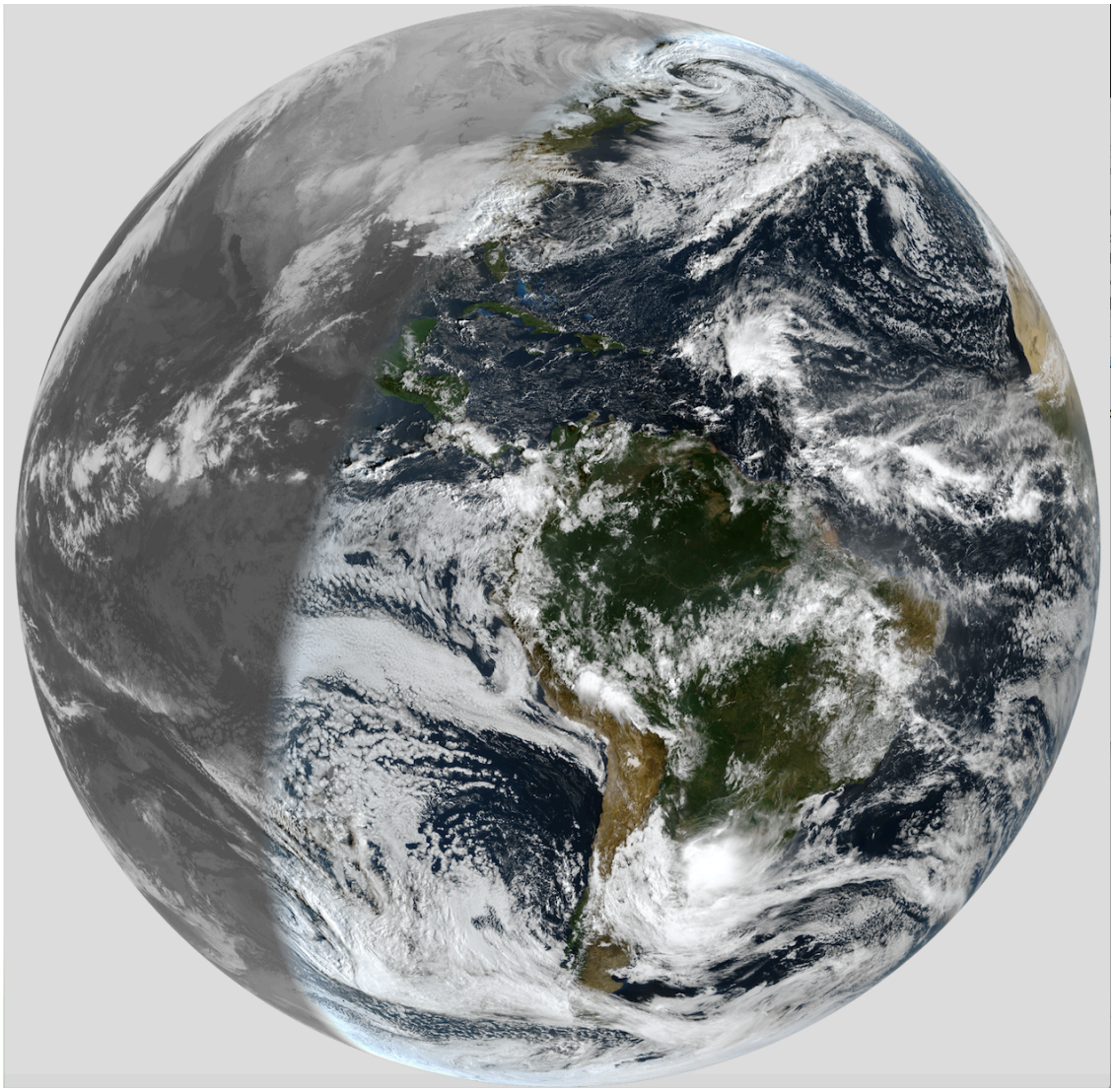
`DayNightCompositor` this should almost always be `day_night_mix`.

Once the composite recipe has been added to the `<instrument>.yaml` file it will appear in the list of available products when using the `--list-products` option under the `### Custom User Products` heading. It can then be invoked like any other product to `geo2grid.sh`.

The existing `true_color_night` composite can be modified directly or used as a template for additional composites. Make sure to change the composite name and what prerequisites are used in the composite. After that the composite can be loaded with your data by using the following command:

```
$GEO2GRID_HOME/bin/geo2grid.sh -r abi_l1b -w geotiff -p true_color_night -f /path/to/
↪files*.nc
```

The image created by executing the command on a GOES-16 ABI Full Disk dataset from 12:30 UTC, 12 November 2018 is shown below.



GOES-16 ABI true color day/Channel 14 brightness temperature night composite using input Full Disk observations from 12:30 UTC, 12 November 2018.

It is possible to use the compositor to combine RGBs as well. In the following example, I want to use the day/night compositor to combine the true color RGB for day data and the nighttime microphysics RGB for nighttime data. In this

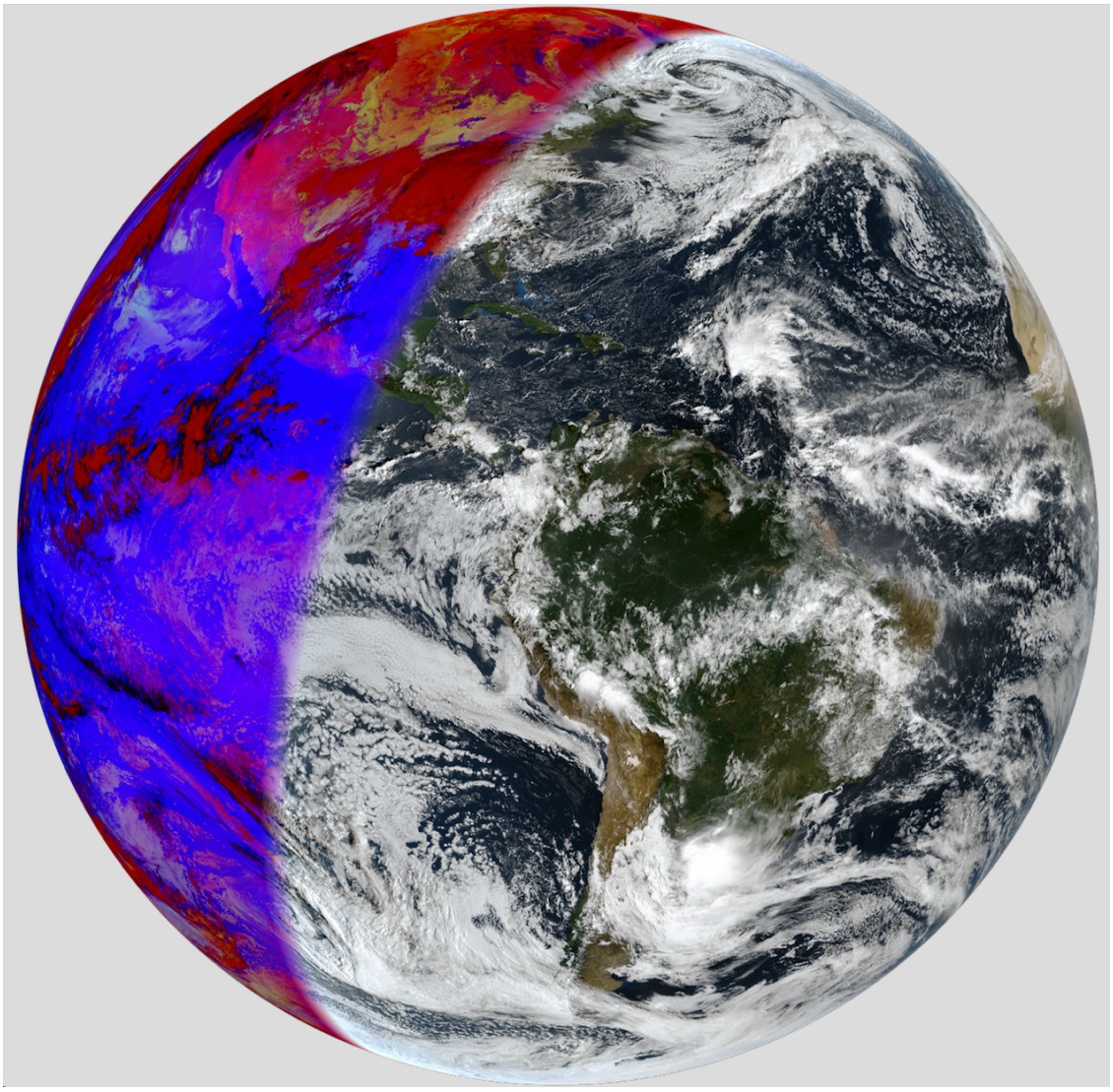
case, I can add the following lines to the `abi.yaml` file. Make sure to follow the formatting exactly, including the indentations.

```
true_color_night_microphysics:
  compositor: !!python/name:satpy.composites.DayNightCompositor
  prerequisites:
    - true_color
    - night_microphysics
  standard_name: day_night_mix
```

Once the .yaml files has been updated, the composite can be generated using the following command:

```
$GEO2GRID_HOME/bin/geo2grid.sh -r abi_l1b -w geotiff -p true_color_night_microphysics␣
↪-f /path/to/files*.nc
```

The image created by executing the command on a GOES-16 ABI Full Disk dataset from 12:30 UTC, 12 November 2018 is shown below.



GOES-16 ABI true color RGB day/nighttime microphysics RGB night composite using input Full Disk observations from 12:30 UTC, 12 November 2018.

CHAPTER

SIX

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 Geo2Grid 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. Geo2Grid offers various options that are described below. Defaults are also configured to provide a good result without any customization necessary.

6.1 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 Geo2Grid 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.

6.2 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`.

6.3 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.

6.4 Grids

Geo2Grid 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.

Geo2Grid 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)

Geo2Grid 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 Geo2Grid, but must be specified in SatPy's typical "areas.yaml" file.

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

6.5 Remapping and Grid Command Line Arguments

```
usage: polar2grid.sh -r <reader> -w <writer> [-h] [--method {native,nearest}]
                                             [-g [GRIDS [GRIDS ...]]]
                                             [--grid-coverage GRID_COVERAGE]
                                             [--cache-dir CACHE_DIR]
                                             [--grid-configs GRID_CONFIGS [GRID_
↪CONFIGS ...]]
                                             [--ll-bbox lon_min lat_min lon_max lat_
↪max]
                                             [--antimeridian-mode {modify_extents,
↪modify_crs,global_extents}]
                                             [--radius-of-influence RADIUS_OF_
↪INFLUENCE]
```

6.5.1 Resampling

--method Possible choices: native, nearest

resampling algorithm to use (default: native)

-g, --grids Area definition to resample to. Empty means no resampling (default: "MAX")

--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: ()

--ll-bbox Crop data to region specified by lon/lat bounds (lon_min lat_min lon_max lat_max). Coordinates must be valid in the source data projection. Can only be used with gridded input data.

--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

SEVEN

WRITERS

Writers are the final step in the Geo2Grid 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.

7.1 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.

7.1.1 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.

CHAPTER

EIGHT

UTILITY SCRIPTS

The following are scripts that can be used to aid in the creation of customized Geo2Grid products. All utility scripts are stored in the bin directory:

```
$GEO2GRID_HOME/bin/<script>.sh ...
```

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

8.1 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
```

8.1.1 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

8.1.2 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 `geo2grid.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:

```
geo2grid.sh -r abi_l1b -w geotiff --grid-configs /home/user/my_grids.yaml -g my_grid_
↪name -f <path_to_files>
```

8.2 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-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 ...]
```

8.2.1 Positional Arguments

input_tiff Input geotiff(s) to process

8.2.2 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

8.2.3 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

8.2.4 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

8.2.5 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

8.2.6 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

8.2.7 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 Tick interval in data units

Default: 5.0

--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 GOES-18_ABI_RadF_true_color_night_microphysics_20221115_123020_GOES-
↪West.tif --add-coastlines --add-rivers --rivers-resolution=h --add-grid -o abi_true_
↪color_coastlines.png
add_coastlines.sh --add-coastlines --add-borders --borders-resolution=h --borders-
↪outline='red' --add-grid GOES-17_ABI_RadF_natural_color_20181211_183038_GOES-West.
↪tif -o abi_natural_color_coastlines.png
```

8.3 Add Colormap

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

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

8.3.1 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 Geo2Grid 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 `geo2grid.sh` call which will add the colormap as a geotiff color table.

8.4 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  GOES-18_ABI_RadF_natural_color_20221115_183020_GOES-West.tif
```

8.5 Overlay GeoTIFF Images

The `overlay.sh` script can be used to overlay one GeoTIFF image (ex. Gridded Geostationary Lightning Mapper (GLM)) on top of another image (ex. GOES infrared brightness temperature Image). 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 GOES ABI AIT Level-2 Cloud top Tempetaure Product on top of a GOES ABI Band 14 brithtness temperature image.

```
 overlay.sh GOES-17_ABI_RadF_C14_20221123_183031_GOES-West.tif GOES-17_ABI_TEMP_
↪20221123_183031_GOES-West.tif abi17_fd_overlay.tif

overlay GOES-18_ABI_RadF_true_color_20221110_190020_GOES-West.tif  GOES-18_GLM_flash_
↪extent_density_20221110_190000_GOES-West.tif overlay_true_color_flash_extent_
↪density.tif
```

8.6 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
```

8.7 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. Geo2Grid 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`).

8.8 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
```

8.8.1 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

NINE

VERIFYING YOUR GEO2GRID INSTALLATION

9.1 Executing the ABI Geo2Grid Test Case

To confirm a successful Geo2Grid installation, follow these instructions to create a set of GOES-16 ABI GeoTIFF files from a single time period.

Unpack the test data as shown in Section 2.2 and execute the following commands:

```
cd geo2grid_test/abi
mkdir work
cd work
geo2grid.sh -r abi_l1b -w geotiff --num-workers 8 --progress  -f ../input
```

The test case consists of a 7 band subset of Full Disk Level 1B GOES-16 ABI calibrated NetCDF files from 17:45 UTC on 19 December 2018. The dataset includes GOES-16 ABI Bands 1, 2, 3, 5, 7, 9, and 14.

In this test, the Geo2Grid software will create by default as many single band full resolution Full Disk GeoTIFF files as possible, as well as true and natural color images. The true and natural color images are by default atmospherically corrected and spatially sharpened to 500 m resolution in native satellite projection. As it is executing, a progress bar will update, and 8 CPU threads will be used if available. On modern computers, the execution should take around 4-6 minutes.

The output files are large, including the true and natural color images that are greater than 1 GB (21696 lines x 21696 elements).

The software uses ABI Band 1 (.47 micron), Band 2 (.64 micron) and Band 3 (.86 micron) reflectances to create a "Green" Band, which is in turn combined with Bands 1 and 2 to create the true color imagery. The natural color image is created by combining ABI Band 2 (.64 micron), Band 3 (.86 micron) and Band 5 (1.6 micron) reflectances. Both true and natural color images are sharpened to 500 m spatial resolution by using the textural information provided by Band 2 (.64 micron).

If the ABI Geo2Grid processing script runs normally, it will return a status code equal to zero. If the ABI Geo2Grid 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 the GeoTIFF files for all ABI Bands found. The output from our test system is shown below.

```
p2g_compare.sh output work
Comparing 'work/GOES-16_ABI_RadF_C01_20181219_174533_GOES-East.tif' to known valid␣
↪file 'output/GOES-16_ABI_RadF_C01_20181219_174533_GOES-East.tif'.
0 pixels out of 235358208 pixels are different
```

(continues on next page)

(continued from previous page)

```
Comparing 'work/GOES-16_ABI_RadF_C05_20181219_174533_GOES-East.tif' to known valid␣
↪file 'output/GOES-16_ABI_RadF_C05_20181219_174533_GOES-East.tif'.
0 pixels out of 235358208 pixels are different
Comparing 'work/GOES-16_ABI_RadF_natural_color_20181219_174533_GOES-East.tif' to␣
↪known valid file 'output/GOES-16_ABI_RadF_natural_color_20181219_174533_GOES-East.
↪tif'.
0 pixels out of 1882865664 pixels are different
Comparing 'work/GOES-16_ABI_RadF_C07_20181219_174533_GOES-East.tif' to known valid␣
↪file 'output/GOES-16_ABI_RadF_C07_20181219_174533_GOES-East.tif'.
0 pixels out of 58839552 pixels are different
Comparing 'work/GOES-16_ABI_RadF_C02_20181219_174533_GOES-East.tif' to known valid␣
↪file 'output/GOES-16_ABI_RadF_C02_20181219_174533_GOES-East.tif'.
0 pixels out of 941432832 pixels are different
Comparing 'work/GOES-16_ABI_RadF_C14_20181219_174533_GOES-East.tif' to known valid␣
↪file 'output/GOES-16_ABI_RadF_C14_20181219_174533_GOES-East.tif'.
0 pixels out of 58839552 pixels are different
Comparing 'work/GOES-16_ABI_RadF_C03_20181219_174533_GOES-East.tif' to known valid␣
↪file 'output/GOES-16_ABI_RadF_C03_20181219_174533_GOES-East.tif'.
0 pixels out of 235358208 pixels are different
Comparing 'work/GOES-16_ABI_RadF_C09_20181219_174533_GOES-East.tif' to known valid␣
↪file 'output/GOES-16_ABI_RadF_C09_20181219_174533_GOES-East.tif'.
0 pixels out of 58839552 pixels are different
Comparing 'work/GOES-16_ABI_RadF_true_color_20181219_174533_GOES-East.tif' to known␣
↪valid file 'output/GOES-16_ABI_RadF_true_color_20181219_174533_GOES-East.tif'.
0 pixels out of 1882865664 pixels are different
All files passed
SUCCESS
```

A montage of the 9 output GeoTIFF files (7 single band and 2 RGB images) is displayed below.

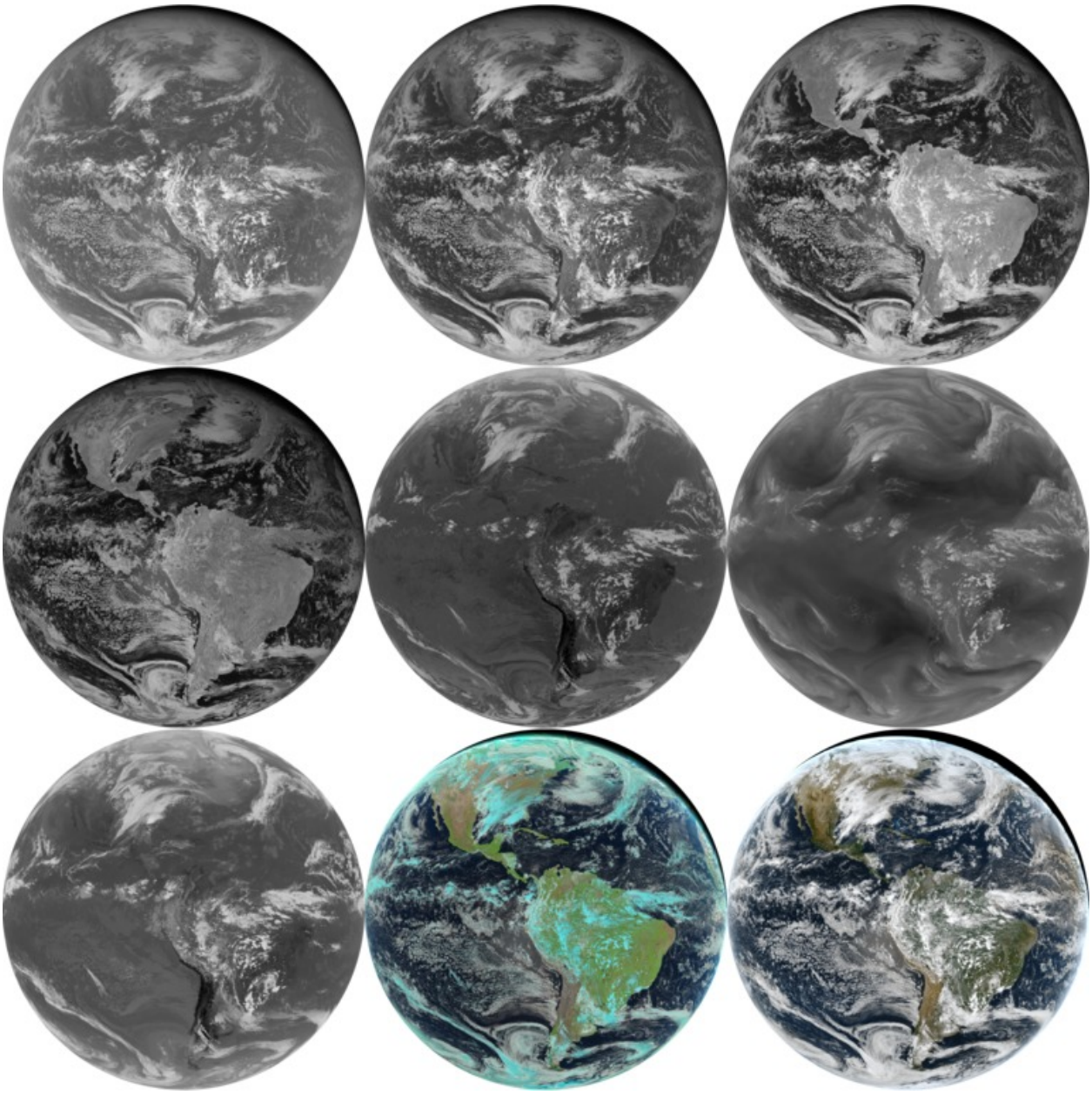


Fig. 9.1: GOES-16 ABI montage of images created from the Geo2Grid verification data observed on 19 December 2018 at 17:45 UTC. The images are from top to bottom, left to right, bands 1, 2, 3, 5, 7, 9, 14, natural color and true color.

CHAPTER

TEN

EXAMPLES

10.1 Working with ABI Files

This example walks through the creation of GOES ABI GeoTIFF subset image files and adding overlays.

10.1.1 The Basics of Geo2Grid for ABI GeoTIFF File Creation

Find the options available when creating GOES-16, -17 and -18 GeoTIFFs:

```
geo2grid.sh -r abi_l1b -w geotiff -h
```

List the products that can be created from your ABI dataset:

```
geo2grid.sh -r abi_l1b -w geotiff --list-products -f <path_to_files>
```

To create GeoTIFF output files of all bands found in your data set, including true and natural color full resolution sharpened 24 bit RGBs in standard satellite projection using 8 worker threads:

```
geo2grid.sh -r abi_l1b -w geotiff --num-workers 8 -f <path_to_files>
```

Create a subset of ABI band output Geotiff image files for Channels 1, 2, 3 and 5:

```
geo2grid.sh -r abi_l1b -w geotiff -p C01 C02 C03 C05 true_color -f
<path_to_abi_files>
```

Create ABI images over the given latitude/longitude region:

```
geo2grid.sh -r abi_l1b -w geotiff --ll-bbox <lonmin latmin lonmax lat-
max> -f <path_to_abi_files>
```

Create a natural color full resolution GeoTIFF from GOES-18 ABI observations acquired on 15 November 2022, 18:30 UTC over a latitude/ longitude bounding box of 128W,30N to -118W,40N . This command assumes that all bands required to create the false color image are available:

```
geo2grid.sh -r abi_l1b -w geotiff -p natural_color --ll-bbox -128 30 -118 40 -f OR_
↪ABI-L1b-RadF-M6C*_G18_s20223191830*.nc
```

The resulting image is displayed below.

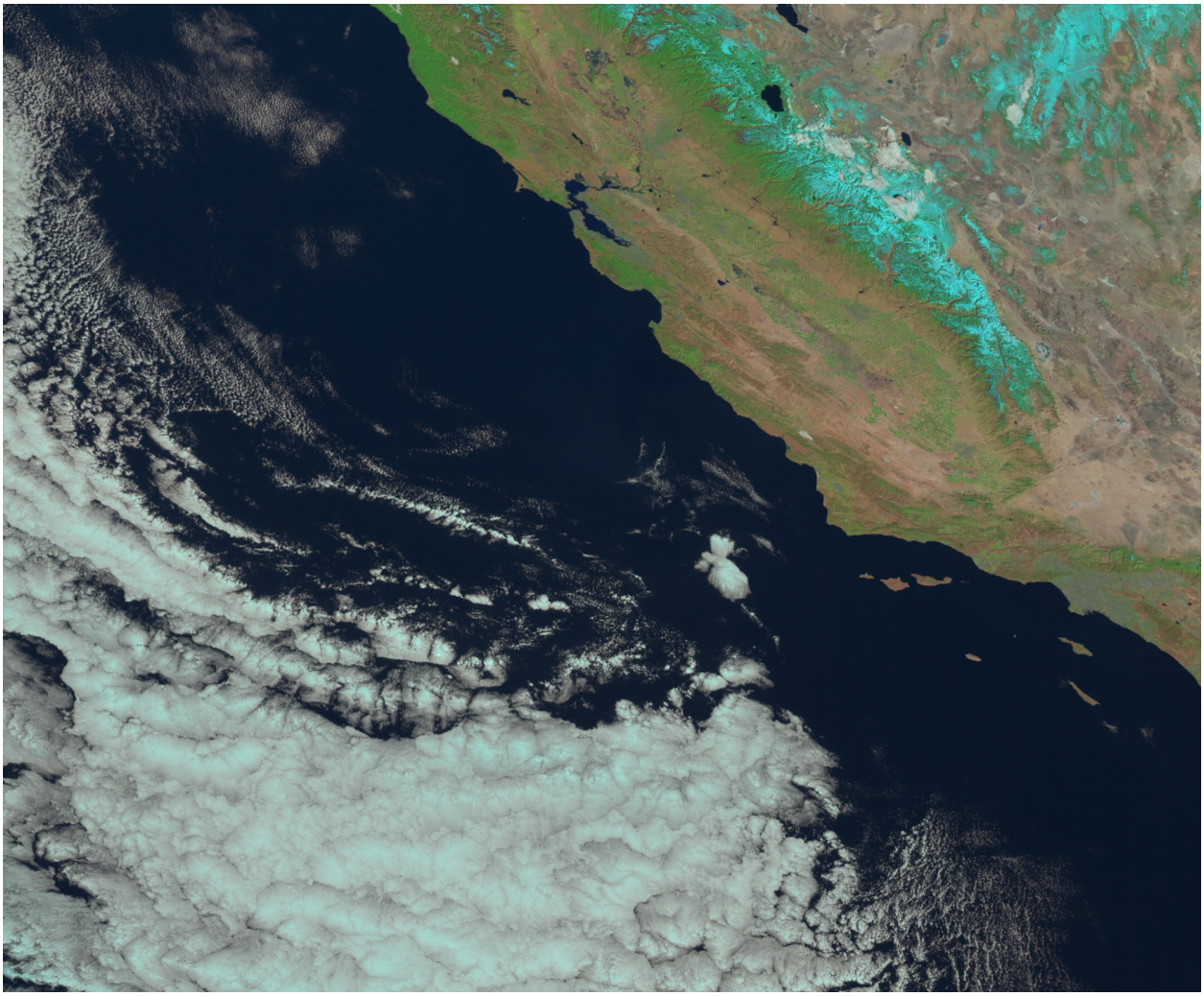


Fig. 10.1: ABI Natural color subset GeoTIFF image (GOES-18_ABI_RadF_natural_color_20221115_183020_GOES-West.tif)

Add coastlines, borders and latitude/longitude grid lines to the image, and write the output to the file "my_goes18_abi_naturalcolor.png":

```
add_coastlines.sh --add-coastlines --add-borders --borders-resolution=h --borders-
↪outline='red' --add-grid GOES-18_ABI_RadF_natural_color_20221115_183020_GOES-West.
↪tif -o my_goes18_abi_naturalcolor.png
```

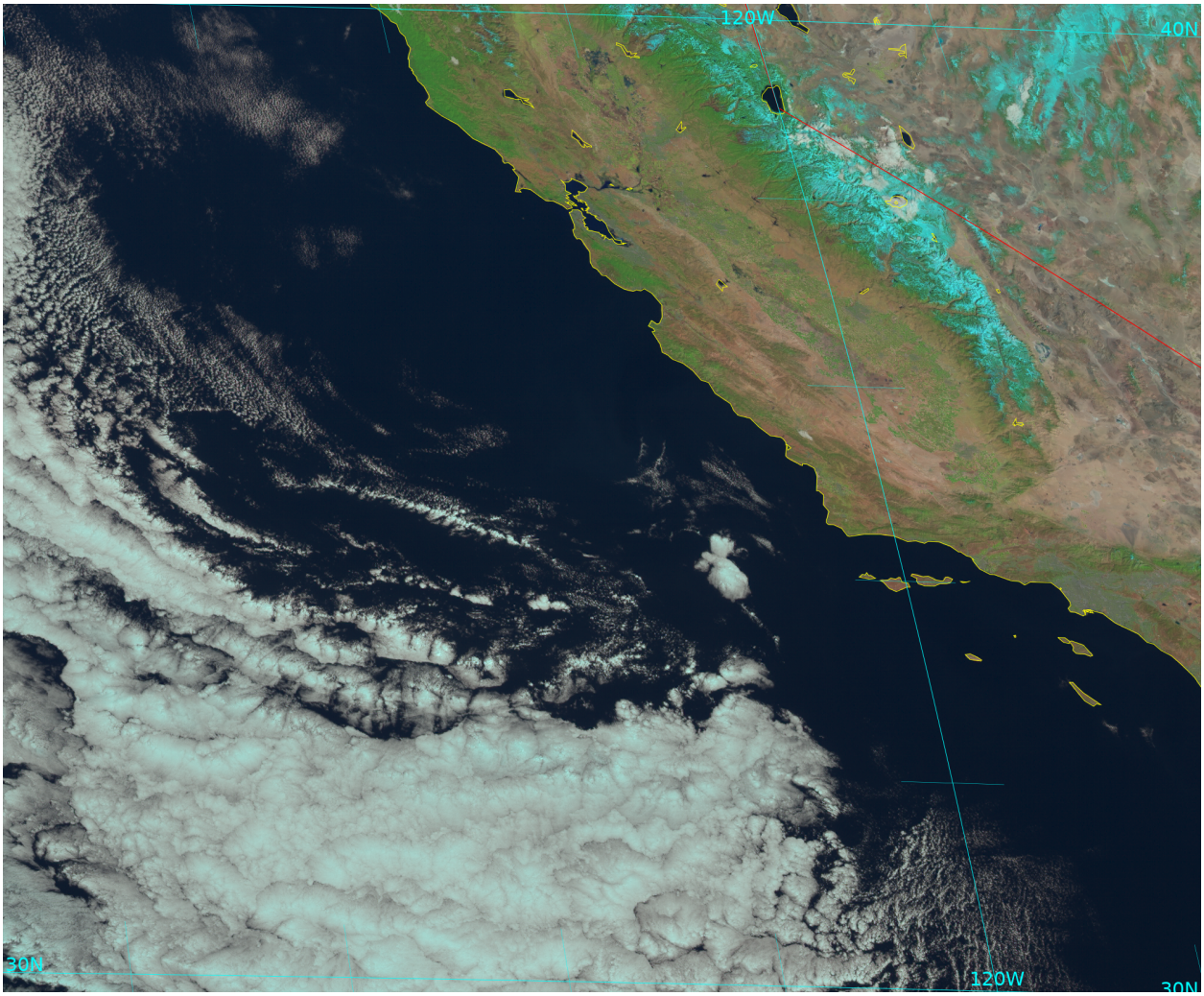


Fig. 10.2: GOES-18 natural color image with overlays (my_goes18_abi_naturalcolor.png).

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

```
gtiff2kmz.sh GOES-18_ABI_RadF_natural_color_20221115_183020_GOES-West.tif
```

which creates the *GOES-18_ABI_RadF_natural_color_20221115_183020_GOES-West.kmz* file which can then be displayed easily in the Google Earth GeoBrowser.

10.2 Working with ABI Level 2 Cloud Product Files

This example walks through the creation of GOES ABI Level 2 GeoTIFF product image files and adding overlays.

10.2.1 The Basics of Geo2Grid ABI L2 GeoTIFF File Creation

Find the options available when creating GOES-16, -17 and -18 ABI Level 2 GeoTIFFs:

```
geo2grid.sh -r abi_l2_nc -w geotiff -h
```

List the products that can be created from your ABI dataset:

```
geo2grid.sh -r abi_l2_nc -w geotiff --list-products -f <path_to_files>
```

Geo2Grid now supports two different ABI Cloud Products, cloud top height (HT) and cloud top temperature (TEMP). The height files use product names `ACHA` while the temp files using `ACHT`. For instance

```
CG_ABI-L2-ACHAC-M6_G17_s20223271831172_e20223271833556_c20223271834370.
nc
```

```
CG_ABI-L2-ACHTC-M6_G17_s20223271831172_e20223271833556_c20223271834370.
nc
```

You can provide Geo2Grid with both files for the same date/time and it will provide you the option to create both products with one execution.

To create GeoTIFF output files of both products found in this data set,

```
geo2grid.sh -r abi_l2_nc -w geotiff --num-workers 8 -f <path_to_files>
```

By default the products are color enhanced using the colormap:

```
$GEO2GRID_HOME/colormaps/abi_l2_modified_cloud_top.cmap.
```

Create just a Cloud Top Height Geotiff image:

```
geo2grid.sh -r abi_l2_nc -w geotiff -p HT -f <path_to_abi_files>
```

Create a Cloud Top Temperature image from the GOES-17 CONUS domain product created from 23 November 2022, 18:31 UTC ABI observations.

```
geo2grid.sh -r abi_l2_nc -w geotiff -p TEMP -f CG_ABI-L2-ACHTC-M6_G17_s20223271831172_
↪e20223271833556_c20223271834370.nc
```

The resulting image is displayed below.

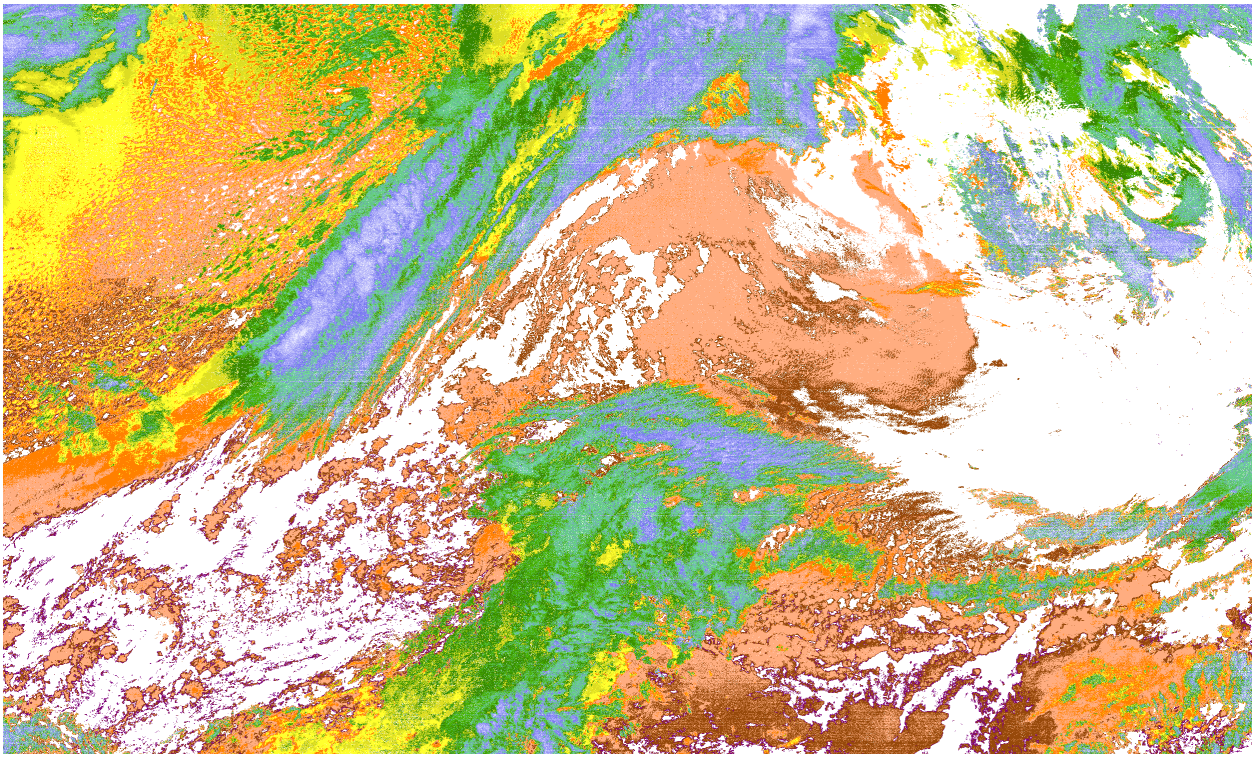


Fig. 10.3: CSPP GEO ABI AIT GOES-17 Cloud Top Temperature GeoTIFF image (GOES-17_ABI_TEMP_20221123_183117_GOES-West.tif)

Add a color table, coastlines, borders and latitude/longitude grid lines to the image, and write the output to the file `my_goes17_abi_ctt.png` using the `add_coastlines.sh` script. The script provides many options, including the ability to add a colorbar title using the font of your choice (provide the path to the font location on your local machine):

```
add_coastlines.sh GOES-17_ABI_TEMP_20221123_183117_GOES-West.tif --add-colorbar --
↪colorbar-text-color="black"   --colorbar-title="GOES-17 ABI Cloud Top Temperature␣
↪(°K)  23 November 2022  18:30 UTC"   --add-coastlines --coastlines-outline "black" -
↪-coastlines-level 1 --coastlines-resolution=i --add-borders --borders-level 2 --
↪borders-outline gray --coastlines-width 2  --colorbar-tick-marks 10 --colorbar-font␣
↪/usr/share/fonts/gnu-free/FreeSerifBold.ttf -o my_goes17_abi_ctt.png
```

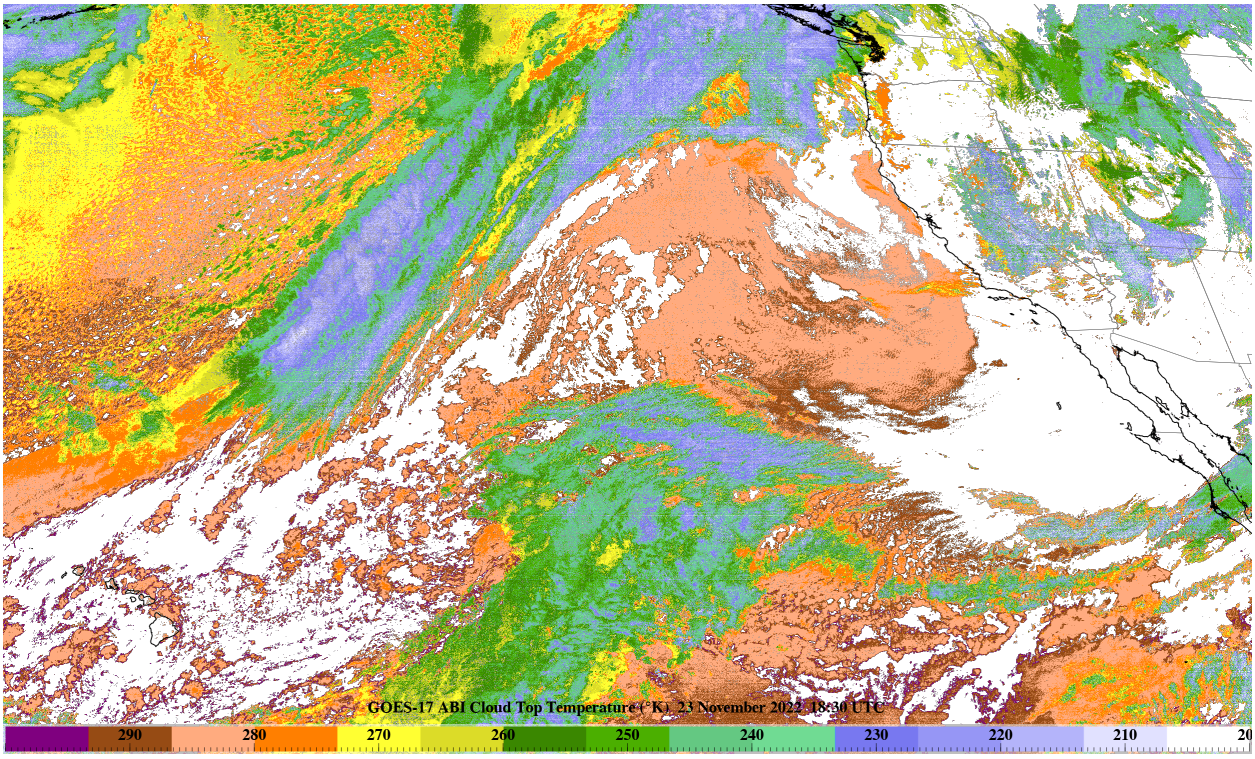


Fig. 10.4: CSPP GEO ABI AIT GOES-17 Cloud Top Temperature GeoTIFF image with overlays (my_goes17_abi_ctt.png).

Users can aslo overlay Level 2 images onto other GeoTIFFS. In the example execution below, we overaly the Cloud Top Temperature GeoTIFF image on top of the GOES-17 true color image from the same time and name the output GeoTIFF "goes17_overlay_true_color_cloud_temperature.tif".

```
overlay.sh GOES-17_ABI_RadC_true_color_20221123_183117_GOES-West.tif GOES-17_ABI_TEMP_
↪20221123_183117_GOES-West.tif goes17_overlay_true_color_cloud_temperature.tif
```

The new combined GeoTIFF is displayed below.

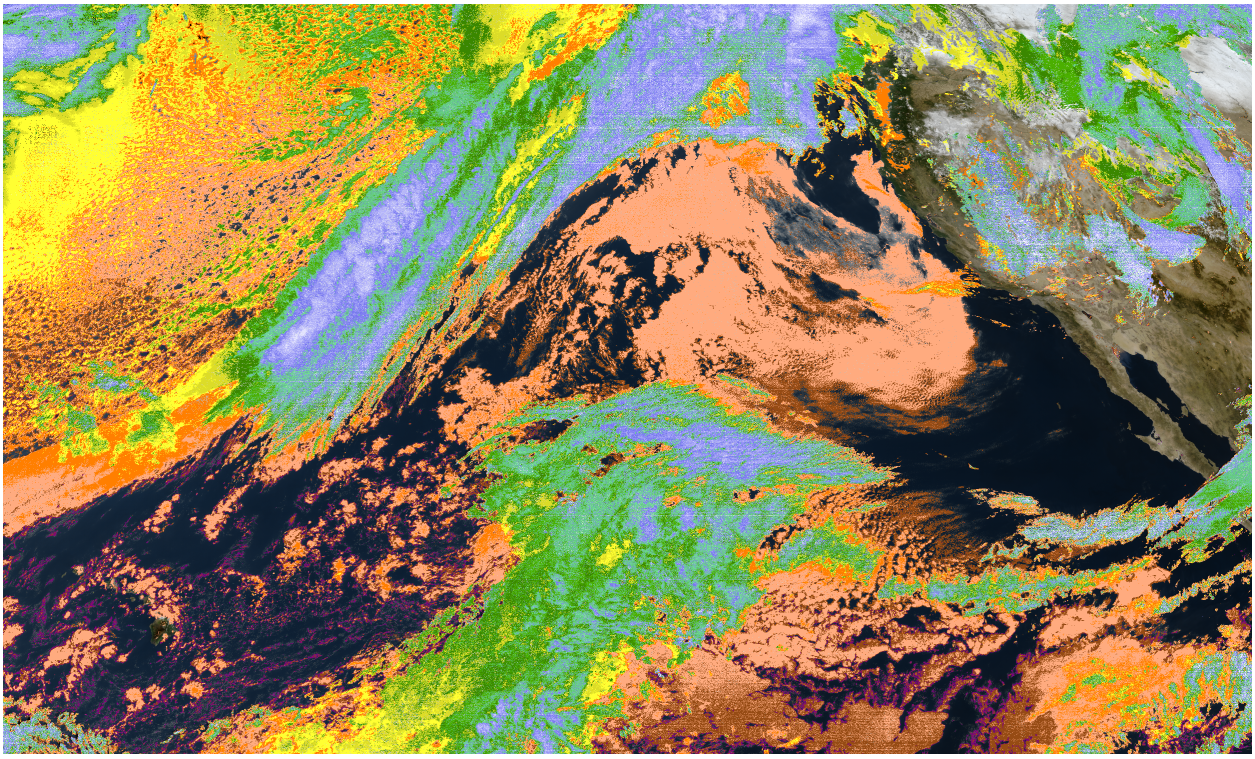


Fig. 10.5: CSPP GOES-17 ABI cloud top temperatures overlaid on the coincident true color image from 23 November 2022, 18:31 UTC (goes17_overlay_true_color_cloud_temperature.tif).

10.3 Working with AHI Files

This example walks through the creation of Himawari AHI GeoTIFF subset image files and adding overlays.

10.3.1 The Basics of Geo2Grid for AHI GeoTIFF File Creation

Find the options available when creating AHI HSD GeoTIFFs:

```
geo2grid.sh -r ahi_hsd -w geotiff -h
```

List the products that can be created from your AHI HSD dataset:

```
geo2grid.sh -r ahi_hsd -w geotiff --list-products -f <path_to_files>
```

To create GeoTIFF output files of all bands found in your data set, including true and natural color full resolution sharpened 24 bit RGBs in standard satellite projection using 8 worker threads:

```
geo2grid.sh -r ahi_hsd -w geotiff --num-workers 8 -f <path_to_files>
```

Create a subset of AHI band output Geotiff image files for Bands 1, 2, 3, 4 and 5:

```
geo2grid.sh -r ahi_hsd -w geotiff -p B01 B02 B03 B04 B05 natural_color
-f <path_to_ahi_files>
```

Create AHI images over a Lambert Conic Conformal (LCC) grid centered over Perth, Australia.

Run the grid helper script to define the grid center, areal extent, spatial resolution and projection .

```
p2g_grid_helper.sh perth 117.9 -32.4 500 500 1500 1500
```

```
perth:
  projection:
    proj: lcc
    lat_1: -32.4
    lat_0: -32.4
    lon_0: 117.9
    datum: WGS84
    units: m
    no_defs: null
    type: crs
  shape:
    height: 1500
    width: 1500
  center:
    x: 117.9
    y: -32.4
    units: degrees
  resolution:
    dx: 500.0
    dy: 500.0
```

Copy the output grid projection information into a grid configuration yaml file (my_grid.yaml). Use the grid to create an HSD AHI true color image from data observed on 12 November 2017, at 23:30 UTC.

```
geo2grid.sh -r ahi_hsd -w geotiff -p true_color --grid-configs /geo/hsd/my_grid.yaml -
↪g perth --method nearest -f /data/ahi8/hsd/2330/*FLDK*.DAT
```

The resulting image is displayed beow.

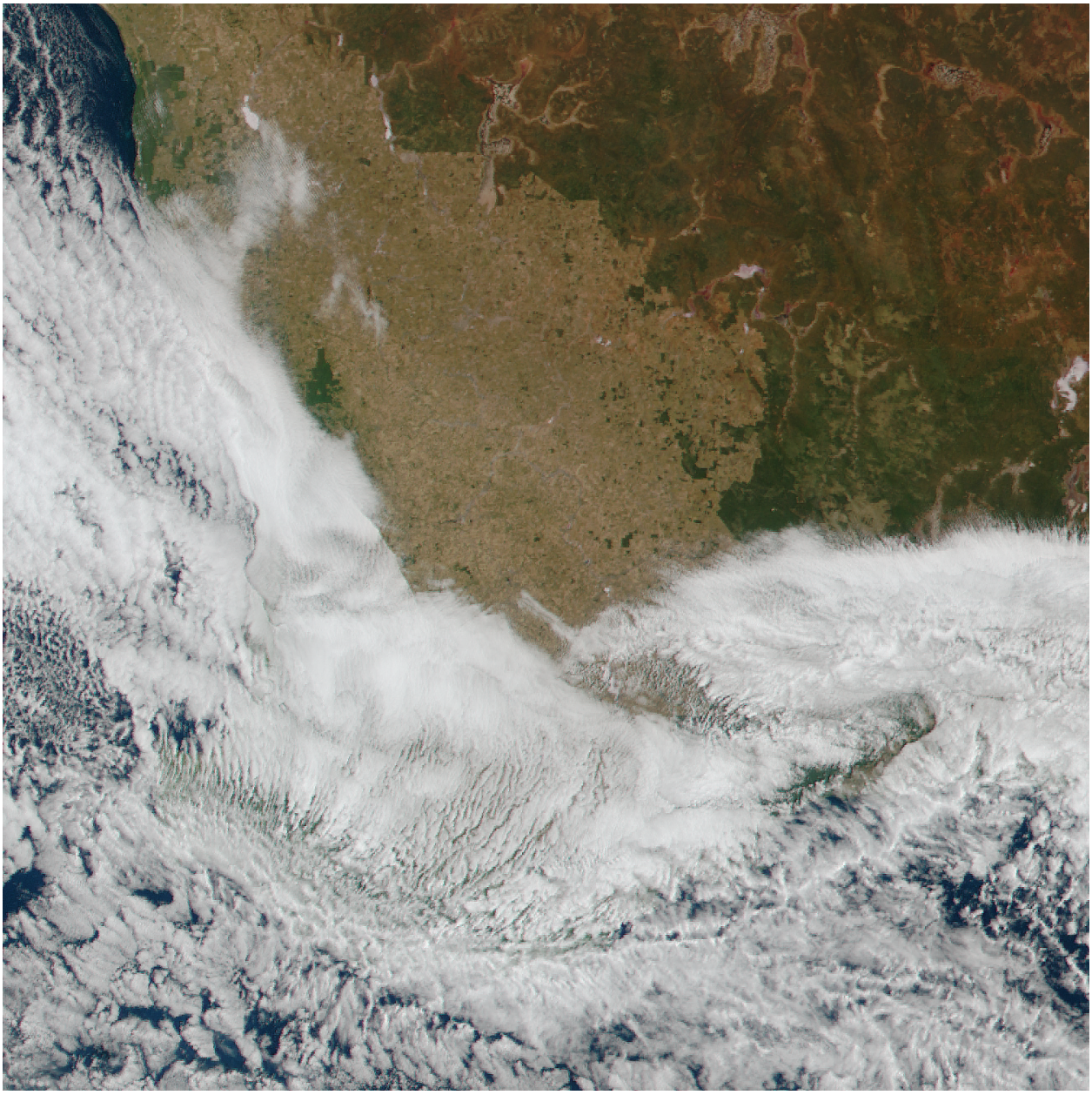


Fig. 10.6: AHI True color GeoTIFF image centered on Perth, Australia (HIMAWARI-8_AHI_true_color_20181112_233020_perth.tif).

Add coastlines, borders and latitude/longitude grid lines and rivers to the image.

```
add_coastlines.sh --add-coastlines --add-rivers --rivers-resolution=h --add-grid␣
↪HIMAWARI-8_AHI_true_color_20181112_233020_perth.tif
```

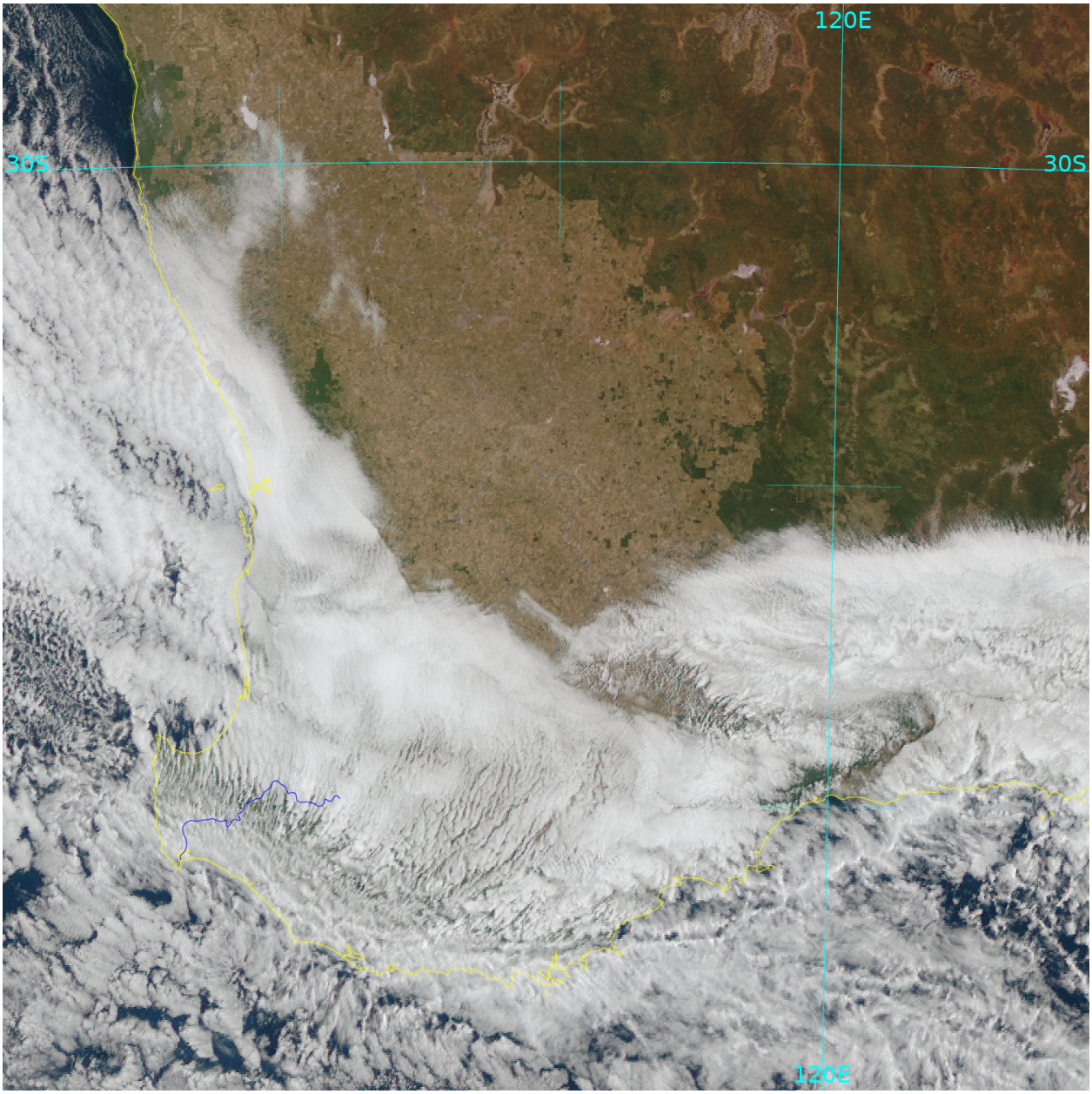


Fig. 10.7: Himawari-8 AHI true color image with overlays (HIMAWARI-8_AHI_true_color_20181112_233020_perth.png)

10.4 Using Geo2Grid to Create Animations

The advantage of Geostationary Satellites is the temporal resolution of the observations. Geo2Grid offers an easy interface for creating animations from Geo2Grid GeoTIFF files. The following example demonstrates how Geo2Grid software can be used to create an animation of files from a latitude/longitude subset of GOES-16 ABI CONUS GeoTIFF images located over the Southeastern United States.

Create a series of GOES-16 ABI GeoTIFF files from a time sequence of data. In the bash shell script example below, I use the ABI CONUS Band 1 files to search for all files we have available from 4 January 2019. The files for this day are all located in the same directory. I then create true and natural color images from all time periods that are available.

```
#!/bin/bash

# Set GEO2GRID environment variables

export GEO2GRID_HOME=/home/g2g/geo2grid_v_1_1
export PATH=$PATH:$GEO2GRID_HOME/bin

# Get input list of files/times based upon ABI Band 1 files

ls -1 /data/abi16/20190104/OR_ABI-L1b-RadC-M3C01_G16_s2019004*.nc > file_list.txt

sort_list=$(cat file_list.txt | sort)

# Make images for each time period available
for file in ${sort_list} ; do

        echo ${file}
        # get date/time for geo2grid file search
        datetime=`basename $file | cut -c27-38`
        echo "datetime :"$datetime

        # Cut out a box with lat/lon bounds of 23N, 105W to 37N 75W
        geo2grid.sh -r abi_l1b -w geotiff --ll-bbox -105 23 -75 37 --num-workers 8 -p␣
↪true_color natural_color -f /data/abi16/20190104/*${datetime}*.nc

done

exit 0
```

This script created 120 GeoTIFF images for my time period 10:00 UTC through 20:00 UTC, with a time step of every 5 minutes.

To create a 120 image animation, I use the Geo2Grid utility script `gtiff2mp4.sh`.

```
gtiff2mp4.sh my_true_color_animation.mp4  *true_color*.tif
```

The script wraps the `ffmpeg` video software, and combines all of the `*true_color*.tif` files found in the directory into an animation based upon defaults that make the output animations most compatible with modern video players. The output frame rate is 24 frames per second. The images will automatically be resized if they are large in order to ensure a smooth animation. I chose an output filename of `my_true_color_animation.mp4`. The software can also create animations from input `.png` files.

The figure below is the last image in the 120 loop sequence. The output MP4 animation is available for viewing at this site.

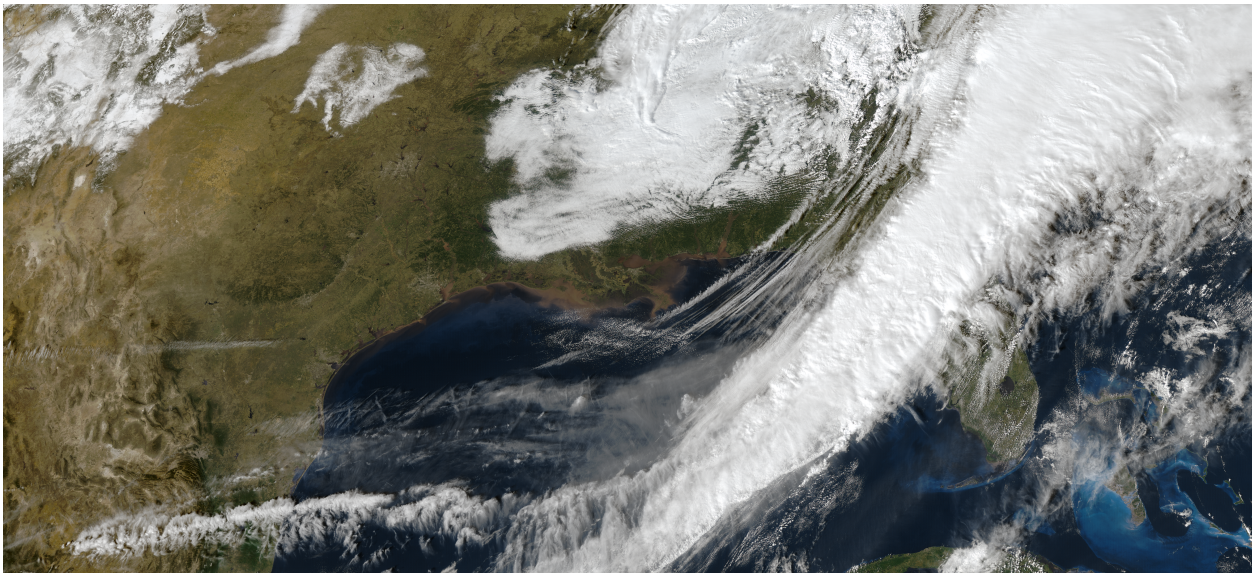


Fig. 10.8: The last GOES-16 ABI image from the 120 frame loop created with data from 4 January 2019. The image observations are from 19:57 UTC.

CHAPTER

ELEVEN

DATA ACCESS

Geo2Grid software is designed with the direct broadcast community in mind as the target end user. However, the software can be used by anyone to create images from standard mission compliant input files. NOAA GOES ABI data is now freely available from Cloud Service Providers (CSPs) including Amazon Web Services (AWS), Google and Microsoft Azure. More information about accessing data that is part of NOAA's Open Data Dissemination Program can be found at this website:

https://www.noaa.gov/nodd/datasets

And a very nice interface that allows direct downloads from the AWS can be found here:

http://home.chpc.utah.edu/~u0553130/Brian_Blaylock/cgi-bin/goes16_download.cgi

CHAPTER

TWELVE

GRIDS

Geo2Grid allows users to remap to one or more projected grids. A grid defines the uniform geographic area that an output image covers. Geo2Grid 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.

12.1 Provided Grids

Below are descriptions for a few of the grids provided with Geo2Grid. For information on all of the grids provided by Geo2Grid 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).

12.1.1 WGS84 Dynamic Fit

Grid Name
wgs84_fit

Description
Longitude/Latitude WGS84 Grid

Projection
EPSG:4326

Resolution
0.0057 degrees

12.1.2 WGS84 Dynamic Fit 250m

Grid Name
wgs84_fit_250

Description
Longitude/Latitude WGS84 Grid at ~250m resolution

Projection
EPSG:4326

Resolution
0.00225 degrees

12.1.3 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

12.1.4 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

12.1.5 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

12.1.6 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

12.1.7 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

12.1.8 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

12.1.9 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

12.1.10 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

12.1.11 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

12.1.12 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

12.1.13 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

12.1.14 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

12.1.15 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

12.1.16 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]

12.1.17 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]

12.1.18 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]

12.1.19 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]

12.1.20 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]

12.1.21 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]

12.1.22 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]

12.1.23 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

THIRTEEN

CUSTOM GRIDS

Geo2Grid provides a set of grids to suit most use cases, but sometimes these grids are not enough. This is why Geo2Grid 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 Geo2Grid to calculate these values from the data.

13.1 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 `geo2grid.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 Geo2Grid. Follow the *Grid Configuration File Format* section below. The grid file is in the YAML text format.
3. Call the `geo2grid.sh` script and add the command line option `--grid-configs grids.conf <your-file.yaml>`. The builtin grids in Geo2Grid are included when "grids.conf" is provided. If you would like only your grids and not the Geo2Grid provided grids don't include the "grids.conf" in the command line option.

Geo2Grid 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.

13.2 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 Geo2Grid bundle at:

`$GEO2GRID_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.

```yaml
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 Geo2Grid.
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

FOURTEEN

IMAGE PROCESSING TECHNIQUES

Many composites in Geo2Grid 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 Geo2Grid. See the various *Readers* documentation for more information on what products are available and descriptions of what corrections are used.

14.1 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, Geo2Grid 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 Geo2Grid to make invalid or missing data values transparent (completely opaque or completely transparent).

14.2 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.

14.3 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 Geo2Grid. 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.

14.4 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

14.5 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.

14.6 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.

14.7 Non-linear True Color Scaling

As a final step for some RGB images, Geo2Grid 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

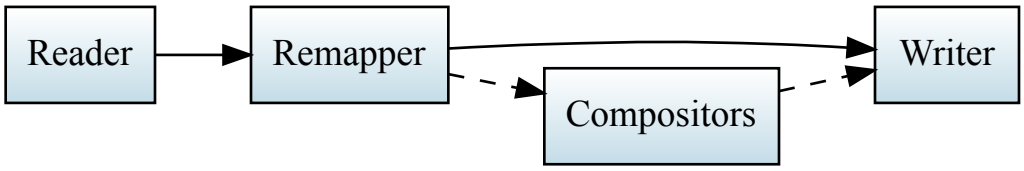
APPENDIX

A

SOFTWARE DESIGN OVERVIEW

The primary goal of Geo2Grid 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.

Geo2Grid 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 Geo2Grid 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 Geo2Grid 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.

A.1 Data Container

Geo2Grid, 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.

A.2 Readers

The Reader is responsible for reading provided data files to create Geo2Grid 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.

A.3 Compositors

Compositors are an optional component of Geo2Grid 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.

A.4 Remapping

Remapping is the process of putting satellite data pixels into an equidistant grid for easier viewing, manipulation, and storage. Geo2Grid currently offers multiple different algorithms for achieving this gridding. See the *remapping documentation* for more information.

A.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*.