Experimental Global Vegetation Index Processing System: GVI-x version 0.05

Aleksandar Jelenak
I. M. Systems Group, Inc.
Aleksandar.Jelenak@noaa.gov

October 1, 2004

Contents

1 Overview **1**

2 About the GVI-x System **1**
2.1 The GVI-x Grid 2
2.2 What Input Data GVI-x Requires 2
2.3 Compositing Period Computation 2
2.4 GVI-x Output 2
2.4.1 Output Data 3

3 GVI-x Processing **3**
3.1 Processing Options and Arguments 4

4 GVI-x Output File Format and Metadata **5**
4.1 HDF File Metadata 6
4.2 Conversion of Scaled Data 6

1 Overview

This document describes the GVI-x, version 0.05—an experimental, new global vegetation index (GVI) processing system for the Global Area Coverage (GAC) AVHRR data. Explanations are given for the processing algorithm, its options, required input data, and format of the output data.

2 About the GVI-x System

The GVI-x processing system comprises of a single software application, written in the C programming language primarily for the GNU/Linux operating system. It acts as a data aggregator, ingesting data for one NOAA satellite from the Clouds from AVHRR (CLAVR-x) processing system. GVI-x output is a subset of orbital AVHRR data extracted from the CLAVR-x output and collated onto a global GVI-x grid.

It is important to note that GVI-x does not processes GAC AVHRR Level 1b orbital data itself. Instead, it relies on CLAVR-x for this task. GVI-x uses the orbital *pixel-level* CLAVR-x output.

2.1 The GVI-x Grid

The GVI-x grid is based on the Plate Carreé map projection. It spans from 75.024° (north edge) to -55.152° (south edge) in the latitudinal direction; it's longitudinal range is from -180° (west) to 180° (east). The choice of latitude range values is such that it allows seemless integration into the full latitude range (from 90° to -90°) while preserving the traditional range for GVI processing: 75° to -55° degrees.

Three different GVI-x grid resolutions are supported, expressed in nominal grid cell length at the Equator as: 16-km (grid dimension: 904×2500), 8-km (1808×5000), and 4-km ($3616 \times 10,000$).

2.2 What Input Data GVI-x Requires

GVI-x needs data from the CLAVR-x NAV, OBS, and GEO files for each GAC Level 1b orbit. Absence of any of the three files will cause GVI-x to stop further processing.

GVI-x reads pixel geolocation information from NAV files. Pixel sensor, solar, and relative azimuth angle data are read from GEO files. OBS files provide GVI-x with pixel observation data, such as: channel reflectance or brightness temperature, CLAVR-x cloud mask, channel calibration information, etc.

2.3 Compositing Period Computation

GVI-x supports two methods for computing what days belong to a certain compositing period. They are referred to as *traditional* and *default*.

With the traditional method, compositing periods are seven days long (weekly composites), starting on Monday and ending next Sunday. The compositing period must have at least four days in the same year or is considered as belonging to the neighbouring years.

The new, default, method offers more flexibility by allowing the compositing period length to vary from one to 127 days. Counting periods begins on the first day of the year. A period must have at least four days in the same year, otherwise, the program refuses to process.

2.4 GVI-x Output

GVI-x output is a single file for each processing period in the Hierarchical Data Format (HDF). This is the same file format used for CLAVR-x output. Metadata for each GVI-x output variable and the period itself is present in this file. The metadata is compatible with the CLAVR-x HDF content guidlines and partially supports the generic HDF content requirements (predefined attributes).

Output files are named according to the following scheme:

GVIX_*id*_G*rr*_Y*yyyy*_P*pp*_D*ddd*.hdf

where: *id*—the two-letter spacecraft ID; *rr*—two characters describing grid resolution; *yyyy*—four-digit year of the first processed day; *pp*—two-digit period number; and, *ddd*—three-digit Julian date of the first processed day. Example: `GVIX_NL_G04_Y2003_P22_D146.hdf` holds the

GVI data from the NOAA-16 satellite, on the 4-km grid resolution, from the year 2003, 22nd period, starting on the 146th day of the year.

2.4.1 Output Data

The list of GVI-x output variable names and their descriptions is given below:

- `composite_years`: The year for each day of the composite period.
- `composite_julian_days`: Julian dates for each day of the composite period.
- `cell_jday`: Julian date of the pixel placed in the GVI-x grid cell.
- `cell_time`: Fractional hour of the day of the pixel placed in the GVI-x grid cell.
- `sensor_zenith`: Sensor zenith angle.
- `solar_zenith`: Solar zenith angle.
- `relative_azimuth`: Relative azimuth angle.
- `packed_cloud_mask`: CLAVR-x cloud mask with auxiliary data, packed into an 8-bit number. The meaning of the bits is:

Bit order	Meaning
(LSB) 0	CLAVR-x invalid (1) / valid (0) pixel flag
1	Day (1) / night (0) flag
2	Land (1) flag
3	Coast (1) flag
4	Glint (1) flag
5	Snow (1) flag
6,7	Cloud mask: clear (0); mixed-clear (1); mixed-cloudy (2); cloudy (3)

- `ch1_count`: Channel 1 count.
- `ch2_count`: Channel 2 count.
- `ch4_temperature`: Channel 4 brightness temperature.
- `ch5_temperature`: Channel 5 brightness temperature.

3 GVI-x Processing

GVI-x processing consists of two stages: daily and periodical compositing. For the purpose of this description, period will mean a certain number of consecutive days which may extend into the following year.

First, pixels from orbital files for each day are composited onto the GVI-x grid using the following set of rules:

1. Pixel is located inside the GVI-x grid;

2. Pixel is over land;

3. Pixel's solar zenith angle is smaller than the threshold;

4. Pixel's normalized vegetation index (NDVI) is computed;

5. Pixel's sensor (a.k.a. satellite) zenith angle is smaller than the value stored for that grid cell.

If all of the above conditions are met, the pixel "wins" the grid cell, and its information replaces the grid cell's content.

When all of orbital files for the day are processed, the day's composite data is compared with the period's based on this set of rules:

1. Grid cell belongs to the day in question (orbital files at the end of the day usually cross into the next—such cases are rejected);

2. Cell's NDVI is larger, i.e. the cell is "greener", than the cell's value of the period's composite.

If both of the above conditions are met, the day's composite data for this grid cell replaces the period's grid cell content.

When all of the days in the period are processed, the resulting period's composite data are saved in a GVI-x HDF output file.

Table 1: GVI-x command-line options.

Option format	Meaning
`-h`	Print help message and exit.
`-i` *`directory`*	Base input directory; GVI-x assumes that required input files are located in the *`obs`*, *`nav`*, and *`geo`* subdirectories. Default: current directory.
`-n` *`number`*	Minimal number of orbital files per day to continue processing. Default: 14.
`-o` *`directory`*	Output directory; must exist before program execution. Default: current directory.
`-p` *`number`*	Number of days per period. Allowed range: 1–127. Default: 7.
`-r` *`number`*	Grid resolution descriptor; allowed values for *`number`* are: 4, 8, and 16. Default: 8.
`-t`	Selects traditional seven-day period processing. Begins on Monday; ends on Sunday; minimum of four days of the period in the same year.
`-z` *`number`*	Solar zenith angle threshold, in degrees. Default: 85.

3.1 Processing Options and Arguments

GVI-x system is designed to be very flexible and allows user control over input data selection, grid resolution, and some compositing variables. User control is implemented using the traditional

UNIX method of command-line arguments and options. Currently supported command-line options are listed in Table 1.

In addition to command-line options, GVI-x requires three input arguments: satellite ID, period of the year, and year. Satellite ID is specified in the form {n|N}*DD*, where *DD* is a one- or two-digit number, e.g. n7, or N09, or n16, etc. Period of the year must be an integer greater than zero. Year must be in the four-digit format. The order of input arguments on the command-line is arbitrary.

An example how GVI-x can be started:

```
gvix -i /my/input/dir -n 10 -p 5 2001 n14 34
```

This translates to: CLAVR-x data is stored in the /my/input/dir directory; allow 10 or more orbital files per day; one processing period is five days long; year is 2001; satellite is NOAA-14; period of the year is 34.

Table 2: Description of GVI-x HDF global metadata.

Metadata tag	Description
PROCESSOR	Program that created the file; it is of the form: GVI-x_*version*.
CREATED	UTC date and time of file creation in the ISO 8601 format, e.g. 2004-09-16T15:49:08Z.
FILENAME	File's name in case it gets mangled during file manipulations.
HOST	Short name of the computer which ran GVI-x.
SATELLITE	Two-letter spacecraft ID for the NOAA satellite whose data was processed.
PERIOD_OF_YEAR	Ordered number of the period in the year; this value comes from the user-supplied command-line input argument.
DAYS_PER_PERIOD	Number of days in the period.
PROJECTION	Map projection of the GVI-x grid; value: Plate_Carree.
GRID_ROWS	Number of GVI-x grid rows.
GRID_COLUMNS	Number of GVI-x grid columns.
START_LATITUDE_RANGE	Latitude of the GVI-x grid's northern edge.
END_LATITUDE_RANGE	Latitude of the GVI-x grid's southern edge.
START_LONGITUDE_RANGE	Longitude of the GVI-x grid's western edge.
END_LONGITUDE_RANGE	Longitude of the GVI-x grid's eastern edge.

4 GVI-x Output File Format and Metadata

As previously stated, GVI-x saves all period composite data into one file in the Hierarchical Data Format (HDF). This file format stores not only data itself but allows for numerous metadata to be saved as well. Many features of the data can be described using the metadata content, making each HDF file self-explanatory.

In order to save disk space, all floating point data is scaled to two- or one-byte integers prior to saving in the HDF file. Scale factors are chosen in such a way that any quantization error

introduced by scaling is negligible. Further disk space saving is achieved by compressing the data using the GZIP algorithm. Reading such compressed data from HDF files is transparent to the user, except for slightly longer data access time and higher CPU load.

4.1 HDF File Metadata

GVI-x HDF file metadata content exists both for the file itself (so-called *global* metadata) and for each variable stored (*local* metadata). Global GVI-x metadata content is described in Table 2.

Local metadata content depends whether the variable is scaled or not. Table 3 lists all local metadata. CLAVR-x compatible metadata tags are in capital letters, while the lowercase tags correspond to the supported HDF predefined attributes.

Table 3: Description of GVI-x HDF local metadata.

Metadata tag	Description
`UNITS`	Units of the variable data.
`MISSING`	Value in the variable data indicating missing elements.
`SCALED`	Non-zero if variable data was scaled.
`RANGE_MIN`	Minimal data value excluding missing elements.
`RANGE_MAX`	Maximal data value excluding missing elements.
`units`	Same content as `UNITS`.
`coordsys`	Coordinate system of the data; value: `cartesian`.
`valid_range`	`RANGE_MAX` and `RANGE_MIN` as a two-element array.

If `SCALED` is non-zero, the following tags are also present:

`SCALED_MISSING`	`MISSING`'s counterpart for scaled data values.
`SCALED_MIN`	Minimal value of scaled data excluding missing elements.
`SCALED_MAX`	Maximal value of scaled data excluding missing elements.

4.2 Conversion of Scaled Data

Scaled data in the HDF file can be converted to its original values using a very simple algorithm given below:

1. Check the value of the `SCALED` local metadata tag. Continue if it's non-zero. (Otherwise, the data is not scaled.)

2. Identify all elements in the variable with `SCALED_MISSING` values.

3. Convert variable data according to the formula:

$$\texttt{val} = \frac{\texttt{RANGE\_MAX} - \texttt{RANGE\_MIN}}{\texttt{SCALED\_MAX} - \texttt{SCALED\_MIN}} \left(\texttt{sclval} - \texttt{SCALED\_MIN}\right) + \texttt{RANGE\_MIN}$$

   where `sclval` is the scaled value.

4. Assign `MISSING` value to all `SCALED_MISSING` elements using information from the previous steps.