

Release Notes GAMMA Software, 20230701

Urs Wegmüller, Christophe Magnard, Charles Werner, Andreas Wiesmann
Gamma Remote Sensing AG
Worbstrasse 225, CH-3073 Gümligen
<http://www.gamma-rs.ch>
1-Jul-2023

Introduction

This information is provided to users of the GAMMA software. It is also available online at https://www.gamma-rs.ch/uploads/media/GAMMA_Software_upgrade_information.pdf.

This release of the Gamma software includes new programs that provide new capability, additional features to existing programs and bug fixes.

Gamma Software on Linux, macOS, and Windows

The Gamma software has been compiled and tested on Linux (different distributions), Apple macOS Monterey (12.6) and Ventura (13.4), and Windows 10 and 11. Computationally intensive programs such as used in co-registration and resampling and geocoding have been parallelized using the OpenMP API built into the GCC compiler. Processing speed on Linux, macOS, and Windows systems is comparable.

Linux Distribution:

The Gamma software is developed on Ubuntu 22.04 LTS 64-bit Linux and is tested extensively with this distribution. The Gamma software is also available for Ubuntu 20.04 LTS.

Announcement: Support for Ubuntu 20.04 LTS will be available until the end-of-2023 upgrade.

Versions of the Software will also be uploaded for RHEL7 based on CentOS7, RHEL8 based on CentOS8, and RHEL9 based on Rocky Linux 9.

Note for **CentOS7**: Some Gamma Software programs now require a GDAL version newer than the version provided in CentOS7 standard repository. Hence, newer versions of GDAL and PROJ have to be installed. The installation method is described in the installation instructions (INSTALL_linux.html).

Announcement: Support for RHEL7 / CentOS7 will be available until the mid-2024 upgrade.

Note that Red Hat has ended support for CentOS8 at the end of 2021. Consequently, the Gamma software built for RHEL8 / CentOS8 uses the final CentOS8 release. This is the last time this version will be provided. In the next release (end of 2023), it will be replaced by a version based on Rocky Linux 8.

For installation instructions for the binary LINUX distributions see the HTML file `INSTALL_linux.html` (provided with the distribution E-mail or found in the main directory of the distribution).

Apple MacOS Distribution:

The software in this version has been compiled using macOS Monterey (12.6). You will need to install libraries such as GDAL using MacPorts. The build uses GCC 12 compiler on macOS Monterey. A version for Apple Silicon (M1/M2) is also available for macOS Ventura (13.4).

Announcement: This is the last time MacOS Monterey will be supported.

For installation instructions for the binary macOS distributions see the HTML file `INSTALL_macOS.html` (provided with the distribution E-mail or found in the main directory of the distribution).

Windows Distribution:

The Windows 10 and 11 version of the Gamma software is compiled with 64-bit support and multi-threaded. The build uses the MINGW64 GCC 12 compiler.

For installation instructions for the binary Windows distributions see the HTML file `INSTALL_win64.html` (provided with the distribution E-mail or found in the main directory of the distribution). Notice that installing the latest `GAMMA_LOCAL_w64` version is mandatory because a new GCC compiler and new libraries were used to build the software. Furthermore, the `.bashrc` file needs to be updated following the installation instructions.

On both Windows 10 and 11, it is also possible to install the Windows Subsystem for Linux (WSL) and run a Linux distribution of the Gamma software on that environment. Instructions for this setup are available in the HTML file `INSTALL_wsl.html` located in the main directory of the distribution.

Documentation and Program List

The Gamma documentation browser is an HTML based system for viewing the web pages and pdf documents. The documentation browser includes for each module a Contents sidebar on the right side of the screen and a search functionality.

The program `gamma_doc` facilitates the access to the documentation related to a given module or program:

<code>gamma_doc</code>	Opens the main page of the Gamma documentation browser and shows the program list.
<code>gamma_doc DIFF</code>	Opens the DIFF&GEO documentation.
<code>gamma_doc gc_map2</code>	Opens the reference manual web page for <code>gc_map2</code> .

Further information related to the GAMMA Software is available online:

General information:

gamma-rs.ch/uploads/media/GAMMA_Software_information.pdf

Technical reports, conference and journal papers:

gamma-rs.ch/uploads/media/GAMMA_Software_references.pdf

Release notes / upgrade information:

gamma-rs.ch/uploads/media/GAMMA_Software_upgrade_information.pdf

In case the program list is incomplete, run the python script `program_list.py` after successful installation of the Gamma Software in the main folder of the Gamma Software distribution:

```
./program_list.py Gamma_documentation_base.html Gamma_documentation_contents_sidebar.html -a
```

Python and Matlab wrappers

The Gamma Software is integrated into Python and Matlab through wrappers.

The `py_gamma` Python module permits a smooth usage of the Gamma Software within Python scripts as well as within a Python Interactive Development Environment (IDE) such as Spyder or PyCharm or using Jupyter Notebooks.

In the same way, the Matlab (and Octave) wrapper, composed of `mat_gamma` and `par_file` classes, permits a smooth usage of the Gamma Software within an interactive use of Matlab as well as within Matlab scripts.

Hardware Recommendations

Using multi-core processors (6 or more cores) will bring substantial improvement in processing speed due to parallelization of the code base. There should be at least 8 GB RAM available for each processor core with 16 GB per core recommended. Disk storage requirements for using the Gamma Software effectively depend on the amount of input data and data products that will be produced. Based on our experience we recommend considering at least 16 TB space, especially when working with stacks of Sentinel-1 or very high-resolution data (TerraSAR-X, Cosmo-Skymed) data. The current trend towards larger data products requires substantially increased storage capacities.

GAMMA Software Training Courses

A SAR/INSAR (MSP/ISP/DIFF&GEO/LAT) training at GAMMA (near Bern, Switzerland) is planned for 23. – 26. October 2023. A PSI (IPTA) training is planned to be held online with seven half-day sessions, in the period for 9. – 25. October 2023. See also our website under <https://www.gamma-rs.ch/software/training>.

Significant Changes in the Gamma Software Modules since the mid 2022 Release

Display program migration from GTK2 to GTK3

All display programs (*dis**, *gcp_ras*, *gcp_2ras*, *poly2kml*, *polyras*, *tree_edit*) were migrated from the GTK2 to the GTK3 library.

Furthermore, options have been added or completed to select a region of interest by specifying its starting pixel/line and its width/height. Previously, these options were typically only available for the "vertical" dimension (lines). This enables displaying parts of very large images that the GTK3 library wouldn't otherwise be able to display. The display size limit is 32767 pixels and/or lines.

The display programs in IPTA (*pdis**, *vu_disp**, *dis_data*, *dis_ipta*) were also migrated from the GTK2 to the GTK3 library.

Exporting IPTA results for use in GIS software

The export of IPTA results for use in GIS software has been enhanced.

The new program *pdata2gis* permits converting point data to vector file formats used in GIS software. The following output file formats are supported: GeoJSON, ESRI Shapefile, GeoPackage vector (GPKG), and Geography Markup Language (GML).

The new program *disp_tab2gis* permits converting displacement time-series text files (generated using either *disp_prt* or *disp_prt_2d*) to vector file formats used in GIS software. The following output file formats are available: GeoJSON, ESRI Shapefile, GeoPackage vector (GPKG), and Geography Markup Language (GML).

A new *field_names* option has been added to *disp_prt* and *disp_prt_2d*. It permits adding a first line containing field names, for simplified use as CSV file. Several options are provided for the date format in the displacement time-series field names.

Morphological operations

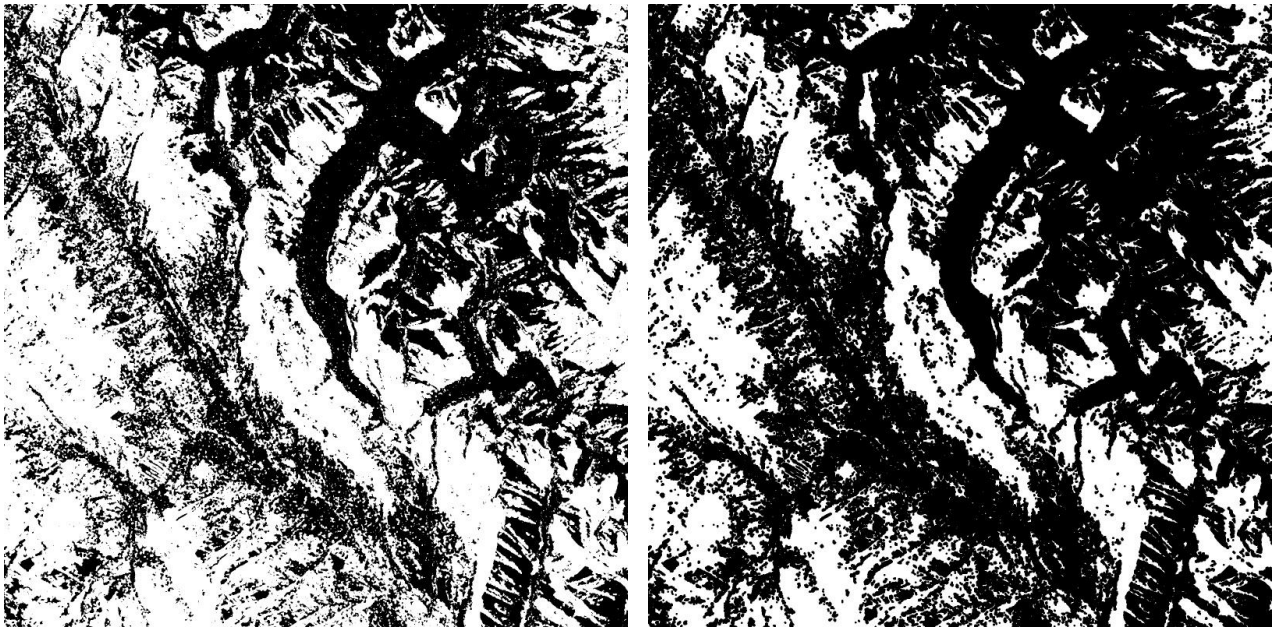
Morphological operations can now be performed using the new program *morph_op*.

Morphological operations adjust the values of the image pixels based on the pixels in their neighborhood. The neighborhood is defined by a structuring element, i.e., a window that informs which pixels of the neighborhood will be probed and which will not. The result of the

morphological operation shows how the shape defined in the structuring element fits or misses the shapes in the image.

The following morphological operations are available in *morph_op*: ***erosion***, ***dilation***, ***opening***, and ***closing***. Erosion will assign to the current pixel the minimum value within its neighborhood defined by the structuring element, effectively shrinking the bright areas. Dilation will assign to the current pixel the maximum value within its neighborhood as defined by the structuring element, effectively expanding the bright areas. Opening is defined as an erosion followed by a dilation, effectively removing narrow and small bright shapes. Closing is defined as a dilation followed by an erosion, effectively filling narrow and small dark shapes.

Operations using *flat* and *non-flat* structuring elements are supported.



(a) Coherence mask

(b) Coherence mask erosion

Figure 1. Coherence mask (a) and the same coherence mask after performing an “erosion” morphological operation using a small structuring element (b): the white areas have shrunk significantly.

Polarimetric decompositions starting from quad-pol SLC data

Support for polarimetry was complemented with the new Python program ***quad_pol_decomposition.py***, which permits conducting several polarimetric decompositions in a single step, starting from the quad-pol SLC. The intention is to strongly facilitate the access to this functionality. The new program makes it easy, for example, to apply several decompositions and to compare the results. To facilitate the intercomparison, we harmonized the scaling used (now primarily intensities and not magnitudes).

In addition, a related demo example has been added (“Gamma_Polarimetry_demo”, see below), containing an example for each of the quad-pol data from PALSAR-1, PALSAR-2, and SAOCOM.

Furthermore, the Polarimetry User Guide (Manual-POLSAR, in LAT/html directory) was updated.

SLC/MLI parameter files

We added a “polarization” parameter to the SLC/MLI parameter file. Typically, it consists of two letters (HH, HV, VH, VV, RH, RV, LH, LV, RL, LR, ...), with the first letter being the transmitted

polarization and the second letter the received polarization. The polarization is automatically written in the parameter file by the `par_*` programs, though some of them still need to be updated.

In addition, we increased the precision of `azimuth_line_time`, `range_pixel_spacing`, and `azimuth_pixel_spacing` parameters by one decimal.

DEM parameter files

We increased the precision of the `post_lat` and `post_lon` parameters (DEM pixel spacing in lat/lon coordinates) by two decimals.

Output TIFF files

A lossless LZW compression is now used when writing out TIFF files (e.g. using `ras...` programs).

Orbit data filtering: *ORB_filt_spline.py*

Nowadays all SAR satellites use GNSS measurements to determine orbit state vectors. The actual GNSS measurements include a significant noise term and so a processing is necessary to derive high quality orbit state vectors. While, we are used to getting high quality orbit state vectors for ESA, JAXA, and DLR missions, this is not always the case for some of the new SAR satellites. For SAOCOM and LT-1, for example, we observed that the state vectors provided with the SLC data deviate significantly from filtered (smoothed) state vectors.

We included the new program *ORB_filt_spline.py* to apply a suited filtering of state vectors. The program provides on one hand the filtered state vectors and on the other hand it provides tables and plots comparing the filtered and unfiltered values – to assess the quality of the unfiltered data and to decide if filtering is necessary, at all. Figure 2 shows the deviation of the initial orbit components from the filtered orbit for a SAOCOM (a) and Sentinel-1 (b) SLC.

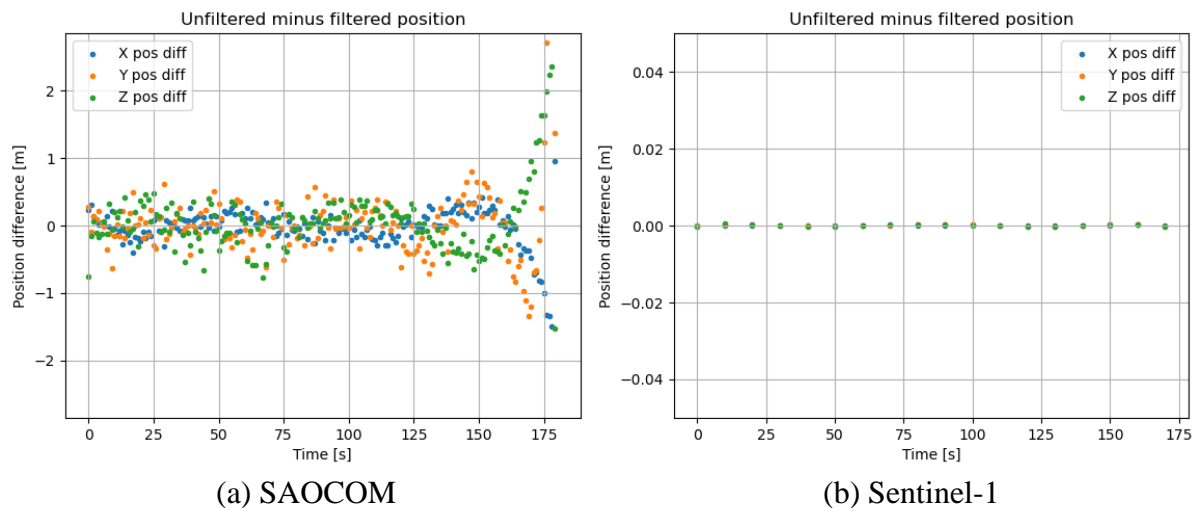


Figure 2. Difference between unfiltered and filtered orbit state vectors for a SAOCOM (a) and Sentinel-1 (b) SLC. In (b) the blue and orange dots are “behind” the green dots.

Python wrapper (*py_gamma*)

Using *py_gamma*, parameters of Gamma programs can now also be entered using keyword arguments: the input parameter names are used as the keywords. With this method, unused options don't have to be entered and readability may be improved. Keyword arguments can be entered in any order. A mix of positional and keyword arguments can also be used, with the positional arguments coming first.

The Python help, e.g., “`help(pg.disdt_pwr)`”, now shows all the keyword arguments of the Gamma program.

For this, a new database file named “`gamma_param.db`” containing the usage information of the Gamma programs and scripts has been added to the main repository of the Gamma Software.

The new function *bash2python* permits converting a bash command into the equivalent Python command using *py_gamma*. The output command uses per default positional arguments; by calling the function using the keyword argument “`use_kwargs = True`”, the output command will use keyword arguments.

The demo example “`py_gamma_demo`” has been updated and now contains both syntaxes (positional and keyword arguments). It also includes an example with the function *bash2python*.

Example (*disdt_pwr*):

Usage:

```
disdt_pwr <data> <pwr> <width> [ystart] [ny] [min] [max] [cflg] [cmap] [scale] [exp]
[bits] [xstart] [nx]
```

input parameters:

```
data      (input) data in FLOAT format (deformation, height, unwrapped phase,
           correlation...)
pwr       (input) intensity image coregistered with data (enter - for none, FLOAT or
           raster image)
width     number of samples/row of data and pwr
ystart    starting line of data and pwr (enter - for default: 1)
ny        number of lines to display (enter - or 0 for default: to end of file)
min       minimum data value (enter - for default: 0.0000e+00)
max       maximum data value (enter - for default: 1.0000e+00)
cflg      cyclic data display flag:
           0: display min <= data < max (default)
           1: display (data - min) modulo (max - min)
           2: autoscale between min and max value found in data
cmap      colormap file (enter - for default: hls.cm)
           NOTE: colormaps are text files in $DISP_HOME/cmaps, examples: cc.cm, rmg.cm,
           hls.cm, gray.cm, turbo.cm, BuYlRd.cm
           colormap swatches: $DISP_HOME/cmaps/*.png,
           $DISP_HOME/cmaps/cmocean/*.png, $DISP_HOME/colorcet/*.png
scale     intensity display scale factor (enter - for default: 1.0)
exp       intensity display exponent (enter - for default: 0.35)
bits      bits/pixel (enter - for default)
           8: 8-bit indexed color map (default)
           24: RGB 8-bits/color
xstart    starting pixel of data and pwr (enter - for default: 1)
nx        number of pixels to display (enter - or 0 for default: to end of line)
```

Bash command:

```
disdt_pwr NZ.EQA.20161018_20161115.diff.unw NZ.EQA.20161018.mli 8896 - - -6.28 6.28 1
rmg.cm - - 24
```

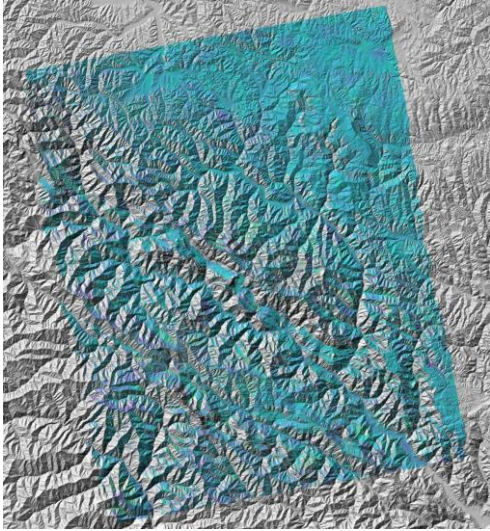
Python command with positional arguments:

```
pg.disdt_pwr('NZ.EQA.20161018_20161115.diff.unw', 'NZ.EQA.20161018.mli', 8896, '-', '-
', -2*np.pi, 2*np.pi, 1, 'rmg.cm', '-', '-', 24)
```

Python command with keyword arguments:

```
pg.disdt_pwr(data = 'NZ.EQA.20161018_20161115.diff.unw', pwr = 'NZ.EQA.20161018.mli',
width = 8896, min = -2*np.pi, max = 2*np.pi, cflg = 1, cmap = 'rmg.cm', bits = 24)
```

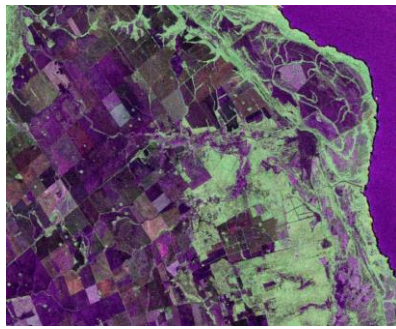

New and updated SAR data readers

New / updated SAR data reader	Short description
<p><i>par_LT1_SLC</i></p> 	<p>The program supports reading SLC data of the Chinese L-band Constellation LuTan-1 (LT1). The first LT1 SAR was launched in January 2022, followed by a second one in February 2022.</p> <p>Assessing data examples, we found that the orbit state vector data are sometimes quite noisy. Using the new ISP program <i>ORB_filt_spline.py</i>, it is possible to identify and reduce this noise.</p> <p>The image to the left shows a differential interferogram generated using an LT1A – LT1B pair with a time interval of 4 days and about 600m perpendicular baseline (after applying also a detrending and atmospheric phase removal). The high coherence indicates a very high potential of LT1 for interferometry.</p> <p>We describe our assessment and the processing done in a short technical report.</p>
<p><i>par_TX_GRD</i></p>	<p>The program was updated and now allows writing out the data as an MLI in slant range / azimuth geometry (new options <i>MLI_par</i> and <i>MLI</i>). The slant range pixel spacing can be manually specified using the new option <i>rps</i>.</p>

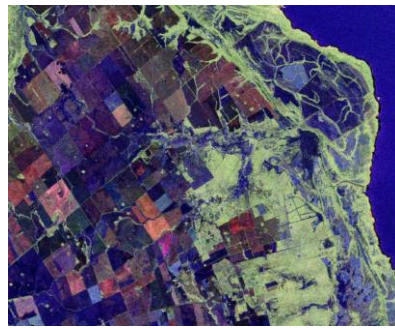
Gamma Software Demo examples

In this period again a Gamma Software Demo example was added. The access to the Gamma Software Demo examples is limited to Gamma Software users with a valid license. The access information is provided with the software delivery.

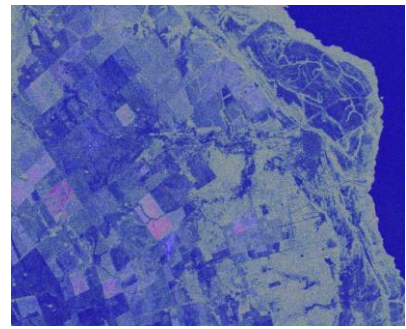
New / modified demo example:	Contents
<i>py_gamma_demo.tar.gz</i>	Update to include both syntaxes (positional and keyword arguments) when calling Gamma programs as Python functions, addition of an example for the new “bash2python” function.
<i>Gamma_Polarimetry_demo.tar.gz</i>	Demo on polarimetric decompositions, with examples using quad-pol. PALSAR-1, PALSAR-2, and SAOCOM SLC data (level 1.1). The demo shows the step-by-step processes for the different decompositions supported in the GAMMA LAT as well as the use of the new Python script <i>quad_pol_decomposition.py</i> that supports doing the decompositions starting from the quad-pol. SLC in a single step. Figure 3 shows the polarimetric decompositions for the SAOCOM example.



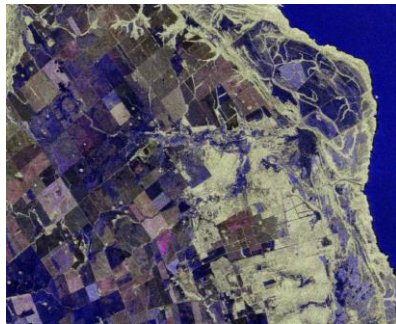
0: Scattering matrix components
(HH, HV, VV)



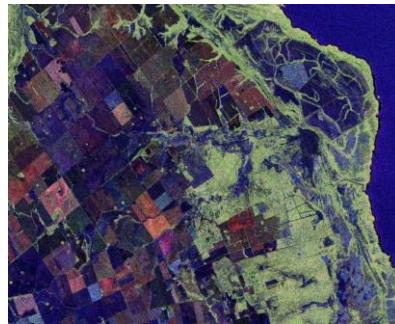
1: Pauli decomposition (beta,
gamma, alpha)



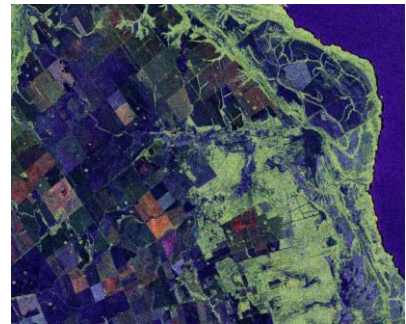
2: H/A/alpha decomposition
(entropy, anisotropy, alpha)



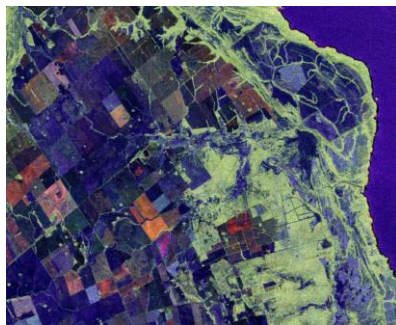
3: H/A/alpha eigenvalue
decomposition (λ_2 , λ_3 , λ_1)



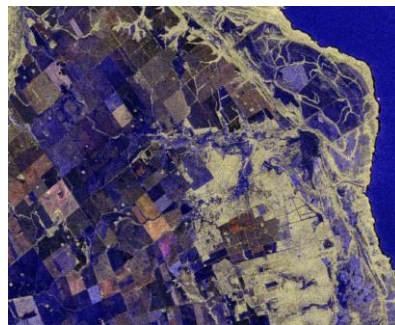
4: Cloude decomposition (2,3,1)



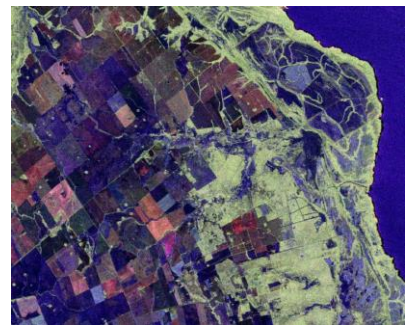
5: Freeman-Durden with complex
multi-looking (pd, pv, ps)



6: Freeman-Durden with single
complex look and subsequent multi-
looking (pd, pv, ps)



7: Krogager decomposition
(diplane, helix, sphere)



8: Average of Pauli, H/A/alpha
eigenvalue, and Freeman-Durden
decompositions (dih, vol, surf)

Figure 3. Polarimetric decompositions (0 to 8) supported by `quad_pol_decomposition.py`. SAOCOM quad-pol. SLC data are used as input. The size of the section shown is about 20km x 16km. In brackets abbreviations for the components used for the red, green, and blue color in the RGB composite are indicated.

MSP

PALSAR_proc: Updated to check consistency of the line header time in detecting possible missing lines. Updated to determine and fill gaps in raw data usually associated with changes in the range window position.

ISP

typedef_ISP.h, *ISP_io*: Added a "polarization" parameter to the SLC parameter file. Typically consists of two letters (HH, HV, VH, VV, RH, RV, LH, LV, RL, LR, ...), with the first letter being the transmitted polarization and the second letter the received polarization.

Increased precision of *azimuth_line_time*, *range_pixel_spacing*, and *azimuth_pixel_spacing* by one decimal.

ORB_filt_spline.py: New program to filter orbit state vectors using splines. This program is very useful for data acquired by new SAR satellites as SAOCOM or LT1. Besides the orbit state vector filtering the programs supports checking the state vector quality, e.g. by generating plots of the deviations of the unfiltered values from the filtered values and by providing statistical parameters.

par_TX_GRD: Added new options *MLI_par* and *MLI* to write output file as an MLI in slant range geometry. The slant range pixel spacing can be manually specified using new option *rps*.

par_LT1_SLC: New program for generating SLC parameter and image files of the Chinese L-band SAR constellation LuTan-1 (LT1) SLC data.

DIFF&GEO

gc_map2: No-data values in the input DEM are now temporarily filled to improve reliability of *gc_map2*. The method used is the following: the DEM is downsampled; no-data values are inpainted in the downsampled DEM; no-data values in the full-resolution DEM are interpolated from the filled downsampled DEM using B-spline interpolation. At the end of the processing, the output data files are set to 0 where the input DEM contained no-data values, as well as in close proximity to these no-data values (≤ 3 pixels).

gc_map2: The layover-shadow map now better covers the SAR acquisition in case large offsets are found in the *DIFF_par* file.

DIFF_io: Increased precision of *post_lat* and *post_lon* by two decimal (DEM pixel spacing in lat/lon coordinates).

typedef_DIFF.h, *DIFF_io*, *DIFF_lib*, *coord_trans*, *coord_trans_list*, *create_dem_par*: The Mercator (1SP) and Pseudo-Mercator projections are now supported.

geocoding.py: New options *--no_ls_map* and *--no_inc* to avoid generating layover-shadow map and/or incidence angle map were added.

LAT

m-alpha, *m-chi*, *m-delta*: The three decompositions (m-alpha, m-chi, m-delta) now use the following convention: For RCP illumination, c1 indicates single-bounce (odd) surface scattering; c3 indicates double-bounce (even) scattering. Conversely, for LCP illumination, c1 indicates double-bounce (even) scattering; c3 indicates single-bounce (odd) surface scattering. For both RCP and LCP illuminations, c2 is the depolarized component and indicates volume scattering.

morph_op: New program to perform morphological operations (erosion, dilation, opening, closing) on real-valued data.

quad_pol_decomposition.py: A new script to calculate Pauli, H/A/alpha, Cloude, Freeman-Durden, and Krogager quad-pol decompositions from full pol. SLC data (at HH, VV, and HV or

VH polarization) in a single step, was added. The script generates the 3 output elements of the selected decomposition as well as an RGB composite. Additional output files are also written when useful. In addition, an average over several different decompositions can be calculated (providing a similar result but with slightly less noise).

DISP

all display programs: dis, gcp_ras, gcp_2ras, poly2kml, polyras, tree_edit*: The display programs now display images using the GTK3 library (previously GTK2). Options have been added or completed to select a region of interest by specifying its starting pixel/line and its width/height. Previously, these options were typically only available for the "vertical" dimension (lines). This permits displaying parts of very large images that the GTK3 library wouldn't otherwise be able to display. The display size limit is 32767 pixels and/or lines.

dis2dt_pwr, dis2mph, dis2mph_pwr: We added a new option to display images using 8 or 24 bits color scale.

DISP_lib: TIFF files are now written using lossless LZW compression.

data2geotiff: The Mercator (1SP) and Pseudo-Mercator projections are now supported.

IPTA

mb, mb_pt: A new option to specify the temporal reference of the output time-series was added. So far the first scene was used as the reference.

pdisdt_pwr, pdismph_pwr, pdis2dt_pwr, pdis2mph_pwr, vu_disp, dis_data, dis_ipta: The IPTA display programs for point data were migrated from GTK2 to GTK3. On both main and zoom windows, the clicking location is now marked by a white cross, while the location of the selected point is shown using a green cross.

vu_disp2d: This IPTA display program for 2D data stacks was also migrated from GTK2 to GTK3.

dis_ipta: Shift + right-clicking is now used to select a (new) reference point, while simple right-clicking is used to select a (new) point. The left-clicking location is marked by a white cross, while the location of the reference point is shown using a red cross and the location of the selected point is shown using a green cross.

pdata2gis: A new program to convert point data to vector file format for use in GIS software was added. The following output file formats are supported: GeoJSON, ESRI Shapefile, GeoPackage vector (GPKG), and Geography Markup Language (GML).

disp_tab2gis: A new program to convert displacement time-series text file (generated using either *disp_prt* or *disp_prt_2d*) to vector file format for use in GIS software was added. The following output file formats are available: GeoJSON, ESRI Shapefile, GeoPackage vector (GPKG), and Geography Markup Language (GML).

disp_prt: When point data layers are not desired or not available, they can now be skipped using a "-". In the output *disp_tab* file, the missing layers can either be filled with zeros or skipped, as selected using new option *[no_data_pol]*.

A new *[field_names]* option permits adding a first line containing field names, for simplified use as CSV file. Several options are provided for the date format in the displacement time-series field names.

disp_prt_2d: A new *[field_names]* option permits adding a first line containing field names, for simplified use as CSV file. Several options are provided for the date format in the displacement time-series field names.

prox_prt, prox_pt: The pixel spacing in both directions can now be specified using options [*xps*] and [*yps*], for more accurate distance sorting in case of different pixel spacings. Different points with the same coordinates are now all displayed using *prox_prt*.

fspf_pt: Memory is now allocated only for a single record at a time, reducing memory usage significantly when filtering all point data records.

disp_tab2gis, pdata2gis: The Mercator (1SP) and Pseudo-Mercator projections are now supported.

ts_rate: A new [DEM_par] option was added to enable support of data in map geometry.

Python wrapper

py_gamma.py, gamma_param_db.py, gamma_param.db: Using *py_gamma*, parameters of Gamma programs can now be entered using keyword arguments: the input parameter names are used as the keywords. With this method, unused options don't have to be entered. Keyword arguments can be entered in any order. A mix of positional and keyword arguments can also be used, with the positional arguments coming first.

The new function "*bash2python*" permits converting a bash command into the equivalent Python command using *py_gamma*. The output command uses per default positional arguments; by calling the function using the keyword argument "*use_kwargs = True*", the output command will use keyword arguments.

Matlab wrapper

-

All packages

The usage of all Gamma programs has been checked and, when necessary, updated to improve consistency of the parameter names in the usage and HTML-based documentation.

Optional parameters can now always be entered using a "-". A default value will be used in that case.