Release Notes GAMMA Software, 20260707

Urs Wegmüller, Christophe Magnard, Charles Werner, Othmar Frey,
Philipp Bernhard, Andreas Wiesmann
Gamma Remote Sensing AG
Worbstrasse 225, CH-3073 Gümligen
http://www.gamma-rs.ch
7-Jul-2026

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, bug fixes and new/updated demo examples.

Gamma Software on Linux, macOS, and Windows

The Gamma software has been compiled and tested on Linux (different distributions), Apple macOS Tahoe (26.5.2) with Apple Silicon processor, 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 24.04 LTS 64-bit Linux and is tested extensively with this distribution. The Gamma software is also available for Ubuntu 22.04 and 26.04 LTS.

Announcement: Support for Ubuntu 22.04 LTS will be available until the end-of-2026 upgrade.

Versions of the Software will also be uploaded for RHEL8 based on Rocky Linux 8, RHEL9 based on Rocky Linux 9, and RHEL10 based on Rocky Linux 10.

For installation instructions for the binary LINUX distributions see the HTML file INSTALL_linux.html (found in two places: the download directory of the distribution and the main directory of the distribution).

Apple MacOS Distribution:

The software in this version has been compiled using macOS Tahoe (26.5.2) with Apple Silicon processor. You will need to install libraries such as GDAL using **Homebrew** (no longer MacPorts!). The build uses GCC 16 compiler.

For installation instructions for the binary macOS distributions see the HTML file INSTALL_macOS.html (found in two places: the download directory of the distribution and the main directory of the distribution).

Windows Distribution:

The Windows version of the Gamma software is compiled with 64-bit support and multi-threaded. The build uses the MINGW64 GCC 16 compiler.

For installation instructions for the binary Windows distributions see the HTML file INSTALL_win64.html (found in two places: the download directory of the distribution and the main directory of the distribution).

The Gamma Software and its environment are now provided through two installers:

- Gamma Software Installer – Includes the Gamma Software. Its filename follows the pattern: GAMMA_SOFTWARE-YYYYMMDD_GEO_LAT.mingw64_msys2.exe
- Gamma Environment Installer – Provides shared libraries and support files for the Gamma Software, including MSYS2 (Linux-like environment), the current Gamma Local version, WinPython, 7-Zip, and Gnuplot. Its filename follows the pattern: GAMMA_ENV_YYYYMMDD.exe

Alternatively, the method previously used to install Gamma Software and Gamma Local is still available and is also documented in INSTALL_win64.html.

Notice that installing the latest Gamma Local 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.

The Gamma Software binaries as well as the Gamma Software and environment installers are now signed using EV (Extended Validation) code signing.

The Gamma Plugin for ArcGIS is available in all Gamma Software distributions for Windows that include the *GEO* or *ISP/DIFF&GEO* modules. Full functionality requires access to the *LAT* module.

On both Windows 10 and 11, it is also possible to install the Windows Subsystem for Linux (WSL2) 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. With Windows 10 reaching end-of-support in October 2025, the Windows distribution is now tested exclusively on Windows 11.

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 main Gamma documentation browser page *Gamma_documentation.html* is found in the main software directory.

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

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

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. Gamma Software program calls become Python / Matlab function calls where command line arguments can be used as function arguments, and system outputs can be stored in variables or written to log files. Binary images, point lists and data, parameter files, tab files, can be easily read, inspected, and written using additional functions provided with the wrappers.

The *py_gamma* Python module permits a smooth usage of the Gamma Software within Python scripts, Jupyter Notebooks as well as within a Python Interactive Development Environment (IDE) such as Visual Studio Code, Spyder or PyCharm. Using *py_gamma*, function arguments can be entered either as positional arguments or as keyword arguments, with the Gamma command line parameter names becoming the keyword names.

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.

Gamma plugin for ArcGIS

The Gamma plugin for ArcGIS permits using some Gamma software functionalities (tools) from ArcGIS Pro (Windows only) using a convenient interface. The Gamma plugin allows to perform the following operations:

- Reading SAR data from various sensors / formats
- Detection, radiometric calibration and geocoding of SAR data
- Co-registration of SLC and MLI SAR images in slant range / azimuth geometry
- Adaptive interferometric coherence estimation
- Multi-temporal processing and filters
- Spatial filtering of 2D SAR images
- Change detection in SAR images
- Polarimetric decompositions

Using ArcGIS ModelBuilder, it is possible to generate dedicated processing chains using the Gamma tools as building blocks. It is also possible to use each Gamma tool as a Python function (ArcGIS Pro is required). Note that the LAT module is required to be able to use all the tools; without the LAT module, only a subset of the tools is available. See also gamma-rs.ch/uploads/media/2024-1_Gamma_Plugin_for_ArcGIS_presentation.pdf.

Hardware Recommendations

Using multi-core processors (8 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 high-resolution data (TerraSAR-X, Cosmo-Skymed). 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 2 - 6 November 2026.

A PSI (IPTA) training at GAMMA (near Bern, Switzerland) is planned for 9 – 12 November 2026. See also https://www.gamma-rs.ch/software/training.

Significant Changes in the Gamma Software Modules since the End-of-2025 Release

Extended Omega-K algorithm & CPHD data support

The CPHD format is a generic format for SAR data [1]. Each CPHD data file includes a header, an XML block containing metadata, a PVP (Per Vector Parameters) block containing parameters for each data azimuth line, and a signal block containing the SAR data. CPHD data are range compressed data which have been motion-compensated relative to a Stabilization Reference Point (SRP). This SRP may be a single point (for spotlight data) or a moving point (for stripmap or sliding spotlight data). This format makes it relatively straightforward to compress the data using a Polar Format Algorithm (PFA). Alternatively, the motion compensation can be reversed to obtain a standard range compressed image. This is what the new *par_CPHD* reader does.

par_CPHD reads a CPHD data file and from the information included in the CPHD file, it generates the corresponding SAR sensor parameter file, elevation antenna pattern file, processing parameter file, and range-compressed data file. The XML metadata file contained in the CPHD data file can also be written out. As mentioned above, *par_CPHD* converts the data back to uncompensated data. The conversion includes an FFT in case the data are in frequency domain in the range direction, as well as a resampling and corresponding phase correction in range direction. An orbit state vectors / navigation data CSV file can be written out.

The new *prep_rc_data* program can be optionally used to prepare range-compressed data for azimuth compression. It includes features such as cropping an area of interest (AOI), applying an RFI filter, performing azimuth decimation or oversampling, setting the azimuth extension and bandwidth fraction, and updating the processing parameters accordingly.

The new *eok* program performs azimuth compression of range-compressed data. It is an implementation of the Extended Omega-K algorithm described in [2], with minor corrections described in [3]. This algorithm was modified from the conventional Omega-K algorithm [4] by separating the azimuth compression from the range cell migration (RCM) correction now achieved by a modified Stolt mapping, allowing a second-order, range-dependent, motion compensation.

eok expects range-compressed data as input, such as obtained from *par_CPHD*, *prep_rc_data*, or *pre_rc* / *pre_rc_JERS* / *pre_rc_RSAT*. *eok* supports focusing spaceborne data including data acquired in stripmap, spotlight or sliding spotlight modes, as well as airborne data (see images from various sensors focused using *eok* in Figure 4).

eok can be run with or without motion compensation. While focusing spaceborne stripmap data usually doesn't require motion compensation, it is necessary for airborne data due to deviations from a straight path. Motion compensation is also very effective for focusing spotlight data, which would otherwise require autofocus. Note that the motion compensation of spotlight data is based on the scene center indicated in the processing parameter file under "scene_center_latitude", "scene_center_longitude", and "terrain_height", which must therefore be accurate. The motion compensation uses a center-beam approximation; as a consequence, it will work best with narrow radar beams (in azimuth).

The algorithm includes the following steps (including motion compensation):

1) Track linearization using principal component analysis, which fits a line that minimizes the orthogonal distances from all the points from the flight or orbital path to the line.
2) Calculation of the range and azimuth offsets to linearize the data with respect to the linearized track. For each range and azimuth sample, the corresponding ground position (center of aperture) is determined. A plane is defined by the pointing vector from the sensor to this ground position and a vertical unit vector derived from the sensor’s Cartesian coordinates, passing through the sensor position (unit vector of the vector from the center of the Earth to

the sensor position). The intersection of this plane with the linearized track is then used to determine the offsets in the range and azimuth directions.

3) Range-independent motion compensation: For each azimuth line, a range offset is derived from the offsets calculated in (2). This offset is applied by interpolating the data in the range direction accordingly and applying the corresponding phase shift.
4) Azimuth filtering, oversampling in azimuth, and data interpolation for constant velocity:
   a) Without oversampling: The data are first deramped to center the spectrum. Then, an FFT is applied in the azimuth direction. Next, an azimuth bandpass filter is optionally applied to the data, followed by an IFFT in the azimuth direction. The data are then interpolated in azimuth direction using the offsets calculated in (2). Finally, the data are reramped to return the spectrum to its original state.
   b) With oversampling: The data are first deramped to center the spectrum. Then, FFTs are applied in both azimuth and range directions. The data are then oversampled in azimuth direction by zero-padding the azimuth spectrum. This also takes into account the range frequency-dependent Doppler centroid, which can be quite large in the case of strongly squinted data. The optional azimuth bandpass filter is also applied at this stage. Next, IFFTs are applied in both range and azimuth directions. The data are then interpolated in azimuth direction using the offsets calculated in (2). Finally, the data are reramped to return the spectrum to its original state. Azimuth oversampling ensures that there are no ambiguities in cases of large Doppler centroid variations, such as with spotlight data or some airborne data with turbulent flight paths.
5) Range oversampling: In case of large squint angles, the range spectrum may fold, making it difficult to resample or interpolate the data. Oversampling in range direction helps for such cases.
6) Modified Stolt mapping: The data are zero-padded in range and azimuth directions according to the synthetic aperture length and to the squint angles. A 2D FFT is applied. Optionally, a range bandpass filter can be applied at this point. The modified Stolt mapping is performed by interpolating the 2D-FFT transformed data as described in [2] and [3]. A 2D IFFT is performed and the excessive zero-padding is removed.

   The range cell migration has now been corrected and the data have been shifted in range direction to the final zero-Doppler range location. In azimuth direction, the data are still in Doppler centroid geometry.

7) Range-dependent motion compensation: For each range and azimuth pixel, a residual offset is calculated from the offsets determined in (2) and the compensation applied in (3). This offset is applied by interpolating the data in the range direction accordingly and applying the corresponding phase shift.
8) Range and elevation dependent radiometric correction: Correction of the antenna gain pattern in elevation and of the range spreading loss.
9) Azimuth focusing: The data are zero-padded in azimuth direction according to the synthetic aperture length and to the squint angles. An FFT is applied in the azimuth direction. The azimuth compression is performed as described in [2] and [3]. An IFFT is applied in the azimuth direction and the excessive zero-padding is removed.

   The data are now fully focused and are in standard zero-Doppler geometry in both range and azimuth directions.

When selected by the user, a range- and topography-dependent motion compensation is performed, using the central beam approximation: inside all motion compensation steps, the line of sight is computed either using a constant height relative to the WGS84 ellipsoid, or using a low-resolution DEM, i.e., with a resolution below the dimension of the synthetic aperture. For airborne

data, using a DEM can considerably improve the focusing quality as well as the radiometric correction using the antenna pattern.

A comma-separated CSV file with orbit state vectors or navigation data can be optionally provided (nav) to perform the motion compensation. This file is expected to have a line header containing the titles of the columns. It can have 4, 7, or 10 columns. The columns contain the following information:

- Column 1: Time in seconds of the day, synchronized with the information in the processing parameter file.
- Columns 2-4: Sensor position in ECEF X, Y, Z Cartesian coordinates
- Columns 5-7: Sensor velocity in ECEF X, Y, Z Cartesian coordinates
- Columns 8-10: roll, pitch, and heading

Motion compensation works best when the navigation data includes all 10 columns, resulting in the highest possible focusing quality.

Note that lever arm corrections should already have been applied to this file. When this file is provided, motion compensation will be performed by default. When this file is not provided, *eok* will use the state vectors from the processing parameter file to (optionally) perform a motion compensation.

Several options are provided for cropping the data, between tight crop that only keeps complete apertures up to no cropping, where the range and/or azimuth extensions are fully kept.

Additional options include manual setting of the oversampling factors in azimuth and range directions, range and azimuth bandpass filters, radiometric calibration variants.

Note that long spotlight acquisitions (or "dwell-mode" acquisitions) require large oversampling factors. As a consequence, their processing may require an excessive amount of memory (RAM). *eok* will provide an estimation of the required memory at run-time. It is recommended to monitor this memory use estimation and process only a subset of this type of data.

[1] National Geospatial-Intelligence Agency, Compensated Phase History Data (CPHD) Design & Implementation Description, Version 1.1.0, Standard NGA.STND.0068-1, 2021. Accessed: Apr. 20, 2026. [Online]. (https://nsgreg.nga.mil/doc/view?i=5388)

[2] A. Reigber, E. Alivizatos, A. Potsis and A. Moreira, "Extended wavenumber-domain synthetic aperture radar focusing with integrated motion compensation," IEE Proceedings Radar, Sonar and Navigation, vol. 153, issue 3, pp. 301-310, Jun. 2006.

[3] C. Magnard, M. Frioud, D. Small, T. Brehm, H. Essen, and E. Meier, "Processing of MEMPHIS Ka-Band multibaseline interferometric SAR data: From raw data to digital surface models," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 7, pp. 2927-2941, Jul. 2014.

[4] C. Cafforio, C. Prati and F. Rocca, "SAR data focusing using seismic migration techniques," IEEE Transactions on Aerospace and Electronic Systems, vol.27, no.2, pp. 194-207, Mar. 1991.

NISAR-L updates

An initial version of the NISAR data reader *par_NISAR_RSLC* was already included in the end-of-2025 release – but at the time it had only been tested with simulated NISAR acquisitions. Since then, extensive testing has been conducted and an additional reader for NISAR GSLC data – *par_NISAR_GSLC* – was added. Using actual data, we could assess data characteristics and investigate processing sequences for different applications such as DInSAR and polarimetric decomposition. In the following some of the findings are concisely reported. For more detailed

information we refer to the technical report "NISAR-L support within GAMMA Software" (available on the GAMMA website, see https://gamma-rs.ch/uploads/media/GAMMA_Software_references.pdf), and to the new NISAR demos (also mentioned below).

RFI mitigation: We observed radio-frequency interference (RFI) effects quite often in data over the USA, which were mainly obvious at cross-polarization. Spatially adaptive RFI filtering, as supported by the program *SLC_RFI_filt*, significantly reduced the RFI effects. A RFI filtering example has been documented in one of the new demo examples: *NISAR_demo* (see also Figure 5). In other areas, wide-band RFI was observed, probably generated by a "jammer" that could not be mitigated (e.g. near Ukraine).

DInSAR: NISAR-L differential interferometry (DINSAR) was assessed using both RSLC (normal SLC in slant-range – azimuth geometry) and GSLC (geocoded SLC in UTM map geometry) data. Both geometries are supported in the GAMMA Software and the results obtained look consistent. In the DInSAR testing it was obvious that many of the scenes were affected by ionospheric effects. Quite often the ionospheric effect was not very "wild" and could be mitigated. We carefully investigated the use of split-spectrum techniques for the mitigation of the ionospheric phase effects. Figure 1 shows a DInSAR processing sequence with ionosphere mitigation. We also generated a related new demo *NISAR_iono_demo* (see also Figure 6). For a more detailed discussion we refer to *U. Wegmüller, C. Werner, O. Frey, and C. Magnard, "Estimation and Compensation of the Ionospheric Path Delay Phase in PALSAR-3 and NISAR-L Interferograms," Atmosphere, vol. 15, no. 6, p. 632, May 2024, doi: 10.3390/atmos15060632*. Alternatively, the ionosphere related DInSAR phase can be mitigated together with the tropospheric phase effects.

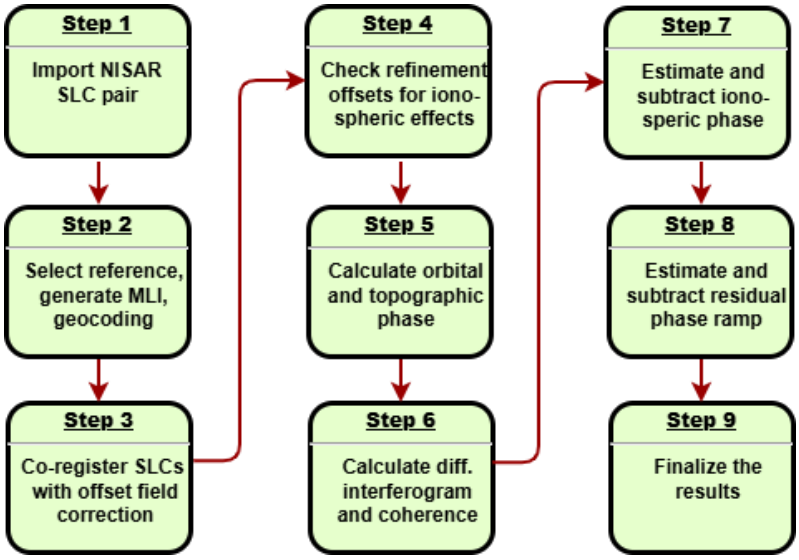


Figure 1 Flowchart of DInSAR processing sequence used.

In the estimation of the ionospheric phase screen, we typically work at a somewhat reduced spatial resolution (using higher multi-looking numbers) to be more robust and faster. For a data set with a 40 MHz bandwidth, we found it reasonable to use 48 range and 48 azimuth looks (similarly for a 20 MHz bandwidth data set it would be 24 range and 48 azimuth looks). Then after the estimation of the ionospheric path delay phase, for the actual InSAR analysis, we most likely want to work at higher resolution. Hence, we need to resample the estimated ionosphere phase to the higher resolution using *resamp_image_par*, before subtracting it from the higher resolution differential interferogram.

Quad-pol. acquisitions: For quality reasons the NISAR-L data are not acquired with the initially planned "variable PRF" mode but with a "constant PRF" mode. While having masked strips with no data can largely be avoided in single or dual-polarization data, this is not possible (so far) for quad-polarization acquisitions. As shown in Figure 2, both the RSLC and GSLC products have no-data areas where the signal was masked because its receive time window overlaps with the strong nadir return. Comparing the two products shows that the GSLC product includes a large fraction of no-data around the scene. The coverage of the scene has been somewhat cropped in the near and far range zones (so that it avoids the masked data area and the area in the far range where the dominant nadir return has not been masked (bright vertical stripe). Besides the data gap issue the polarimetric acquisitions are well suited for polarimetric decomposition (see also the new demo *NISAR_demo* and Figure 5).

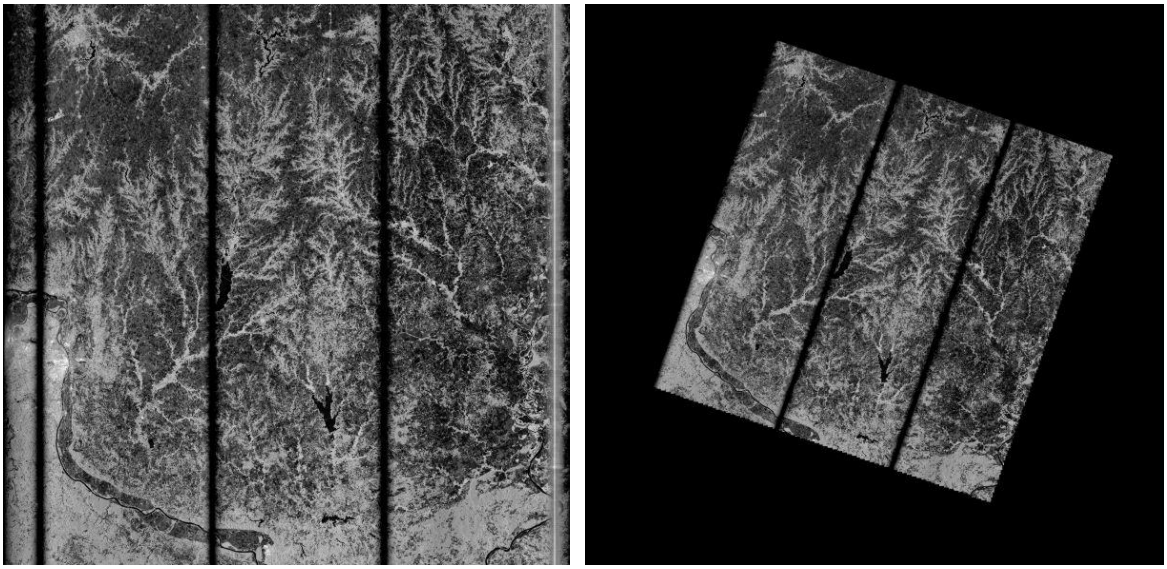


Figure 2 RSLC data (left) and corresponding GSLC data of a quad-pol. acquisition over St. Louis.

IPTA: We tested an interferometric time series analysis sequence using a small NISAR-L stack of 8 scenes over Mexico City. Using the IPTA software we tested a single reference stack processing. For this, we used the ionosphere and very wide-area scale tropospheric path delay estimates that were estimated from the stack of differential interferograms. The orbital tube was narrow during the acquisition of this sequence (see also the new demo *NISAR_ipta_demo* and Figure 7).

PALSAR-3 updates

For PALSAR-3 we also conducted further tests. The support of the data is consolidated. We also generated a new demo (PALSAR3_demo) that shows the reading of the data, data quality assessment, the mosaicking of multiple sub-swath SLCs into a single SLC, geocoding, co-registration and DInSAR. An important element was also the testing of cross-mode (e.g. ScanSAR – Stripmap) and cross-sensor (PALSAR-2 – PALSAR-3) interferometry. Furthermore, we successfully conducted an IPTA test using a small stack of 5 PALSAR-3 and 2 PALSAR2 SLC over Mexico City.

BIOMASS-P updates

After the end-of-2025 release, further tests were conducted using BIOMASS-P data acquired during the operational phase. The reader programs are working well. We assessed the data quality, RFI filtering, DInSAR, ionosphere mitigation and polarimetry. For further information on these investigations, we refer to the related technical report *GAMMA_TR_BIOMASS-P_Support_in_GAMMA* available on the GAMMA website. Some of the elements are also demonstrated in the demo example *BIOMASS_demo*. A poster on the ionosphere mitigation in interferometric BIOMASS-P pairs using split-spectrum interferometry was presented at Fringe'2026 in Krakow. The poster is also available on the GAMMA website.

Ionosphere mitigation

Ionospheric effects on SAR data are present at all wavelengths, but more pronounced at longer wavelengths. Early results of the BIOMASS P-band SAR mission confirmed its interferometric capability, but they also show that many of the interferograms are significantly affected by ionospheric effects. Relevant effects are also frequently observed in L-band data (PALSAR, NISAR-L). Because of the relevance of the new L- and P-band sensors for interferometry we carefully investigated the identification and mitigation of ionospheric effects. Although the related functionality was already available in the GAMMA Software, we updated some programs and added new programs to make the related processing more robust and easier to use.

The support provided in the GAMMA Software for the identification and mitigation of ionospheric effects on differential interferograms is summarized in the technical report *GAMMA_TR_Ionosphere* available on the GAMMA website. Furthermore, a related demo was generated using NISAR-L data with a separate narrow band (*NISAR_iono_demo*).

Spatial variations in the free electron concentration in the ionosphere affect SAR Interferometry in two ways:

1) Path delay gradients in the azimuth direction cause positional shifts.
2) The spatial variation of the ionospheric path delay affects the interferometric phase.

The ionosphere also causes Faraday rotation effects, which affects the cross-polarization backscatter and polarimetric parameters. The Faraday rotation depends on the total free electron concentration along the imaging path. The first two effects depend on the spatial variation of the free electron content in the ionosphere. The ionosphere mitigation support discussed here is for the identification and mitigation of the ionospheric effects on positional offsets and the interferometric phase.

In the investigation, we used data with a single spectral band as well as data with a separate narrow band, as available for some of the NISAR-L and PALSAR-3 acquisitions. The methodologies used in the two cases are very similar. In the case with just one spectral band, it is necessary to generate separate spectral bands by band-pass filtering (e.g. to get the lowest and highest third of the available spectrum). The principles used in the two cases are the same, but the processing slightly differs. In the NISAR-L case it is for example necessary to bring the data of the main band and the secondary band into the same geometry. An overall "flow-chart" for a processing sequence to identify and mitigate ionospheric effects is shown in Figure 3.

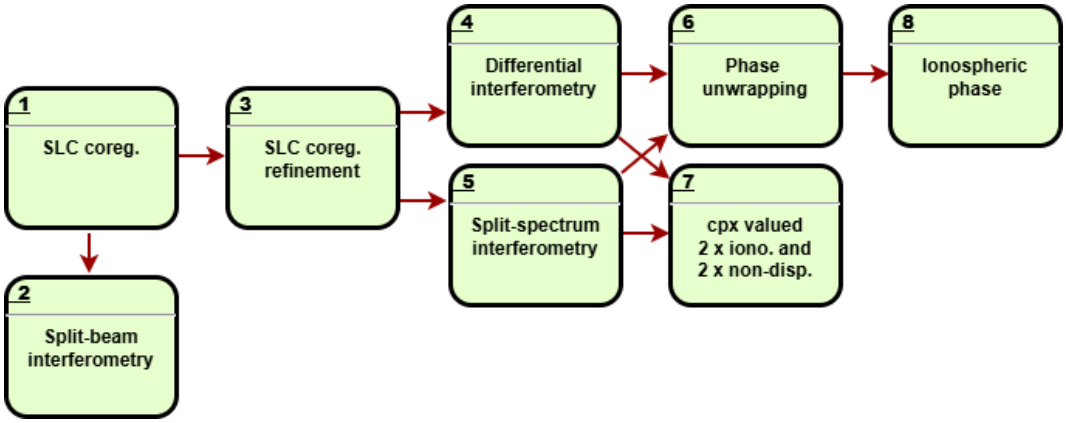


Figure 3 High-level float chart for ionosphere mitigation procedure.

We propose to make the SLC co-registration in two steps: first using *SLC_coreg.py* with only a constant offset refinement (i.e. using "*--npoly 1*"), and then, starting from the resampled secondary SLC obtained in step 1, using *SLC_coreg_refine.py*. The refinement offset field is visualized and used to identify ionospheric effects (spatially varying azimuth offsets are a clear indication for ionospheric effects or along-track motion; based on the spatial pattern and comparing it with the range offset field permits typically to decide if motion is present). The combined transformation lookup table can be derived based on the lookup table generated with *SLC_coreg.py* and the refinement offset field using the new program *lt_refine*.

Alternatively, ionospheric effects can be identified with split-beam interferometry (supported using *SBI_INT*).

In the case of data with a single spectral band, *SSI_INT* supports the generation of two separated bands.

Starting from the unwrapped split-spectrum interferogram phase and the unwrapped DInSAR phase of the main band, the new program *SSI_ionosphere.py* using mode 1 supports the calculation of the ionosphere (calculating the factors needed and applying these in the linear combination of the two phase images). *SSI_ionosphere.py* using mode 2 supports the calculation of twice the ionospheric phase and twice the non-dispersive phase starting from the unwrapped split-spectrum interferogram phase and the complex valued differential interferogram.

The ionosphere estimation is usually done using a quite strong multi-looking, which makes the processing more robust and efficient. Typically, getting an ionospheric phase screen at lower frequency is reasonable, especially if there are areas with intermediate or low coherence.

It is worth noting that there are cases with very strong spatial gradients in the ionospheric phase. In such cases the mitigation of the ionospheric effects may not be reliable.

New and updated SAR data readers

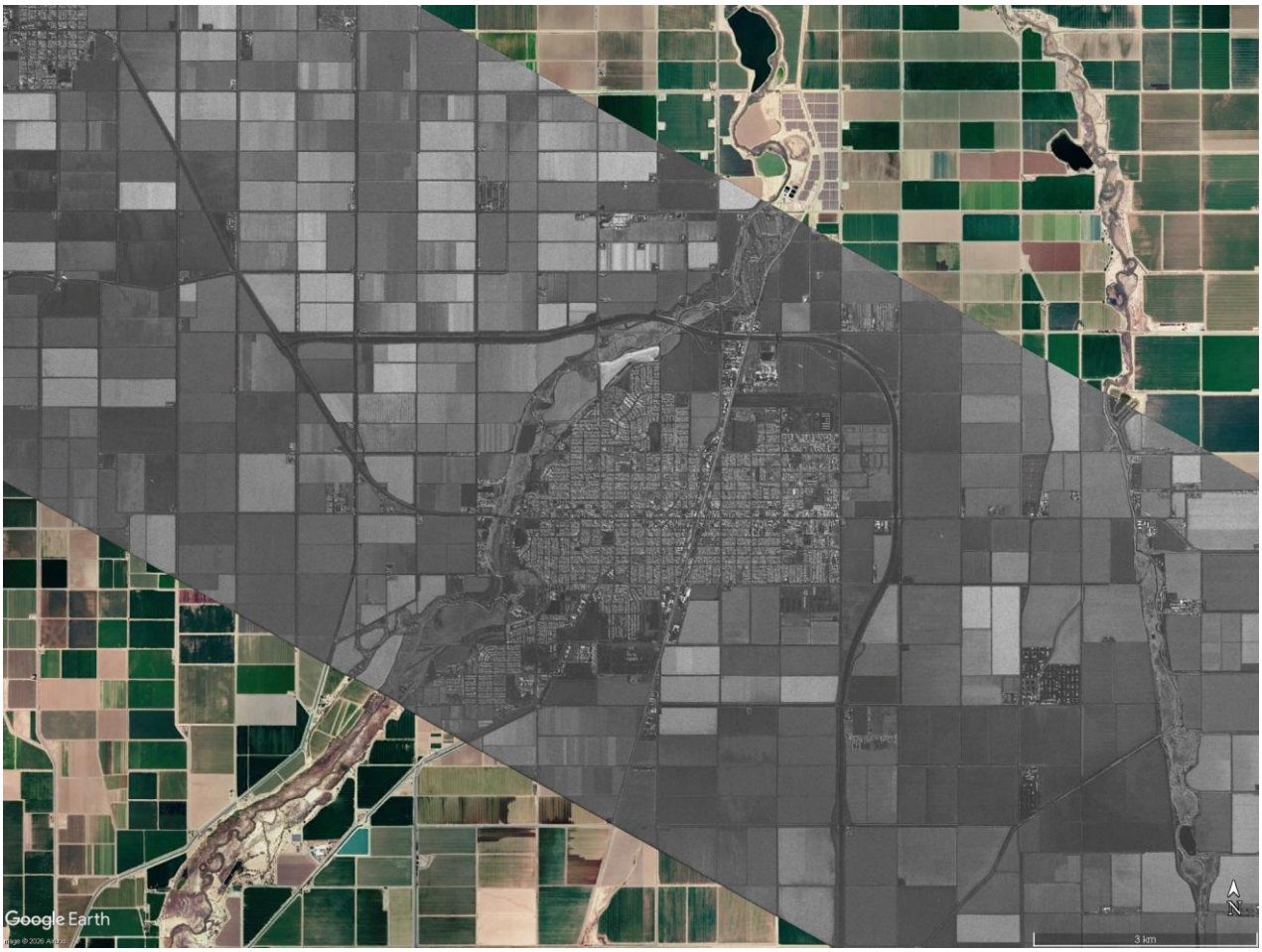
New / updated SAR data reader	Short description
par_CPHD	New program to generate MSP antenna elevation pattern, sensor and processing parameter files, and a range-compressed image file for CPHD data (Umbra, Capella).
par_ICEYE_SLC, par_ICEYE_SLC_HDF5, par_ICEYE_GRD	The legacy version of *par_ICEYE_SLC*, which was used to read ICEYE SLC data in HDF5 format, was renamed to *par_ICEYE_SLC_HDF5*. A new version of *par_ICEYE_SLC* has been implemented to support ICEYE SLC data in the new COG + JSON format. *par_ICEYE_GRD* now also supports the new COG + JSON format. RPC values are now used in priority to convert the data into slant range geometry. In case no RPCs are found, the program uses the ground range to slant range polynomial for the conversion. If none of the two methods is available, a simple model is used to convert the data from ground range to slant range. New [set_geom] option to manually define the input image geometry to zero-Doppler or squinted geometry. If no [set_geom] value is entered, the program will attempt to detect the input image geometry automatically. New [frame] option to add pixels around the area covered by the input image when converting the geometry from squinted to zero-Doppler geometry.
Sentinel-1 programs: Updates for ETAD data support and for Sentinel-1C and 1D: ***S1_ETAD_SLC, read_S1_TOPS_SLC.py, S1_path_number, S1_burstloc***	*S1_ETAD_SLC*, *read_S1_TOPS_SLC.py*: Corrections for ocean tidal loading displacements have been added and can be performed using option [ocean]. They are also included when performing all corrections. New [offsets] option to write the offsets to files in FCOMPLEX format, with the same dimensions as the SLC data. The real part is used for the offset in the slant range direction, and the imaginary part is used for the offset in the azimuth dimension. The offsets are scaled in SLC pixels. When using the --ETAD option (*read_S1_TOPS_SLC.py*) all timing corrections but no phase correction are performed. Added option --gpkg in *read_S1_TOPS_SLC.py* to show the coverage of the selected bursts as gpkg file. *S1_path_number* and *S1_burstloc* were updated for Sentinel-1C and Sentinel-1D.
par_NISAR_RSLC, par_NISAR_GSLC	*par_NISAR_GSLC*: New program to generate DEM parameter, SLC parameter and image files for NISAR Level-2 GSLC data. NISAR RSLC and GSLC data reading was tested using real NISAR-L data.

Gamma Software Demo examples

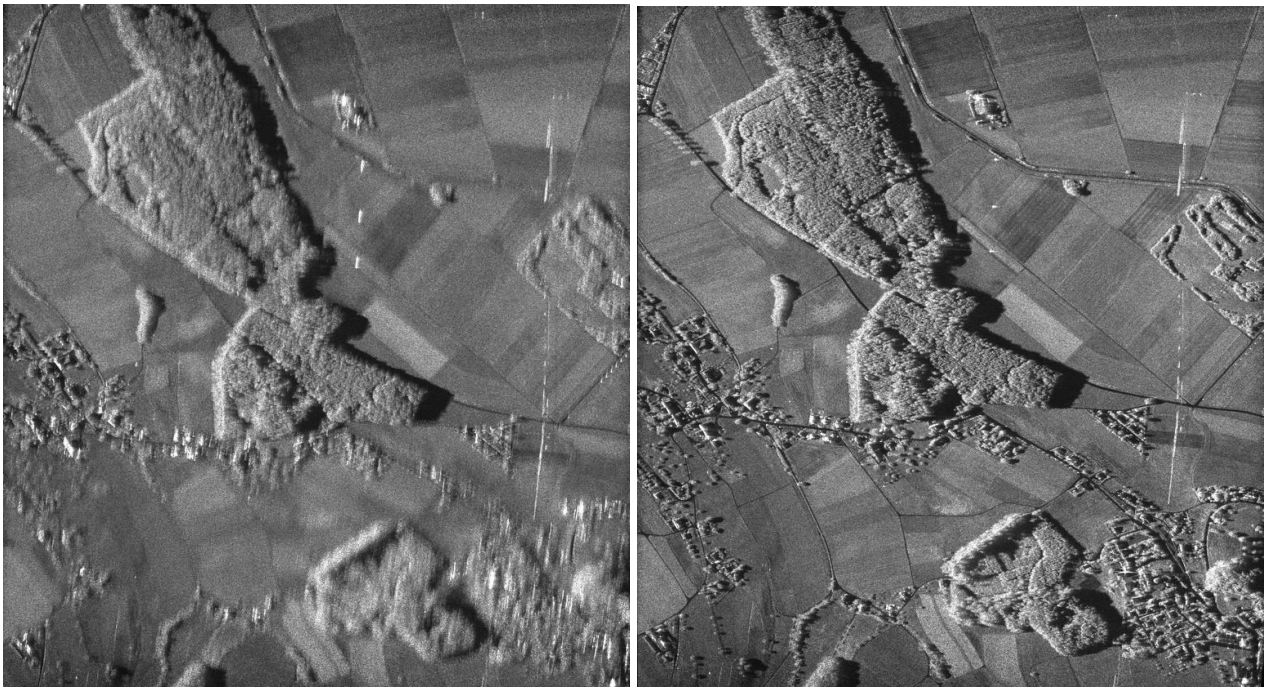
The access to the Gamma Software Demo examples is limited to Gamma Software users with a valid license. The access information is provided with the software delivery. A list of the Demo examples is available here:

https://gamma-rs.ch/uploads/media/README_Gamma_Software_demo.html.

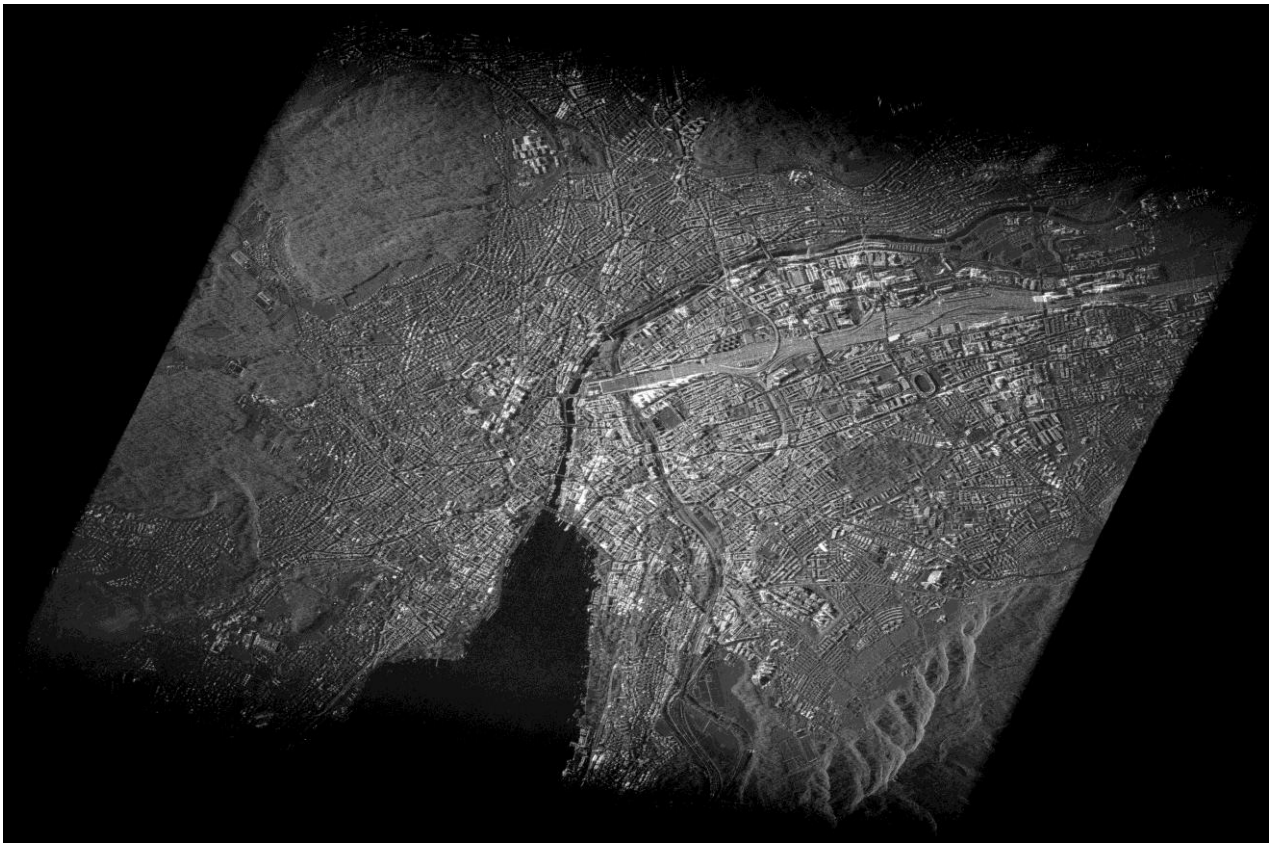
New / modified demo example:	**Contents**
Gamma_demo_MSP_CPHD_and_eok	The contents of the demo include: - Demonstrate reading of Compensated Phase History Data (CPHD) for a stripmap image from Capella Space and a squinted spotlight image from Umbra. - Demonstrate range-compressed data preparation using *prep_rc_data*. - Demonstrate azimuth compression of spaceborne stripmap and spotlight data as well as airborne data using the MSP program *eok*. The spaceborne spotlight data and the airborne data are focused using motion compensation. Using the COSMO-SkyMed interferometric pair, interferometric processing of the results of *eok* is demonstrated. The results are also compared to those obtained using *az_proc*.
NISAR_demo	The demo uses NISAR-L data over 3 sites, including both RSLC and GSLC products. For the first site a “co-seismic” pair over the Rift Valley is used to demonstrate NISAR data reading, geocoding, co-registration and DInSAR. For the second site (St. Louis) polarimetric decompositions, RFI filtering, NESZ estimation, geocoding, DInSAR, and the generation of a coherence product are shown. For the third site the identification and mitigation of narrow-band RFI (Radio Frequency Interference) effects in NISAR-L RSLC data is demonstrated.
NISAR_iono_demo	The demo shows the estimation of an ionospheric path delay phase screen using a NISAR-L interferometric RSLC pair with additional 5 MHz band images, using split-spectrum interferometry.
NISAR_ipta_demo	The demo conducts a “fast” PSI processing for a stack of 7 NISAR-L RSLC acquisitions over Mexico City.
PALSAR3_demo	The PALSAR-3 demo uses data freely available from the JAXA web-site. The demo includes reading the SLC data, assessing various parameters such as the Noise Equivalent Sigma Zero (NESZ), mosaic of sub-beam SLC segments into a single SLC, geocoding, co-registration and DInSAR.



Geocoded stripmap image acquired in January 2021 by the Capella-2 satellite over Imperial Valley, California, USA. Data provided in CPHD format and focused using *eok*. The CPHD file is from the Capella Open Data Collection, licensed under CC BY 4.0.

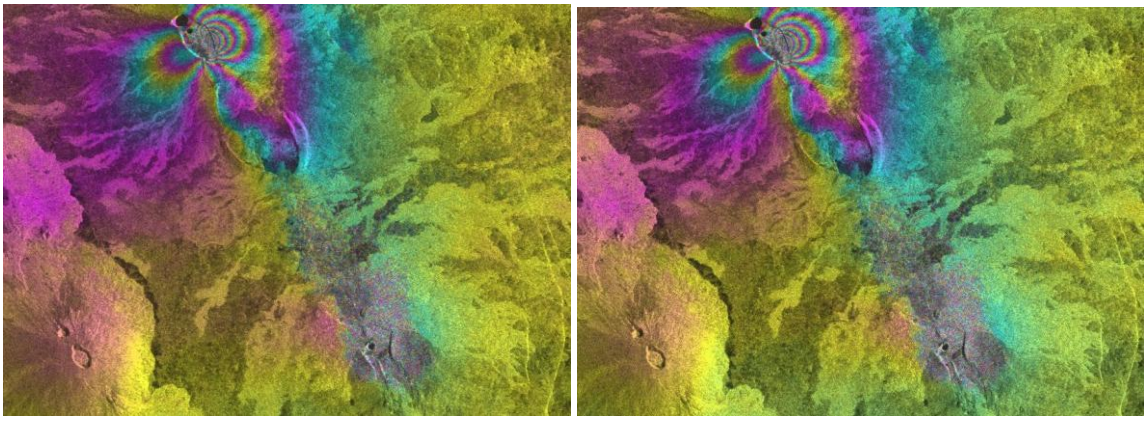


DOSAR C-band image acquired in 1994 over Kuettigkofen near Solothurn, Switzerland. Left: focusing using *eok* without motion compensation. Right: focusing using *eok* with motion compensation.

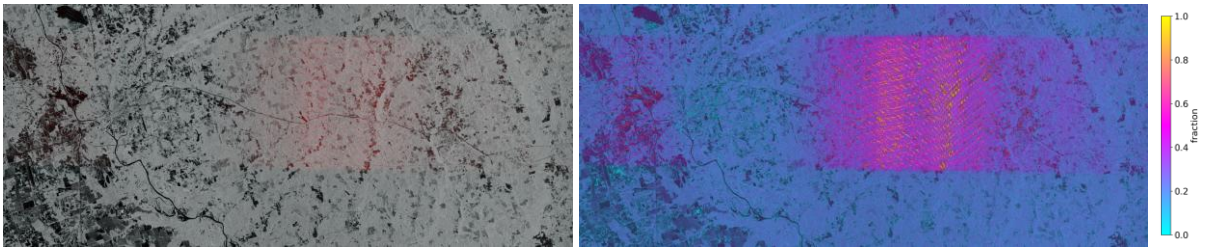


Squinted spotlight image acquired by the UMBRA-5 satellite in January 2024 over Zurich, Switzerland. The image is in zero-Doppler slant range – azimuth geometry! Data provided in CPHD format and focused using *eok*. The CPHD file is from the Umbra open data collection, licensed under CC BY 4.0.

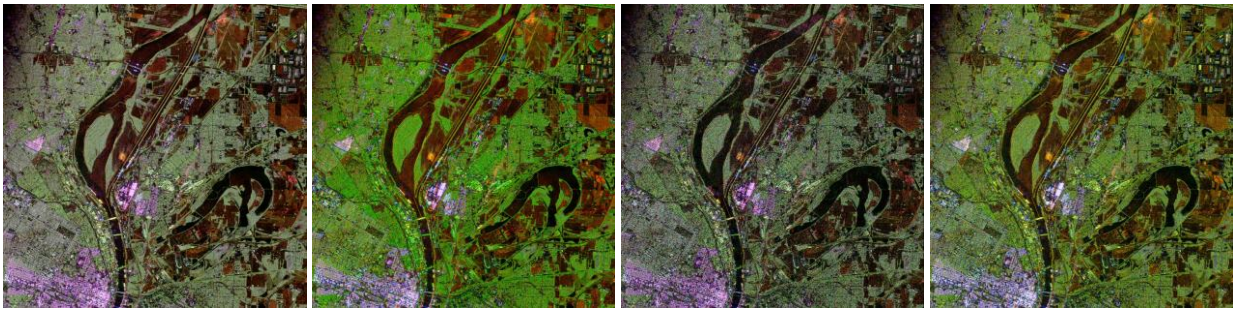
Figure 4 Some results generated in the MSP_CPHD_and_eok demo.



NISAR-L differential interferogram, 22-Nov-2025 to 4-Dec-2025 over Rift Valley, during a volcanic activity, starting from RSLC (left) and GSLC (right) products.



NISAR-L RFI effects observed in HV-pol. NISAR-L data of 20-Jan-2026 over the area east of St. Louis. The image to the left shows and RGB composite of the original HV-pol backscatter (with RFI, as red channel) and the corrected backscatter (without RFI, as green and blue channels). The image to the right shows the RFI fraction.



Pauli Freeman-Durden Cloude Yamaguchi

Polarimetric decomposition examples using quad-pol. NISAR-L data over St. Louis. In the RGBs shown the red channel corresponds to the dihedral, the green to the volume, and the blue to the surface scattering.

Figure 5 Some results generated in the NISAR demo.

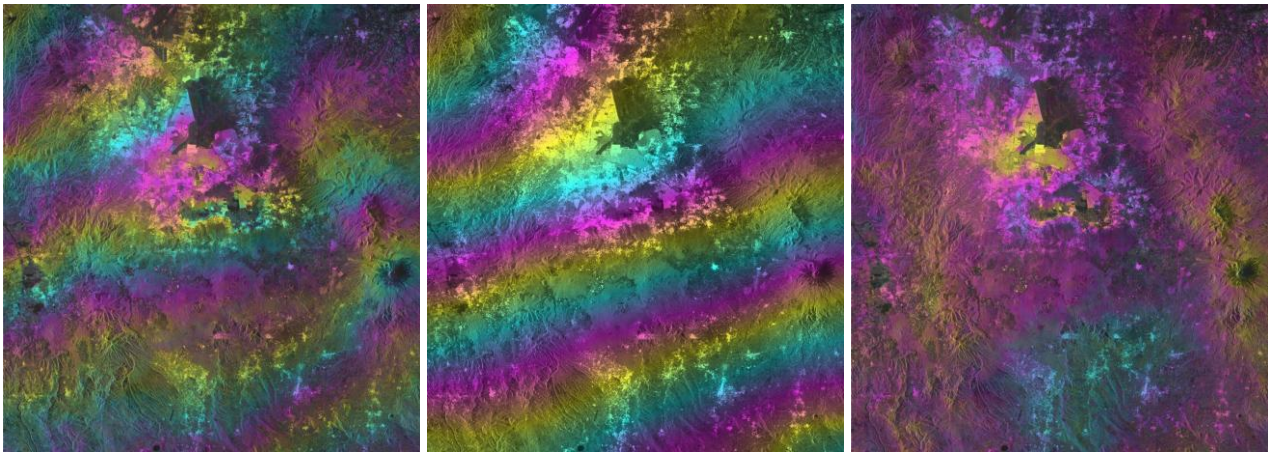


Figure 6 Some results generated in the NISAR ionosphere demo. The images show the differential interferogram (left), the estimated ionospheric phase screen (center), and the "ionosphere-corrected" differential interferogram (right), all using a cyclic color scale with one phase cycle per color cycle.

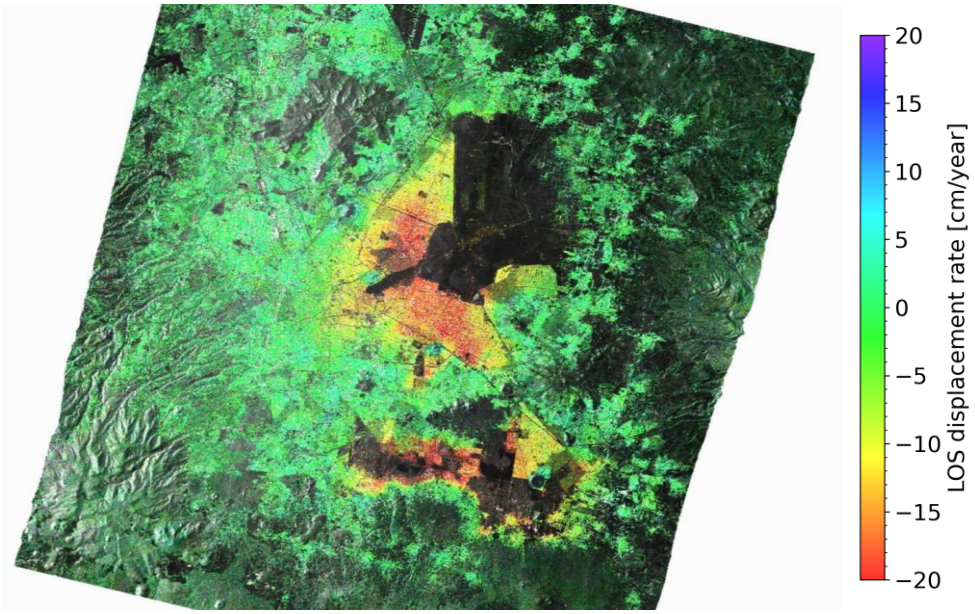


Figure 7 Average line-of-sight deformation rate of the period 24-Oct-2025 to 17-Jan-2026 determined from a stack of 7 NISAR-L acquisitions. For this small, temporally short stack, the displacement estimation uncertainty is around 1cm/year.

MSP

dop_mlcc: The program can now also be used with range-compressed data. New option to specify if the input data are raw data or range-compressed data.

doppler, doppler_2d: New [ofs] option to specify the number of offset samples from the edge to ignore. The value was previously hard-coded in the programs.

par_CPHD: New program to generate MSP antenna elevation pattern, sensor and processing parameter files, and a range-compressed image file from CPHD data.

prep_rc_data: New program to prepare range-compressed data for azimuth compression, including cropping of an AOI, RFI filter, azimuth decimation, azimuth oversampling, azimuth extension, and update of processing parameters.

eok: New program to perform azimuth compression using an Extended Omega-K algorithm. The program includes automatic azimuth extension and azimuth oversampling for spotlight data. Several cropping options are available to select the amount of data that is retained.

Options [nav] to use a navigation data file and [moco] to perform motion compensation. The motion compensation can be performed either based on the navigation data file or based on the state vectors in the processing parameter file.

Increased robustness when signal doesn't reach ground. 10% margin added Doppler centroid variations for the calculation of the toral azimuth bandwidth, azimuth oversampling now performed as soon as the estimated total azimuth bandwidth exceeds the PRF.

ISP

par_ICEYE_SLC, par_ICEYE_SLC_HDF5: The legacy version of *par_ICEYE_SLC*, which was used to read ICEYE SLC data in HDF5 format, was renamed to *par_ICEYE_SLC_HDF5*. A new version of *par_ICEYE_SLC* has been implemented to support ICEYE SLC data in the new COG + JSON format.

New [set_geom] option to manually define the input image geometry to zero-Doppler or squinted geometry. If no [set_geom] value is entered, the program will attempt to detect the input image geometry automatically.

New [frame] option to add pixels around the area covered by the input image when converting the geometry from squinted to zero-Doppler geometry.

par_ICEYE_GRD: New [set_geom] option to manually define the input image geometry to zero-Doppler or squinted geometry. If no [set_geom] value is entered, the program will attempt to detect the input image geometry automatically. Rational Polynomial Coefficients (RPCs) are used to change the geometry.

New [frame] option to add pixels around the area covered by the input image when converting the geometry from squinted to zero-Doppler geometry. In case no ground range to slant range polynomial is found, the program uses the RPCs for the conversion. If these are also not available, a simple model is used instead.

Now also supports the new COG + JSON format. RPC values are now used in priority to convert the data into slant range geometry. In case no RPCs are found, the program uses the ground range to slant range polynomial for the conversion. If none of the two methods is available, a simple model is used to convert the data from ground range to slant range.

mask_data: Processing now performed line by line to reduce memory usage and improve reliability with large data files under Windows.

ionosphere_mitigation_factors.py, LT1_precision_orbit.py, ORB_filt_spline.py, PALSAR_import_SLC_from_zipfile.py, plot_offsets.py, RCM_ORB_filt.py, read_S1_TOPS_SLC.py, S1_BURST_tab_from_zipfile.py, ScanSAR_BURST_tab_poly.py, ScanSAR_deramp_2nd.py, ScanSAR_deramp_reference.py, ScanSAR_ovr.py, SLC_cat_list.py, SLC_copy_PALSAR_ScanSAR.py, SV2_precision_orbit.py, TX_ScanSAR_import_SLC.py: Error messages now include the line number, date and time, and program name where the error occurred (when applicable). The error messages are now printed in the standard error stream (stderr).

OPOD_vec: Previously, a log file with a fixed name (OPOD_vec.log) was automatically generated. A new [log] option now lets the users choose whether to generate the log file and allows them to specify the file name.

par_SICD_SLC: The GDAL library is no longer used to read the SLC data because it cannot read data that are provided in multiple segments. Instead, the SLC image is now read directly using information from the NITF header and the segment subheaders.

ScanSAR_burst_corners: Added options for gpkg, geojson and shapefile output formats.

S1_ETAD_SLC: Corrections for ocean tidal loading displacements have been added and can be performed using option [ocean]. They are also included when performing all corrections.

The default value for the [phase] option has been changed to 0: phase corrections corresponding to the selected timing corrections in range are not applied anymore by default.

New [offsets] option to write the offsets to files in FCOMPLEX format, with the same dimensions as the SLC data. The real part is used for the offset in the slant range direction, and the imaginary part is used for the offset in the azimuth dimension. The offsets are scaled in SLC pixels.

read_S1_TOPS_SLC.py: When using the --ETAD option, all timing corrections are performed, but no phase correction is performed anymore.

New --gpkg option to show coverage of the selected bursts as GPKG file.

SLC_corners, ScanSAR_BURST_tab_poly.py: Added support for .gpkg, .shp, .geojson, .gml file formats.

PALSAR_import_SLC_from_zipfile.py, TX_ScanSAR_import_SLC.py: New --gpkg option to show coverage of the selected bursts as GPKG file.

ScanSAR_deramp_reference.py, ScanSAR_deramp_2nd.py, py_lib.py: Improvement of relative/absolute path support, *py_lib.py* file added to be used in python scripts.

S1_path_number, S1_burstloc: Update for Sentinel-1C and Sentinel-1D.

base_orbit: Now supports much larger baselines. Improved accuracy in baseline calculation.

par_BIOMASS_SLC, par_BIOMASS_GRD: Update to support latest data format changes (Doppler centroid polynomial).

par_NISAR_RSLC: Update to support data subsets obtained e.g. using *openSEPPO*. Avoid writing nan data. Reading float vectors is now more robust (also supports double, in case).

SLC_RFI_filt: As an alternative to using the name of the SLC parameter file, the width of the SLC can now be entered in the <SLC_par> command line argument. This may be useful for filtering geocoded images.

DIFF&GEO

base_calc, base_plot: New [date_format] option to set the date format to either the legacy YYYYMMDD format or to the ISO 8601 YYYYMMDDTHHMMSS format. The ISO 8601 date format is now the default. The bperp_file now has the same number of columns and the same format for single- and multi-reference itab types. The plot has been updated to utilize the increased precision of the ISO 8601 date format.

Previously, a log file with a fixed name (base_calc.log, base_plot.log) was automatically generated. A new [log] option now lets the users choose whether to generate the log file and allows them to specify the file name.

When the bperp text file name ends with ".txt" or a few other similar extensions, the bperp plot file name now replaces that extension with ".png" rather than simply concatenating it.

CARD4SAR.py, crop_image.py, gc_map2_large.py, geocoding.py, gp_geo_fit.py, MLI_coreg.py, MLI_coreg_refine.py, radcal.py, ScanSAR_coreg.py, ScanSAR_coreg_check.py, ScanSAR_coreg_overlap.py, ScanSAR_coreg_pol.py, ScanSAR_coreg_stack.py, SLC_coreg.py, SLC_coreg_refine.py, stack_cpx.py: Error messages now include the line number, date and time, and program name where the error occurred (when applicable). The error messages are now printed in the standard error stream (stderr).

ionosphere_mitigation, ionosphere_mitigation_S1, SSI_int, SSI_int_S1: Improved robustness in case of small images / very large multi-looking.

SLC_coreg_refine.py, MLI_coreg_refine.py: The new --extension option slightly increases the coverage of the offset estimates at the right and bottom edges of the image when the --rstep and --azstep options are both used. If the --extension option is not selected when using both --rstep and --azstep options, the resulting offset field has the same dimensions as a multi-looked image generated using the same number of looks as the step sizes.

par_NISAR_GSLC: New program to generate DEM parameter, SLC parameter and image files for NISAR Level-2 GSLC data.

ScanSAR_coreg.py, py_lib.py: Improved path handling in *ScanSAR_coreg.py*, support of relative and absolute paths with os.path, file read/write using context managers, *py_lib.py* file added to be used in python scripts.

SLC_coreg.py, MLI_coreg.py: New --poly and --mask options to select the area used for the refinement procedure.

lt_refine: New script to refine a co-registration lookup table with an offset field in the same geometry (the inputs can be obtained using *SLC_coreg.py* followed by *SLC_coreg_refine.py*). The combined co-registration lookup table is used in the calculation of the split-spectrum double difference interferogram with the program *SSI_INT*.

SSI_ionosphere.py: *SSI_ionosphere.py* replaces *SSI_ionosphere*. It is a more generic version that supports the estimation of the ionospheric path delay phase screen using split-spectrum interferometry. It uses as inputs the three SLC parameter files that define the carrier frequency of the main spectral band (corresponds to the full bandwidth band in the case of data with a single spectral band), the lower spectral band (is identical with the main spectral band in the case of PALSAR-3 and NISAR-L data with an additional separate narrow spectral band), and the upper spectral band and the complex valued and unwrapped full bandwidth differential interferogram and the unwrapped split-spectrum double-difference interferogram phase. Based on the carrier frequencies, the program calculates the factors for the linear combination of the phases used in the ionosphere phase screen estimation (as can also be done separately using the program

ionosphere_mitigation_factors.py) and applies these in the calculation of the ionosphere phase screen. With mode 1 it calculates the (unwrapped) ionospheric phase screen, as well as the unwrapped non-dispersive (or "ionosphere corrected") phase. With mode 2 twice the ionospheric phase and twice the non-dispersive phase are calculated – here the unwrapped main band differential interferogram is not required. In the ionosphere mitigation, using a higher multi-look factors makes the procedure (in particular the unwrapping) more robust and more efficient. Consequently, the spatial resolution of the estimated ionospheric phase screen has a somewhat reduced spatial resolution – which seems in fact reasonable.

map_copy: New program to copy a segment from a geocoded data file.

mk_kml: Updated KML namespace from http://earth.google.com/kml/2.1 to OGC standard http://www.opengis.net/kml/2.2 . Increased coordinate precision in LatLonBox from 6 to 10 decimal places. Fixed variable name conflict between KML output filename and KML string.

geocoding.py: If only one of the --lat_ovr or --lon_ovr options is entered, the program will automatically calculate the other oversampling factor to produce square pixels (in meters) at the center of the image.

LAT

compact_pol_decomposition.py, quad_pol_decomposition.py: Error messages now include the line number, date and time, and program name where the error occurred (when applicable). The error messages are now printed in the standard error stream (stderr).

frame: Double and triple values for options [null_value] and [frame_value] should now be entered within double quotes, when the input image is in FCOMPLEX format or is a raster image. *frame* should still support the previous format, where each element of the double and triple values was entered as an individual parameter.

sp_cc: New program to calculate the spectral correlation of an SLC. It corresponds to the core functionality of *sp_stat* in the IPTA module. A high spectral correlation indicates a "point-like" scatter characteristic.

DISP

thres_data: Processing now performed line by line to reduce memory usage and improve reliability with large data files under Windows.

ColormapText.py, ras2png.py, vis_colormap_bar.py, visbyte.py, viscpx.py, visdt_pwr.py, vismph_pwr.py, vispwr.py, visras.py: Error messages now include the line number, date and time, and program name where the error occurred (when applicable). The error messages are now printed in the standard error stream (stderr).

dis_dB, dis_linear, discpx, disflag, disgbyte, dismph, dismph_fft, dispwr, disshd, disSLC: The lower right panel of the display window now includes a color bar.

ras programs*: New [xstart] and [nx] options to select area of interest in horizontal direction.

dis2 programs*: Added support for color bar display.

vis programs*: Added support for BMP/TIFF/SUN raster background images. [width] option can now also be a parameter file, the program will try to automatically detect the corresponding parameter file when '-' is entered. min/max and option -c support "pi" as argument. Added missing parameters -r and -n in *vismph_pwr.py*

ras_linear, rascpx, dis2_linear, dis2dt_pwr, discpx, disdt_pwr: min/max support "pi" as argument

dis_linear, gdsp: Added support for signed 2-byte little-endian data type (D_SHORT_LE) in *dis_linear* display handling and *gdsp* cursor value readout.

data2geotiff: New [tfwflg] option for world file sidecar and make <data>.tif default output filename.

disdt_pwr, dismph_pwr, distree, dis2dt_pwr: Added support for colormap display.

viscpx.py, visdt_pwr.py, vispwr.py, vismph_pwr.py, vis_colormap_bar.py: Added support for pi symbol in colormap legends of vis*.py scripts when min/max are set to pi-multiples (e.g. -pi, pi/2), colorbar tick labels now render as pi expressions.

kml_map: Updated KML namespace from http://earth.google.com/kml/2.1 to OGC standard http://www.opengis.net/kml/2.2. Increased coordinate precision in LatLonBox and terminal output from 7 to 10 decimal places. The program now works correctly when the KML filename argument is not provided on the command line. In that case, the KML file name is constructed from the input image name.

rasmph_pwr, rasdt_pwr, rasmph, raspwr: Correction of memory leaks and excessive memory allocation.

IPTA

kml_plist.py, unw_correction_mb_pt.py: Error messages now include the line number, date and time, and program name where the error occurred (when applicable). The error messages are now printed in the standard error stream (stderr).

sp_stat, mk_sp_all: New option [bw_auto] for automatic bandwidth fraction detection, enabled by default. New options [rbwf] and [azbwf] to manually set the SLC range and azimuth bandwidth fractions. Notes: when the bandwidth fraction is automatically detected, options [r_ovr], [az_ovr], [rbwf] and [azbwf] are ignored. When options [rbwf] and/or [azbwf] are specified, [r_ovr] and/or [az_ovr] are ignored, respectively. New option [ref_ph] to select the reference for the phase difference calculation between the looks, either to the SLC data (default), or to a look located around the center of the spectrum. *mk_sp_all* should now also work with GeoSLCs.

data2pt, d2pt: Correction of a memory allocation error when reading SCOMPLEX data.

pt2data, pt2d, pt2data_inpaint, pt2d_inpaint: Correction of the position of the output pixels when multi-looking. The position error was up to 1 pixel in each direction. There should be no change when no multi-looking is used. The programs now work consistently with *data2pt* and *d2pt*.

fspf_unw_pt: Parallel computation of data records is now possible when multiple records are being processed. It is enabled by setting the new option [parallel_flg] to 1. The allocated memory is proportional to the number of threads. In case of excessive memory usage, either set [parallel_flg] to 0 or reduce the number of threads using the environment variable OMP_NUM_THREADS.

pdisdt_pwr, pdismph_pwr, pdis2dt_pwr, pdis2mph_pwr, prasdt_pwr, ras_data_pt: Added colorbar display to the GTK3 interactive point display programs (pdis*): colormap gradient strip with tick marks and labels is shown alongside the image. For FLOAT data programs (*pdisdt_pwr, pdis2dt_pwr, prasdt_pwr, ras_data_pt*) the min and max arguments now accept pi-expressions (e.g. -pi, pi/2, 3pi).

TS_DISP

None

TDBP

The TDBP programs are available for Ubuntu 24.04/22.04 Linux and Windows 11 WSL2 with Ubuntu 24.04/22.04 Linux (and legacy 20.04 Ubuntu Linux and Windows 11 WSL2 versions).

The time-domain back-projection focusing program *az_proc_tdbp_gpu* employs parallelized TDBP imaging implemented in C/CUDA and therefore requires an Ubuntu Linux computer equipped with an NVIDIA GPU.

The TDBP module supports image focusing of SAR data from airborne platforms, UAVs, and mobile-mapping platforms such as cars, or trains, with high-quality 3-D geometry/motion compensation. Subsequent interferometric/tomographic processing and value-adding is supported. The TDBP module supports SAR image focusing of pulsed and FMCW SAR data with a generic sensor-agnostic data interface. In particular, SAR image focusing of FMCW SAR data acquired with our in-house Gamma SAR systems (https://www.gamma-rs.ch/instruments) is also supported. See also the Gamma L-band SAR demo examples and our publications on ground motion/slope stability retrieval plus other SAR imaging examples obtained with Gamma SAR systems mounted on a car or a UAV and processed with the TDBP module: https://www.gamma-rs.ch/instruments/sar-system-publications.

GIS

arc_read_data.py: Support for the new GeoTIFF (COG) + JSON format of ICEYE GRD and SLC data has been added.

Python wrapper

New function *eprint* to print error messages to stderr → Python scripts now print error messages to stderr.