Introduction to ISCE3

Geoffrey Gunter

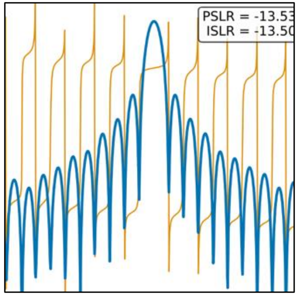
Aug 21, 2025

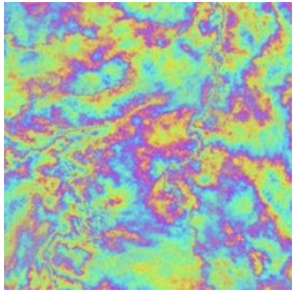
Jet Propulsion Laboratory
California Institute of Technology

This document has been reviewed and determined not to contain export controlled technical data

What is ISCE3?

- “InSAR Scientific Computing Environment: Enhanced Edition”
- Open-source library for SAR processing & analysis
- Supports image formation, calibration, RFI mitigation, co-registration, geocoding, RTC, interferogram formation, phase unwrapping, …
- Python API w/ performant C++/CUDA backend





Why ISCE3?

- Ground-up redesign of ISCE2
- Modular, extensible structure to support future SAR missions and advanced algorithms
- Improve documentation & testing
- Increase support for hardware acceleration using GPUs

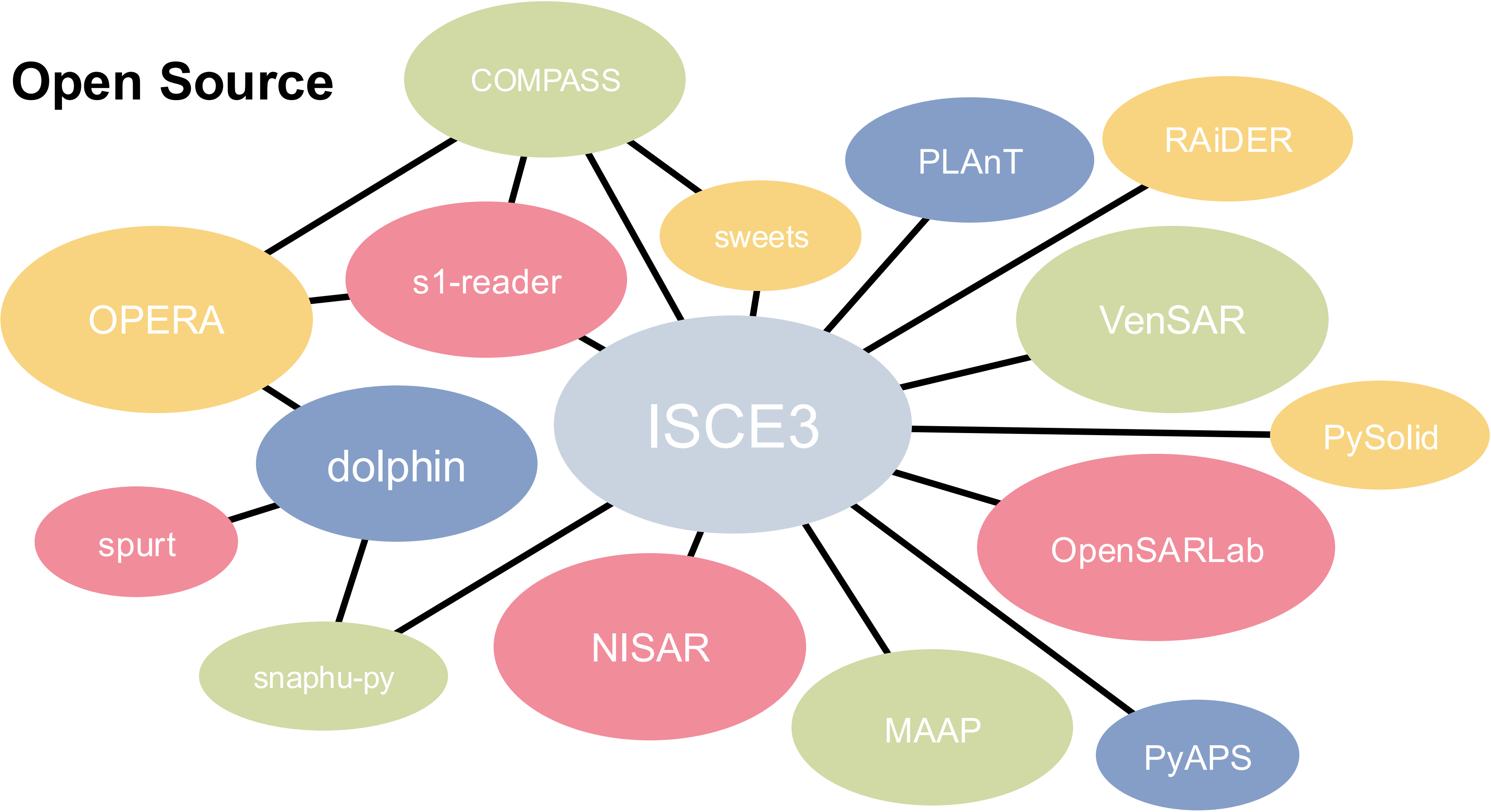
This document has been reviewed and determined not to contain export controlled technical data

Open Source

- NASA Open Science initiative (https://science.nasa.gov/open-science/)
- Engage with the community through workshops, discussion forums, open-source contributions, etc.
- Develop a rich ecosystem of libraries & tools to support:
  - NASA airborne/spaceborne SAR missions
  - Commercial & international partners
  - Science/applications community

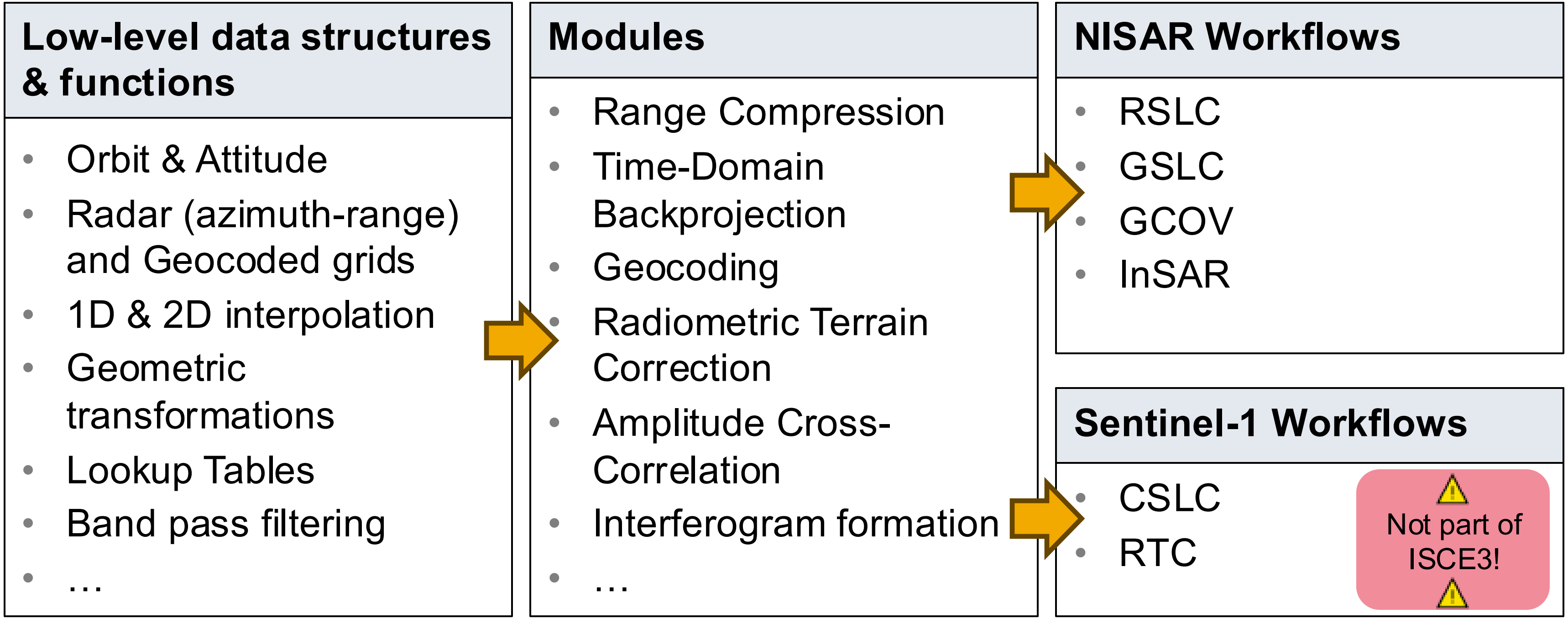
This document has been reviewed and determined not to contain export controlled technical data

Open Source



This document has been reviewed and determined not to contain export controlled technical data

Modular, Extensible Structure



Low-level data structures & functions

- Orbit & Attitude
- Radar (azimuth-range) and Geocoded grids
- 1D & 2D interpolation
- Geometric transformations
- Lookup Tables
- Band pass filtering
- …

Modules

- Range Compression
- Time-Domain Backprojection
- Geocoding
- Radiometric Terrain Correction
- Amplitude Cross-Correlation
- Interferogram formation
- …

NISAR Workflows

- RSLC
- GSLC
- GCOV
- InSAR

Sentinel-1 Workflows

- CSLC
- RTC

⚠ Not part of ISCE3! ⚠

Example: Compute Line of Sight Vector

```python
import isce3
import nisar
import numpy as np

rslc = nisar.products.readers.RSLC(hdf5file="...")
freq = "A" if ("A" in rslc.frequencies) else "B"
orbit = rslc.getOrbit()
doppler = rslc.getDopplerCentroid(freq)
radar_grid = rslc.getRadarGrid(freq)

lon, lat, height = ...
ellipsoid = isce3.core.WGS84_ELLIPSOID
target_pos_ecef = ellipsoid.lon_lat_to_xyz([lon, lat, height])

aztime, _ = isce3.geometry.geo2rdr_bracket(
    xyz=target_pos_ecef, orbit=orbit, doppler=doppler,
    wavelength=radar_grid.wavelength, side=radar_grid.lookside,
)
platform_pos_ecef, _ = orbit.interpolate(aztime)

def normalize(vec):
    return np.asarray(vec) / np.linalg.norm(vec)
los_unit_vec = normalize(platform_pos_ecef - target_pos_ecef)
```

Installation

- Install from conda-forge (recommended)

```
$ conda install -c conda-forge isce3
```

or

```
$ conda install -c conda-forge isce3-cpu
```

or

```
$ conda install -c conda-forge isce3-cuda
```

This document has been reviewed and determined not to contain export controlled technical data

Installation

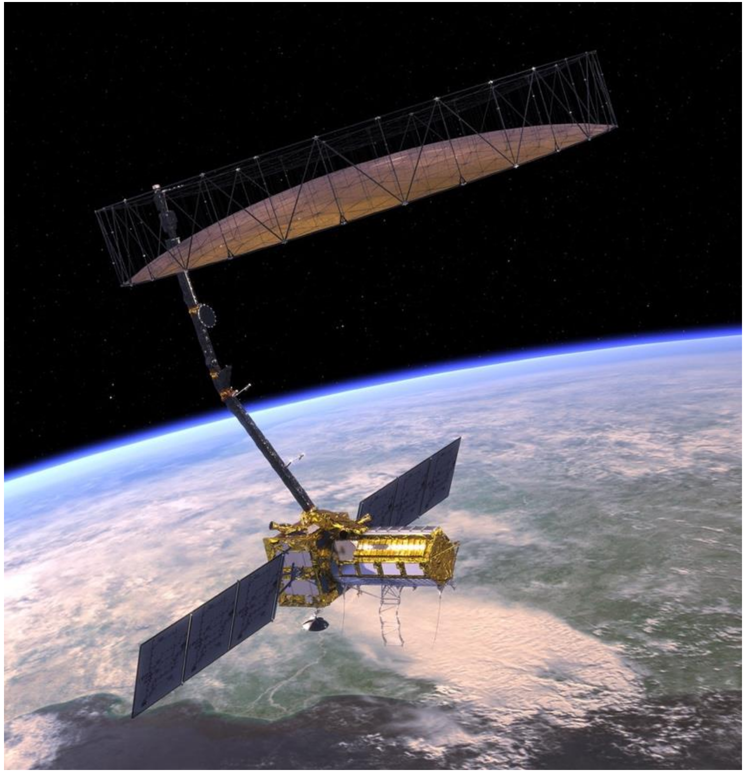
- Install from source with pip

```
$ git clone https://github.com/isce-framework/isce3.git
$ cd isce3
$ conda env create –f environment.yml
$ conda activate isce3
$ pip install .
```

- Install from source with CMake (advanced)
  - https://isce-framework.github.io/isce3/buildinstall/#building-with-cmake-advanced

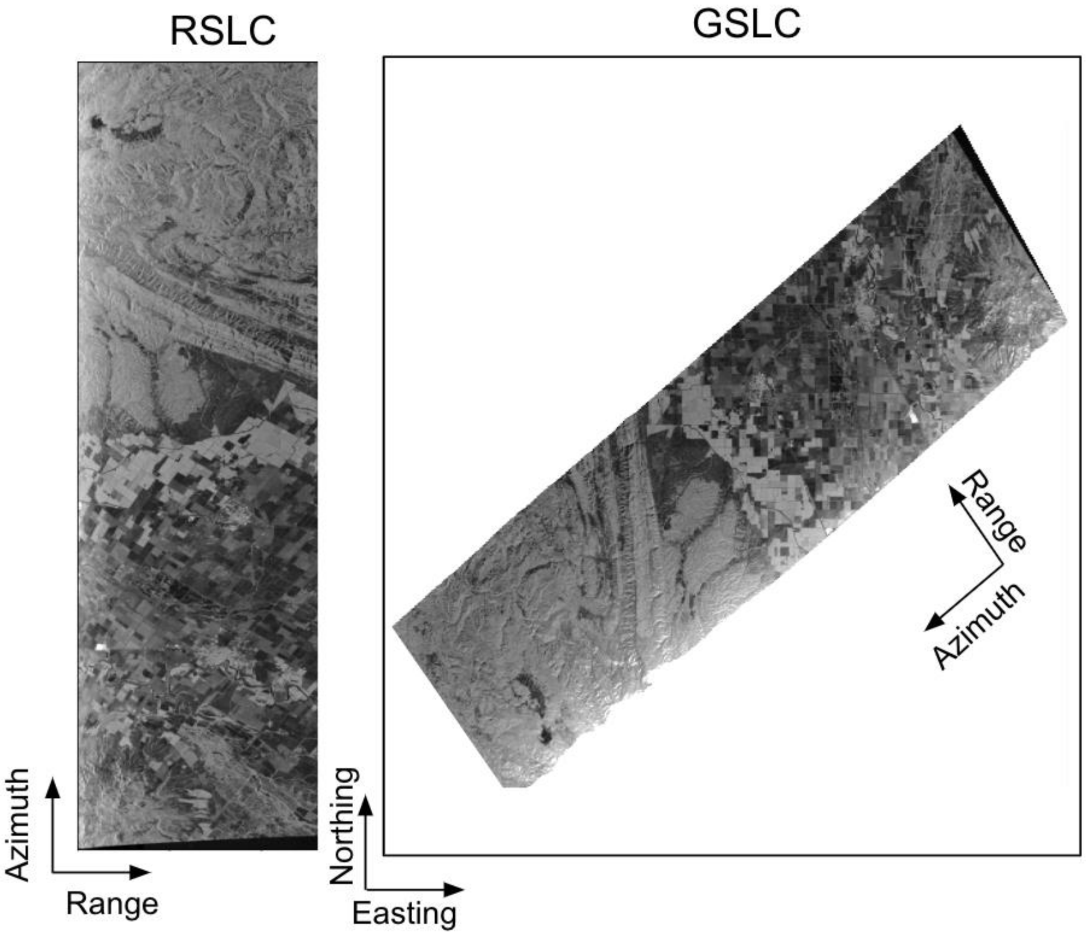
NISAR ❤ ISCE3

- ISCE3 development has been largely driven by NISAR since conception
- Core development team are NISAR ADT
- NISAR L1/L2 science products are generated using ISCE3
- ISCE3 includes tools for reading NISAR HDF5 products



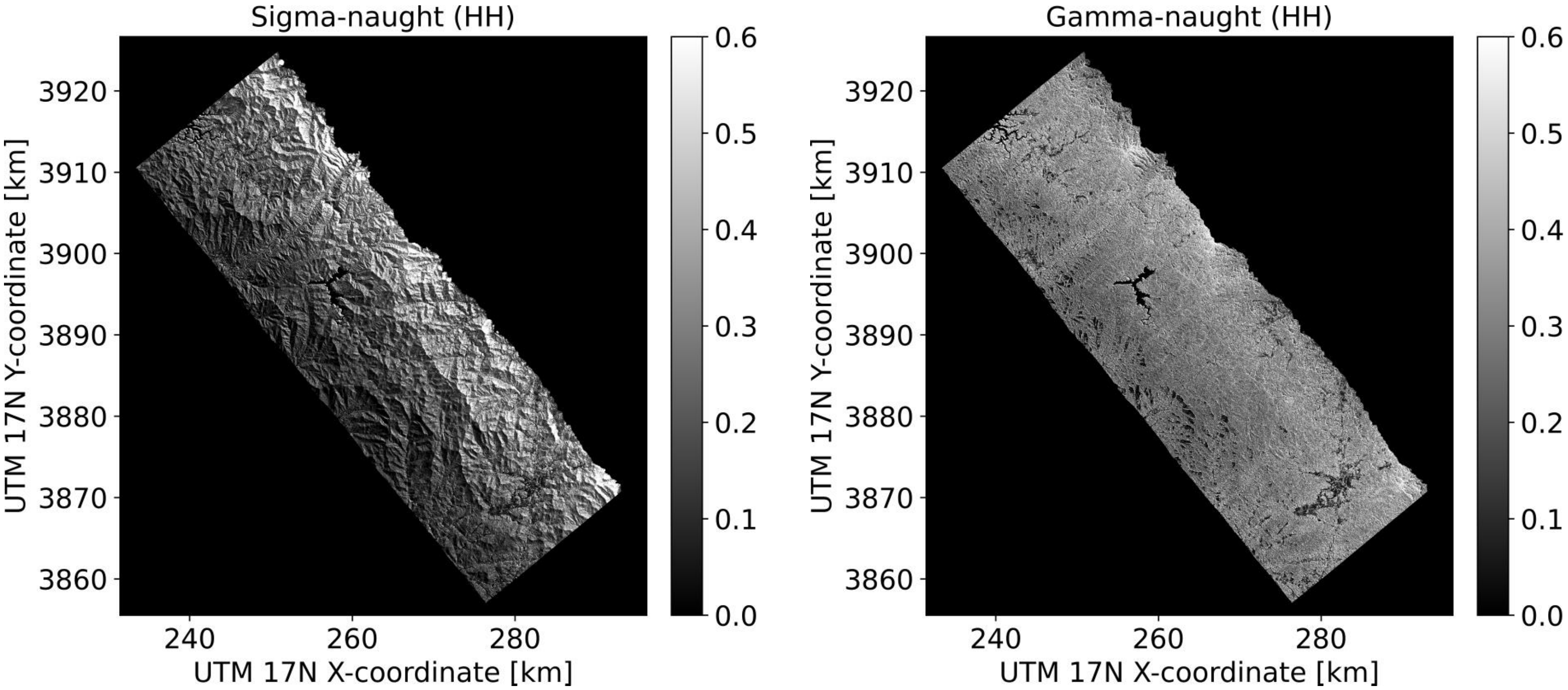
This document has been reviewed and determined not to contain export controlled technical data

NISAR Range-Doppler (RSLC) & Geocoded (GSLC) Single-Look Complex



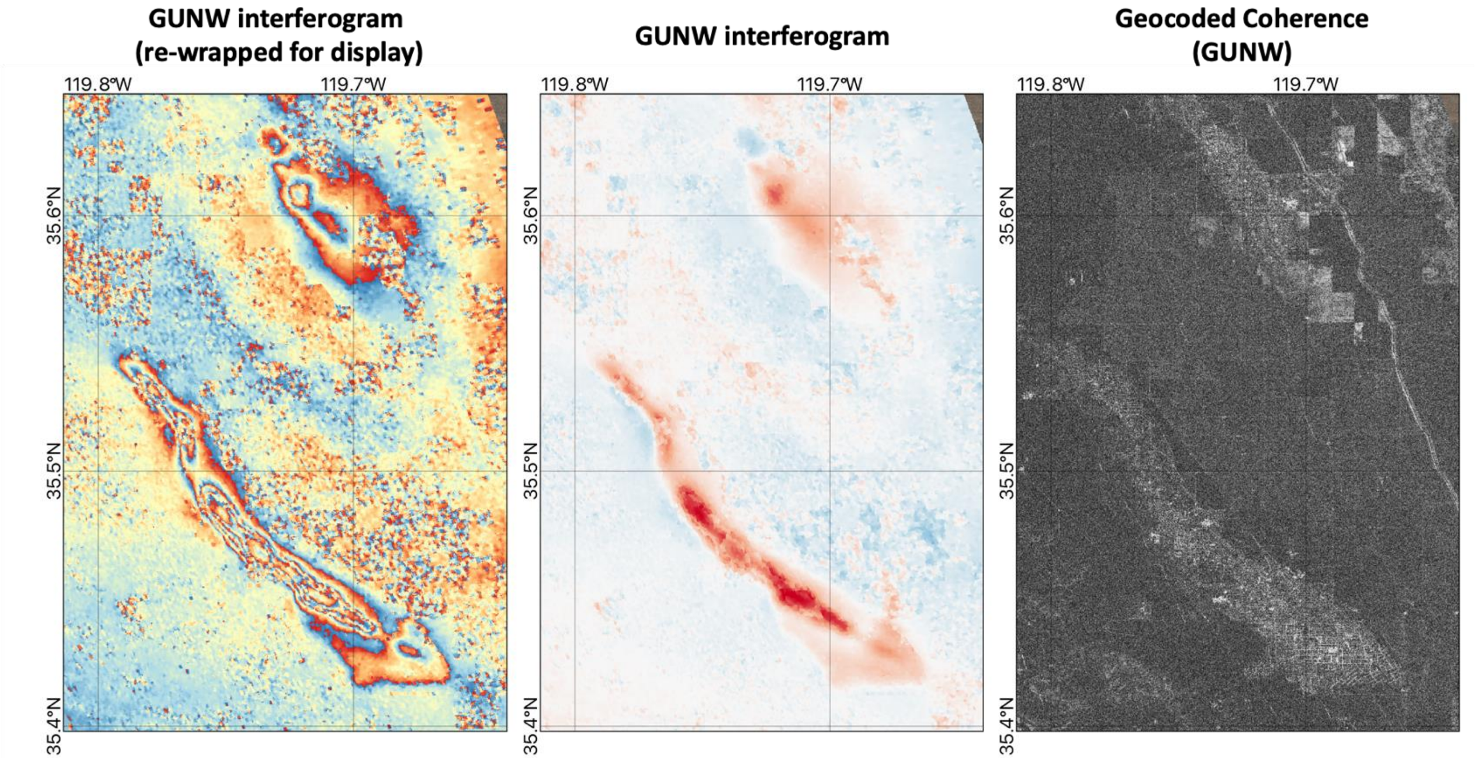
This document has been reviewed and determined not to contain export controlled technical data

NISAR Geocoded Covariance (GCOV)



This document has been reviewed and determined not to contain export controlled technical data

NISAR Range-Doppler (RUNW) & Geocoded (GUNW) Unwrapped Interferogram



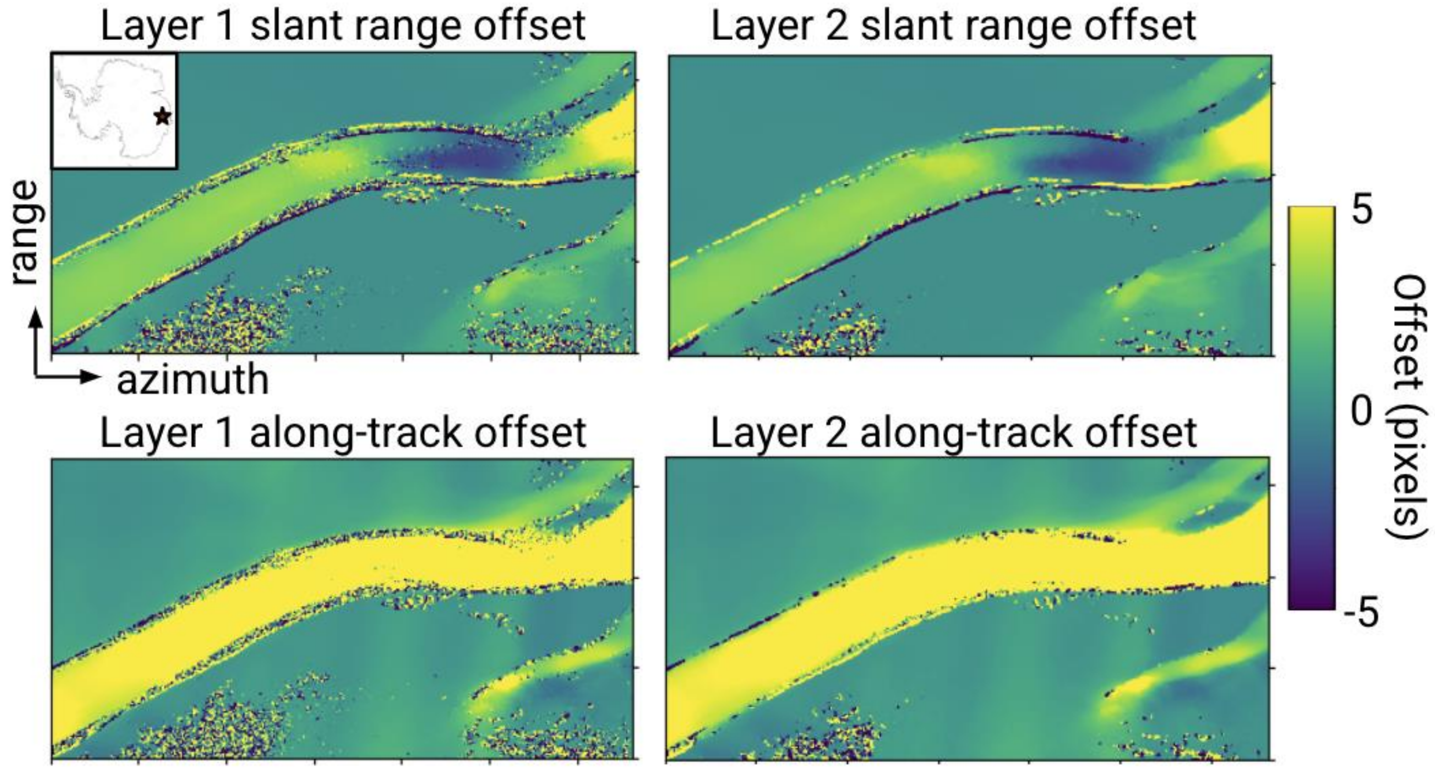
GUNW interferogram (re-wrapped for display)

GUNW interferogram

Geocoded Coherence (GUNW)

This document has been reviewed and determined not to contain export controlled technical data

NISAR Range-Doppler (ROFF) & Geocoded (GOFF) Pixel Offsets


Layer 1 slant range offset
Layer 2 slant range offset
range
azimuth
Layer 1 along-track offset
Layer 2 along-track offset
5
0
-5
Offset (pixels)

This document has been reviewed and determined not to contain export controlled technical data

NISAR Workflows

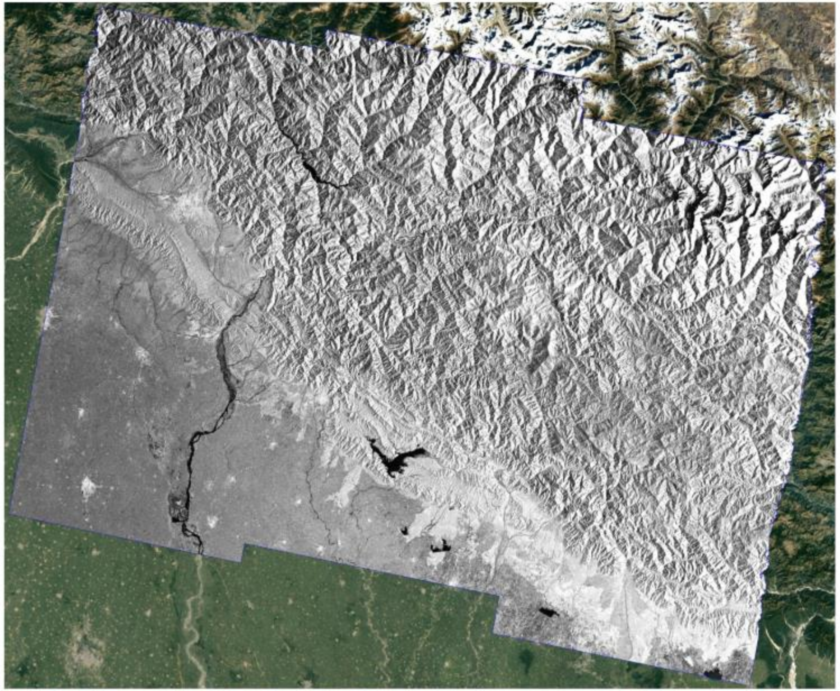
- ISCE3 includes command line tools to generate NISAR science products with custom parameters
- NISAR workflows take configuration files in YAML syntax
  - https://github.com/isce-framework/isce3/blob/develop/share/nisar/defaults
- Example usage:

```
$ python -m nisar.workflows.focus runconfig.yaml
```

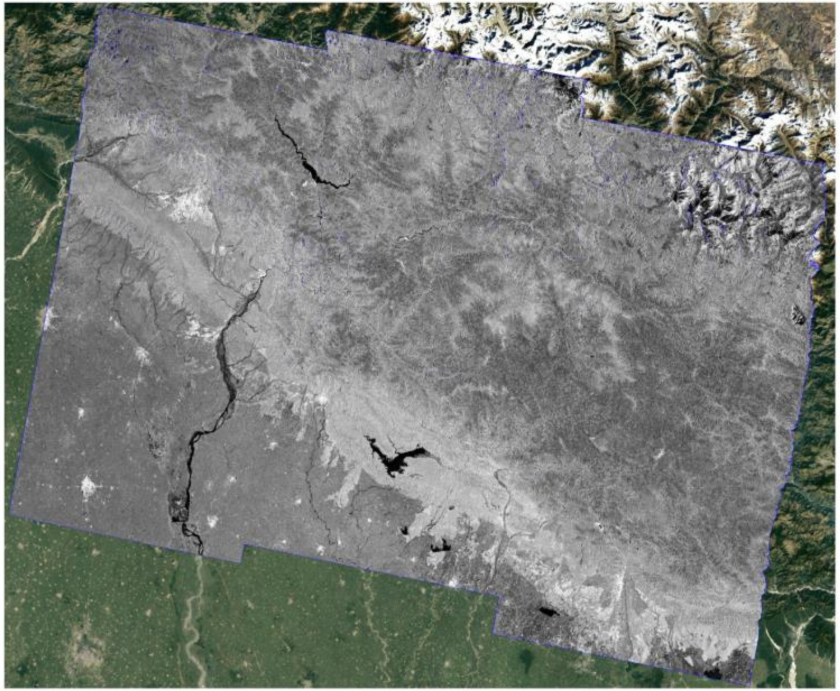
OPERA ❤ ISCE3

- “Observational Products for End-Users from Remote Sensing Analysis”
- Sentinel-1 Co-registered Single-Look Complex (CSLC) and Radiometric Terrain Corrected (RTC) products are generated using ISCE3
  - https://github.com/opera-adt/COMPASS
  - https://github.com/opera-adt/RTC
- Tools for parsing Sentinel-1 products into ISCE3-compatible data structures
  - https://github.com/isce-framework/s1-reader

OPERA Radiometric Terrain-Corrected from Sentinel-1 (RTC-S1)



Uncorrected SAR image - beta-naught
VV (black: 0, white: 0.4)

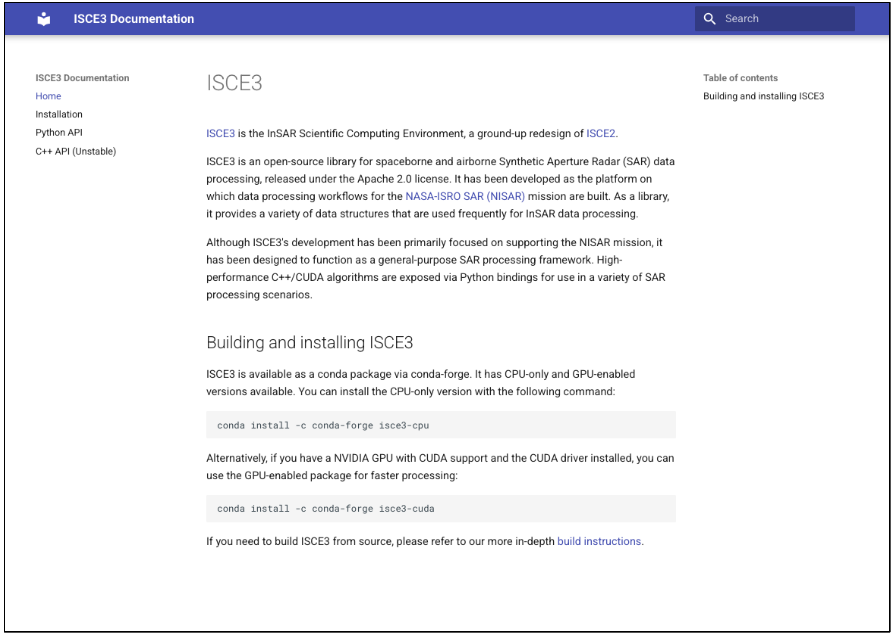


RTC product - gamma-naught
VV (black: 0, white: 0.4)

Current Status & Future Goals

- Beta status; still under active development – API subject to change
- Development recently migrated to github.com 🚀
- Current focus: NISAR Commissioning & Cal/Val
- Working on documentation and tutorials
- Hope to add support for non-NISAR sensor workflows in the future (ALOS, ENVISAT, TSX, Biomass, …)

Current Status & Future Goals



ISCE3

ISCE3 is the InSAR Scientific Computing Environment, a ground-up redesign of ISCE2.

ISCE3 is an open-source library for spaceborne and airborne Synthetic Aperture Radar (SAR) data processing, released under the Apache 2.0 license. It has been developed as the platform on which data processing workflows for the NASA-ISRO SAR (NISAR) mission are built. As a library, it provides a variety of data structures that are used frequently for InSAR data processing.

Although ISCE3's development has been primarily focused on supporting the NISAR mission, it has been designed to function as a general-purpose SAR processing framework. High-performance C++/CUDA algorithms are exposed via Python bindings for use in a variety of SAR processing scenarios.

Building and installing ISCE3

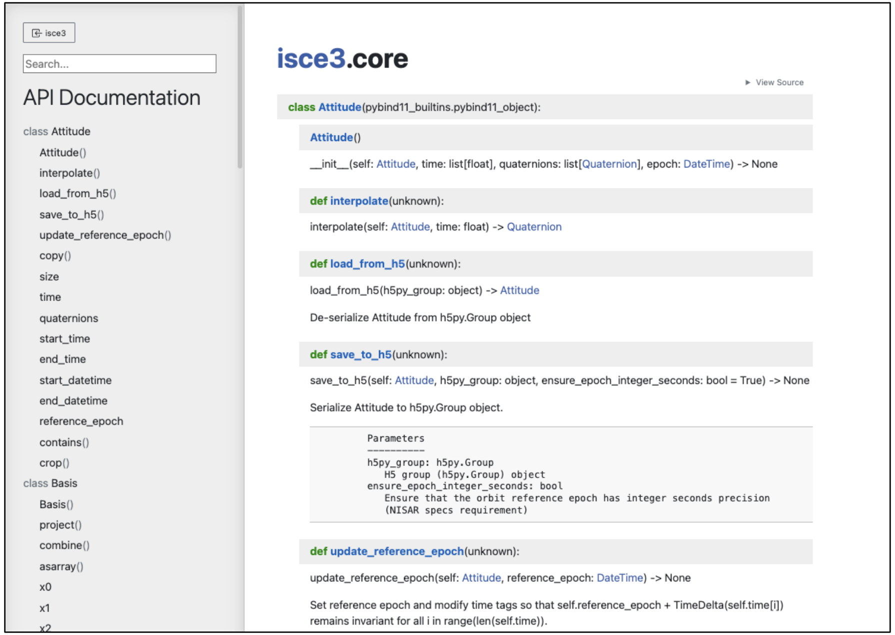
ISCE3 is available as a conda package via conda-forge. It has CPU-only and GPU-enabled versions available. You can install the CPU-only version with the following command:

```
conda install -c conda-forge isce3-cpu
```

Alternatively, if you have a NVIDIA GPU with CUDA support and the CUDA driver installed, you can use the GPU-enabled package for faster processing:

```
conda install -c conda-forge isce3-cuda
```

If you need to build ISCE3 from source, please refer to our more in-depth build instructions.



isce3.core

class Attitude(pybind11_builtins.pybind11_object):

Attitude()

__init__(self: Attitude, time: list[float], quaternions: list[Quaternion], epoch: DateTime) -> None

def interpolate(unknown):

interpolate(self: Attitude, time: float) -> Quaternion

def load_from_h5(unknown):

load_from_h5(h5py_group: object) -> Attitude

De-serialize Attitude from h5py.Group object

def save_to_h5(unknown):

save_to_h5(self: Attitude, h5py_group: object, ensure_epoch_integer_seconds: bool = True) -> None

Serialize Attitude to h5py.Group object.

```
Parameters
----------
h5py_group: h5py.Group
   H5 group (h5py.Group) object
ensure_epoch_integer_seconds: bool
   Ensure that the orbit reference epoch has integer seconds precision
   (NISAR specs requirement)
```

def update_reference_epoch(unknown):

update_reference_epoch(self: Attitude, reference_epoch: DateTime) -> None

Set reference epoch and modify time tags so that self.reference_epoch + TimeDelta(self.time[i]) remains invariant for all i in range(len(self.time)).

ISCE3 Resources

- GitHub: https://github.com/isce-framework/isce3
- Documentation: https://isce-framework.github.io/isce3
- Discussion forum: https://github.com/isce-framework/isce3/discussions

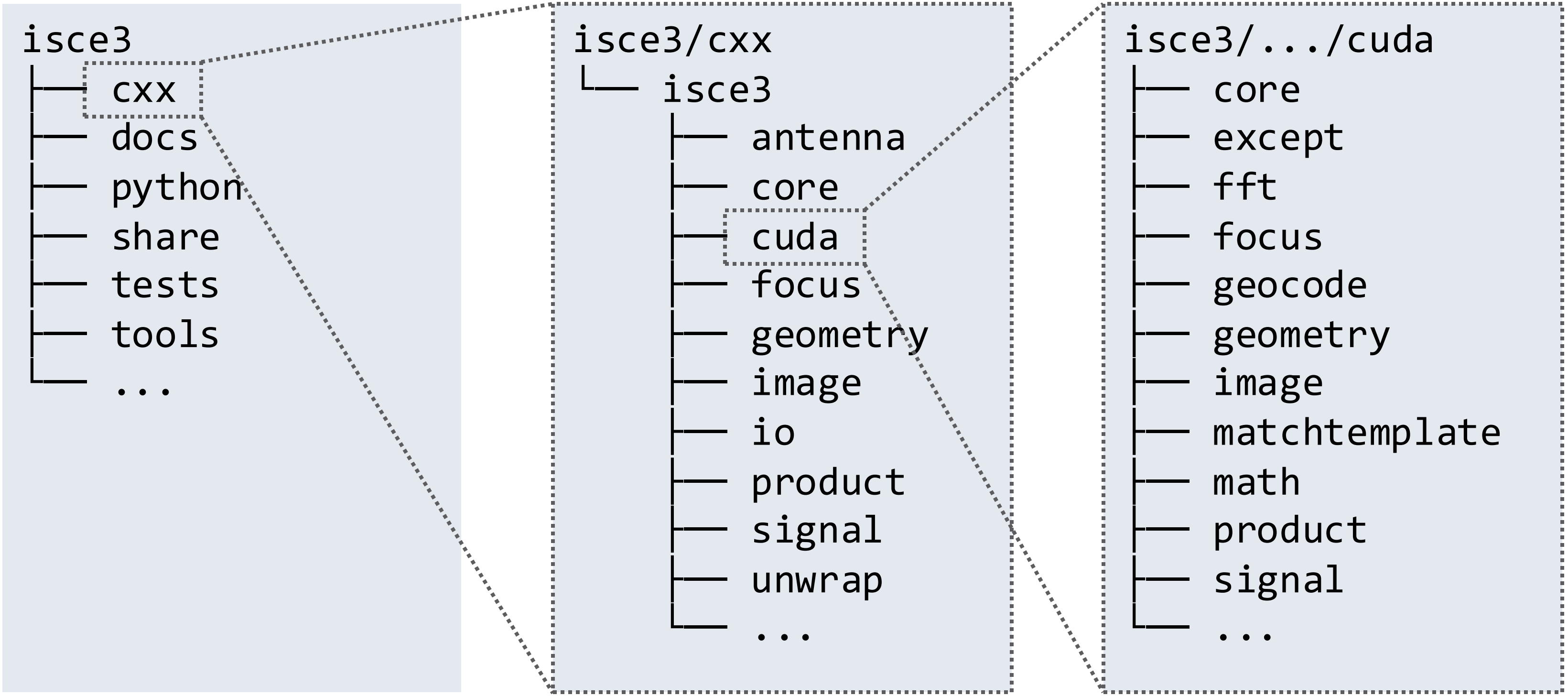
This document has been reviewed and determined not to contain export controlled technical data

Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov

Backups

Repository Layout



```
isce3
├── cxx
├── docs
├── python
├── share
├── tests
├── tools
└── ...
```

```
isce3/cxx
└── isce3
    ├── antenna
    ├── core
    ├── cuda
    ├── focus
    ├── geometry
    ├── image
    ├── io
    ├── product
    ├── signal
    ├── unwrap
    └── ...
```

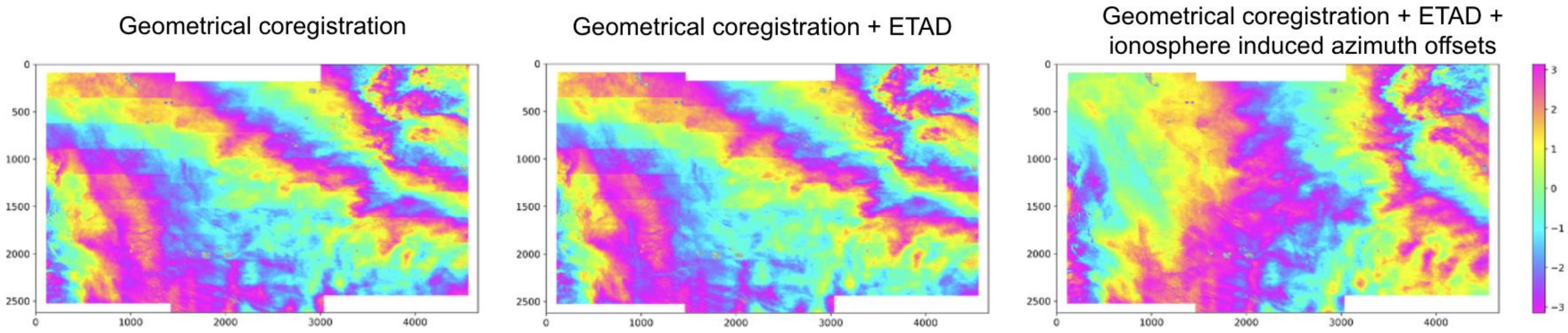
```
isce3/.../cuda
├── core
├── except
├── fft
├── focus
├── geocode
├── geometry
├── image
├── matchtemplate
├── math
├── product
├── signal
└── ...
```

Repository Layout

```
isce3
├── cxx
├── docs
├── python
├── share
├── tests
├── tools
└── ...
```

```
isce3/python
├── extensions
│   └── pybind_isce3
│       ├── antenna
│       └── ...
└── packages
    ├── isce3
    │   ├── antenna
    │   └── ...
    └── nisar
        ├── antenna
        └── ...
```

OPERA Co-registered Single-Look Complex (CSLC)



Geometrical coregistration

Geometrical coregistration + ETAD

Geometrical coregistration + ETAD + ionosphere induced azimuth offsets

Example: Copernicus DEM for NISAR

- Command line usage

```
$ python –m nisar.workflows.stage_dem \
    --bbox -118.57 33.29 -115.20 35.98 \
    --margin 10 --output dem.vrt
```