Dolphin: A Python package for large-scale InSAR PS/DS processing

Scott J. Staniewicz [1], **Sara Mirzaee** [1], **Geoffrey M. Gunter** [1], **Talib Oliver-Cabrera** [1], **Emre Havazli** [1], and **Heresh Fattahi** [1]

1 Jet Propulsion Laboratory, California Institute of Technology

DOI: 10.21105/joss.06997

Software

- Review
- Repository
- Archive

Editor: Michael Mahoney

Reviewers:

- @McWhity
- @margauxmouchene

Submitted: 21 June 2024
Published: 13 November 2024

Summary

Interferometric Synthetic Aperture Radar (InSAR) is a remote sensing technique used for measuring land surface deformation. Conventional InSAR uses pairs of SAR images to create a single map of the relative displacement between the two acquisition times. Dolphin is a Python library that uses state-of-the-art multi-temporal algorithms to reduce the impact of noise sources and produce long time series of displacement at fine resolution.

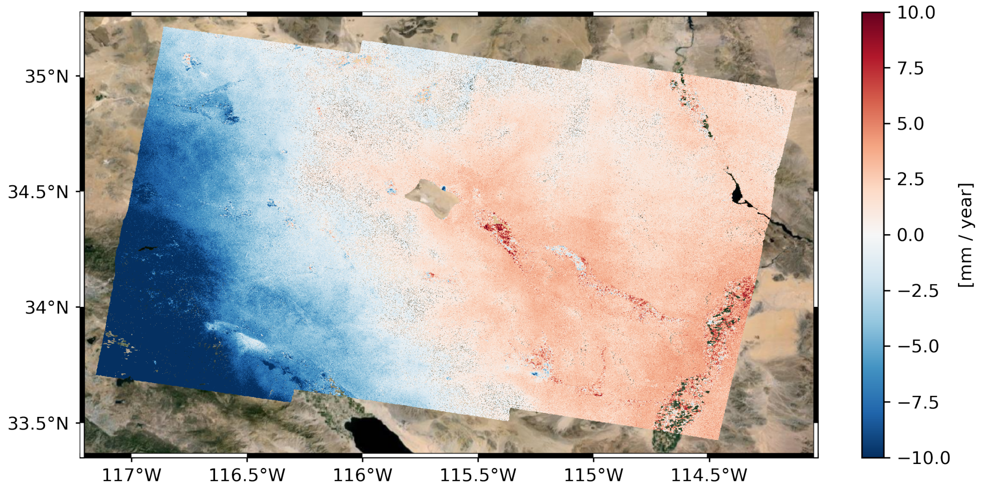

Figure 1: Average surface displacement velocity along the radar line-of-sight between February, 2017 and December, 2020. Red (blue) indicates motion towards (away from) the satellite.

Statement of need

InSAR has been a powerful tool for decades, both in geophysical studies, including tectonics, volcanism, and glacier dynamics, as well as human applications such as urban development, mining, and groundwater extraction. The launch of the European Space Agency's Sentinel-1 satellite in 2014 dramatically increased the availability of free, open-access SAR data. However, processing InSAR data remains challenging, especially for non-experts.

Advanced algorithms combining persistent scatterer (PS) and distributed scatterer (DS) techniques, also known as phase linking, have been developed over the past decade to help overcome decorrelation noise in longer time series (Guarnieri & Tebaldini, 2008). Despite their potential, these methods have only recently begun to appear in open-source tools.

The first phase linking prototype library was the FRInGE C++ library (Fattahi et al., 2019), which implements algorithms and workflows from Ferretti et al. (2011) and Ansari et al. (2018). The MiaplPy Python library contains a superset of the features in FRInGE, as well as new algorithms developed in Mirzaee et al. (2023). Additionally, the MATLAB TomoSAR library was made public in 2022, which implements the "Compressed SAR" (ComSAR) algorithm, a variant of phase linking detailed in Ho Tong Minh & Ngo (2022).

While these tools represent significant progress, there remained a need for software capable of handling the heavy computational demands of large-scale InSAR processing. For example, the TomoSAR library currently requires tens of gigabytes of memory to process more than a small area of interest, while FRInGE and MiaplPy are unable to offer speedups to users who want to process data at a coarser output grid than the full SLC resolution. Additionally, both FRInGE and MiaplPy were designed to process single batches of SLC images.

Dolphin was developed to process both historical archives and incrementally handle new data in near-real time. This capability was specifically designed for the Observational Products for End-Users from Remote Sensing Analysis (OPERA) project. OPERA, a Jet Propulsion Laboratory project funded by the Satellite Needs Working Group (SNWG), is tasked with generating a North American Surface Displacement product covering over 10 million square kilometers of land at 30 meter resolution or finer, with under 72 hours of latency.

Overview of Dolphin

Dolphin processes coregistered single-look complex (SLC) radar images into a time series of surface displacement. The software has an end-to-end surface displacement processing workflow (Figure 2), accessible through a command line tool, which calls core algorithms for PS/DS processing:

- The `shp` subpackage estimates the SAR backscatter distribution to find neighborhoods of statistically homogeneous pixels (SHPs) using the generalized likelihood ratio test from Parizzi & Brcic (2011) or the Kolmogorov-Smirnov test from Ferretti et al. (2011).
- The `phase_link` subpackage processes the complex SAR covariance matrix into a time series of wrapped phase using the CAESAR algorithm (Fornaro et al., 2015), the eigenvalue-based maximum likelihood estimator of interferometric phase (EMI) (Ansari et al., 2018), or the combined phase linking (CPL) approach from Mirzaee et al. (2023).
- The `ps` module selects persistent scatterer pixels from the full-resolution SLCs to be integrated into the wrapped interferograms (Ferretti et al., 2001).
- The `unwrap` subpackage exposes multiple phase unwrapping algorithms, including the Statistical-cost, Network-flow Algorithm for Phase Unwrapping (SNAPHU) (C. W. Chen & Zebker, 2001), the PHASS algorithm (available in the InSAR Scientific Computing Environment (Rosen et al., 2018)), and the Extended Minimum Cost Flow (EMCF) 3D phase unwrapping algorithm via the `spurt` library. Dolphin has pre- and post-processing options, including Goldstein filtering (Goldstein & Werner, 1998) or interferogram masking and interpolation (J. Chen et al., 2015).
- The `timeseries` module contains basic functionality to invert an overdetermined network of unwrapped interferograms into a time series and estimate the average surface velocity. The outputs of Dolphin are also compatible with the Miami INsar Time-series software for users who are already comfortable with MintPy (Yunjun et al., 2019).

To meet the computational demands of large-scale InSAR processing, Dolphin leverages Just-in-time (JIT) compilation, maintaining the readability of Python while matching the speed of compiled languages. The software's compute-intensive routines use the XLA compiler within JAX (Bradbury et al., 2018) for efficient CPU or GPU processing. Users with compatible GPUs can see 5-20x speedups by simply installing additional packages. Dolphin manages memory efficiently through batch processing and multi-threaded I/O, allowing it to handle datasets larger than available memory while typically using a few gigabytes for most processing stages.

These optimizations enable Dolphin to process hundreds of full-frame Sentinel-1 images with minimal configuration, making it well-suited for large-scale projects such as OPERA.

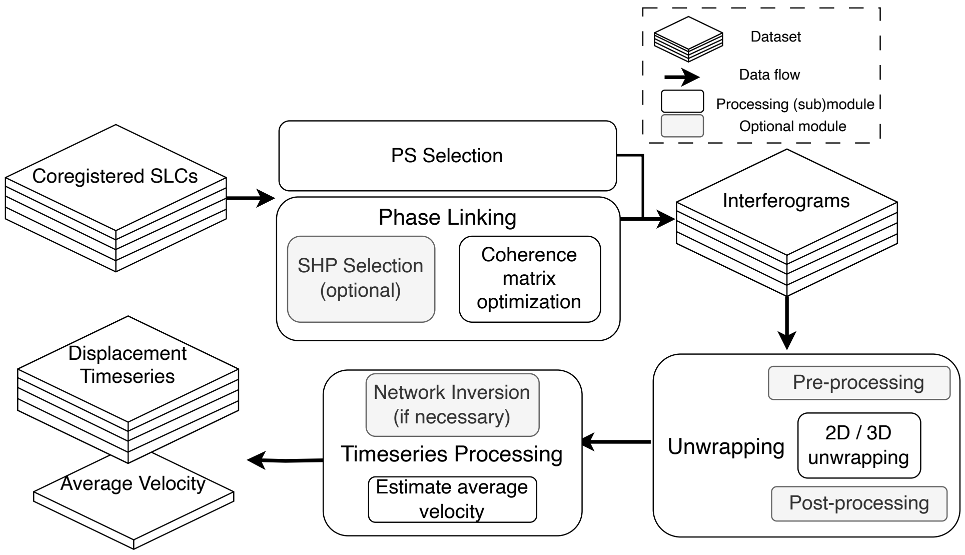

Figure 2: Overview of main workflow to generate surface displacement. Rectangular stacks indicate input or intermediate raster images. Arrows show the flow of data through the configurable submodules of Dolphin.

The Dolphin command line tool provides an interface for running the end-to-end displacement workflow. To illustrate, if a user has created a stack of coregistered SLCs in a `data/` directory, they only need to follow two steps to run the full workflow with all default parameters:

1. Configure the workflow with the `config` command, indicating the location of the SLCs, which dumps the output to a YAML file:

```
dolphin config --slc-files data/*
```

2. Run the workflow saved in the YAML configuration file with the `run` command:

```
dolphin run dolphin_config.yaml
```

The full set of configuration options can be viewed with the `dolphin config --print-empty` command.

Figure 1 shows an example result of the final average surface velocity map created by Dolphin. The inputs were OPERA Coregistered Single-Look Complex (CSLC) geocoded images from Sentinel-1 data between February 2017 and December 2020 over the Mojave Desert.

Acknowledgements

Copyright © 2024, California Institute of Technology ("Caltech"). U.S. Government sponsorship acknowledged. The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). OPERA, managed by the Jet Propulsion Laboratory and funded by the Satellite Needs Working Group, is creating remote sensing products to address Earth observation needs across U.S. civilian federal agencies.

References

Ansari, H., De Zan, F., & Bamler, R. (2018). Efficient Phase Estimation for Interferogram Stacks. *IEEE Transactions on Geoscience and Remote Sensing*, *56*(7), 4109–4125. https://doi.org/10.1109/TGRS.2018.2826045

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs*.

Chen, C. W., & Zebker, H. A. (2001). Two-dimensional phase unwrapping with use of statistical models for cost functions in nonlinear optimization. *Journal of the Optical Society of America A*, *18*(2), 338. https://doi.org/10.1364/JOSAA.18.000338

Chen, J., Zebker, H. A., & Knight, R. (2015). A persistent scatterer interpolation for retrieving accurate ground deformation over InSAR-decorrelated agricultural fields. *Geophysical Research Letters*, *42*(21), 9294–9301. https://doi.org/10.1002/2015GL065031

Fattahi, H., Agram, P. S., Tymofyeyeva, E., & Bekaert, D. P. (2019). *FRInGE; Full-Resolution InSAR timeseries using Generalized Eigenvectors*. *2019*, G11B–0514.

Ferretti, A., Fumagalli, A., Novali, F., Prati, C., Rocca, F., & Rucci, A. (2011). A new algorithm for processing interferometric data-stacks: SqueeSAR. *IEEE Transactions on Geoscience and Remote Sensing*, *49*(9), 3460–3470. https://doi.org/10.1109/TGRS.2011.2124465

Ferretti, A., Prati, C., & Rocca, F. (2001). Permanent Scatters in SAR Interferometry. *IEEE Transactions on Geoscience and Remote Sensing*, *39*(1), 8–20. https://doi.org/10.1109/36.898661

Fornaro, G., Verde, S., Reale, D., & Pauciullo, A. (2015). CAESAR: An Approach Based on Covariance Matrix Decomposition to Improve Multibaseline–Multitemporal Interferometric SAR Processing. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(4), 2050–2065. https://doi.org/10.1109/TGRS.2014.2352853

Goldstein, R. M., & Werner, C. L. (1998). Radar interferogram filtering for geophysical applications. *Geophysical Research Letters*, *25*(21), 4035–4038. https://doi.org/10.1029/1998GL900033

Guarnieri, A. M., & Tebaldini, S. (2008). On the Exploitation of Target Statistics for SAR Interferometry Applications. *IEEE Transactions on Geoscience and Remote Sensing*, *46*(11), 3436–3443. https://doi.org/10.1109/TGRS.2008.2001756

Ho Tong Minh, D., & Ngo, Y.-N. (2022). Compressed SAR Interferometry in the Big Data Era. *Remote Sensing*, *14*(2), 390. https://doi.org/10.3390/rs14020390

Mirzaee, S., Amelung, F., & Fattahi, H. (2023). Non-linear phase linking using joined distributed and persistent scatterers. *Computers & Geosciences*, *171*, 105291. https://doi.org/10.1016/j.cageo.2022.105291

Parizzi, A., & Brcic, R. (2011). Adaptive InSAR Stack Multilooking Exploiting Amplitude Statistics: A Comparison Between Different Techniques and Practical Results. *IEEE Geoscience and Remote Sensing Letters*, *8*(3), 441–445. https://doi.org/10.1109/LGRS.2010.2083631

Rosen, P. A., Gurrola, E. M., Agram, P., Cohen, J., Lavalle, M., Riel, B. V., Fattahi, H., Aivazis, M. A. G., Simons, M., & Buckley, S. M. (2018). The InSAR Scientific Computing Environment 3.0: A Flexible Framework for NISAR Operational and User-Led Science Processing. *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 4897–4900. https://doi.org/10.1109/IGARSS.2018.8517504

Yunjun, Z., Fattahi, H., & Amelung, F. (2019). Small baseline InSAR time series analysis:

Unwrapping error correction and noise reduction. *Computers & Geosciences*, *133*, 104331. https://doi.org/10.1016/j.cageo.2019.104331