SARXarray and STMtools

Open-Source Python Libraries for InSAR Data Processing and Analysis

Ku, O.; Alidoost, F.; Chandramouli, P.; van Lankveld, T.; Nattino, F.; Grootes, M. W.; van Leijen, F. J.; Hanssen, R. F.

DOI
10.1109/IGARSS53475.2024.10642762

Publication date
2024

Document Version
Final published version

Citation (APA)
Ku, O., Alidoost, F., Chandramouli, P., van Lankveld, T., Nattino, F., Grootes, M. W., van Leijen, F. J., & Hanssen, R. F. (2024). *SARXarray and STMtools: Open-Source Python Libraries for InSAR Data Processing and Analysis*. 11036-11038. Paper presented at 2024 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2024, Athens, Greece. https://doi.org/10.1109/IGARSS53475.2024.10642762

IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium | 979-8-3503-6032-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IGARSS53475.2024.10642762

SARXARRAY AND STMTOOLS: OPEN-SOURCE PYTHON LIBRARIES FOR INSAR DATA PROCESSING AND ANALYSIS

O. Ku, F. Alidoost, P. Chandramouli, T. van Lankveld, F. Nattino, M. W. Grootes

Netherlands eScience Center, Amsterdam, the Netherlands

F.J. van Leijen, R. F. Hanssen

Delft University of Technology, Department of Geoscience & Remote Sensing, Delft, the Netherlands

ABSTRACT

We introduce *SARXarray* and *STMtools*, two python libraries designed to support the exploitation of modern Interferometric Synthetic Aperture Radar (InSAR) data by enabling handling of larger-than memory datasets and the incorporation and fusion with relevant contextual information. The libraries are developed upon two innovative and well-established open-source Python libraries: Xarray [5] and Dask [6]. They are implemented as Xarray extensions. *SARXarray* is designed to manipulate and operate on larger-than-memory coregistered raster stacks such as Single-look Complex images (SLC) or interferograms, performing scatterer selection and producing STM objects. *STMtools* leverage the Space-Time Matrix (STM) concept [3,4] and provides functionalities to process STM and perform enrichment/data fusion with other data sources. Both libraries are built on the Xarray library, providing support for a wide range of data formats, and utilize Dask for parallel computation, making them scalable for distributed computation infrastructures. By enabling InSAR data analysis incorporating contextual information, the two libraries enhance the potential to uncover underlying mechanisms driving deformation phenomena.

Index Terms— InSAR, PSI, Distributed Computation, Python, Xarray, Dask

1. INTRODUCTION

The observations acquired by Interferometric Synthetic Aperture Radar (InSAR) can reveal ground surface displacement characteristics driven by various mechanisms. Interpreting these observations with related contextual data brings opportunities and challenges to the research topic. On one hand, the successful application of contextual data related to the displacement mechanism can constrain the complex and ambiguous InSAR observations, providing deeper insights into the displacement process. On the other hand, efficiently enriching the extensive InSAR observations with contextual data from diverse sources, i.e. performing data fusion across the (potentially large) InSAR data set, poses a significant computational challenge. Moreover, the associated data volumes are increasing continuously.

To address these challenges, the Space-Time Matrix (STM) concept [3, 4] has been introduced to store InSAR datasets along with their corresponding contextual information. We introduce two open-source Python libraries, *SARXarray* and *STMTools*, that extend this concept for data processing in Python. These libraries are designed to manage and enrich large InSAR datasets and can be integrated into the InSAR data processing pipeline. They facilitate tasks ranging from loading coregistered radar stacks to interpreting coherent radar scatterers, offering seamless scalability to distributed computation infrastructures.

2. THE TWO PYTHON LIBRARIES

Both *SARXarray* and *STMTools* are developed as extensions of the open-source Python library Xarray [5]. They leverage Xarray's support for labelled multi-dimensional arrays to represent the spatial-temporal features of InSAR data. They support exporting results in Zarr [8], which is a file storage format for chunked, compressed, N-dimensional arrays. Additionally, they utilize the Dask library [6] to perform data chunking and lazy evaluation, enabling the scaling up of the data processing pipeline and parallel processing of larger-than-memory data on various computational infrastructures, ranging from large distributed systems to personal laptops.

2.1. *SARXarray*

SARXarray is designed to handle stacks of complex data in raster format, such as coregistered Single-Look Complex (SLC) images or interferogram stacks. It reads and writes large data stacks as chunked Xarray object (see Figure 1 as an example), which is a user-friendly data model in Python and is often used to represent multi-dimensional scientific data with labels. SAR and InSAR related calculations, such as multi-looking, coherence estimation, and scatterer selection, can be performed chunk-wise, making the processing scalable. As such, *SARXarray* can be used to efficiently process InSAR data with large spatio-temporal coverage.

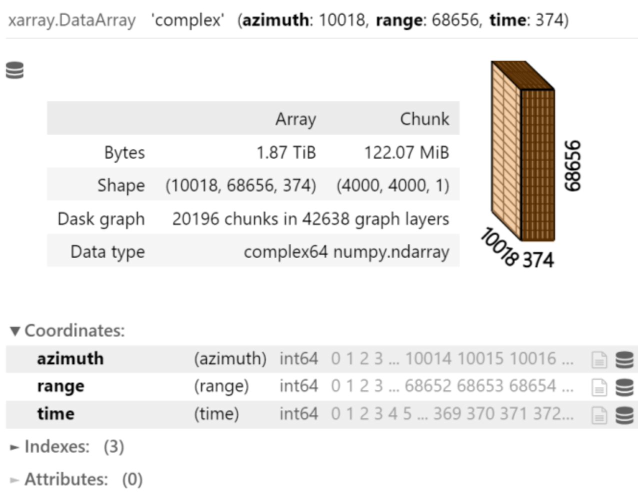

Figure 1 Example of a coregistered Sentinel-1 interferogram stack loaded as an Xarray DataArray object using *SARXarray*, visualized in Jupyter Notebook [9]. The stack contains 374 interferograms with 10018 pixels in azimuth and 68656 pixels in range. Lazy loading is performed, meaning the data is only loaded into memory as needed. The stack is divided into chunks with a size of 4000 azimuth and 4000 range per interferogram. This allows for parallel computation on each chunk.

2.2. *STMTools*

STMTools is designed to present and process InSAR data and corresponding contextual attributes in Space-Time Matrix (STM) [4, 5] format. The STM concept enables the enrichment of InSAR data with location (e.g., Point Scatterer or Distributed Scatterer), epoch, and spatio-temporal related attributes. *STMTools* works with the selected scatterers output from *SARXarray*, as well as other well-known data formats such as csv files. An enrichment functionality is provided to query corresponding contextual information in the background, such as geospatial polygons, and then add the attributes-of-interest to the existing STM object.

Both the *SARXarray* and *STMTools* software repositories [1, 2] contain example Python notebooks to demonstrate the functionality of the libraries.

3. EXAMPLE APPLICATIONS

The *SARXarray* and *STMTools* libraries can be used at different stages of an InSAR data processing chain, focusing on Point Scatterers and/or Distributed Scatterers. We demonstrate their applications through two independent examples of data processing, which are illustrated in Figures 2 and 3, respectively.
Figure 2 shows an example of python native InSAR processing using *SARXarray* early in the data processing pipeline. The figure shows the temporal coherence matrix for part of a Sentinel-1 data stack over the island St. Eustatius in the Caribbean. Using *SARXarray*, the required processing can be expressed and performed natively in python regardless of the data volume, i.e. whether the data fits in memory. The coherence estimates are georeferenced afterwards.

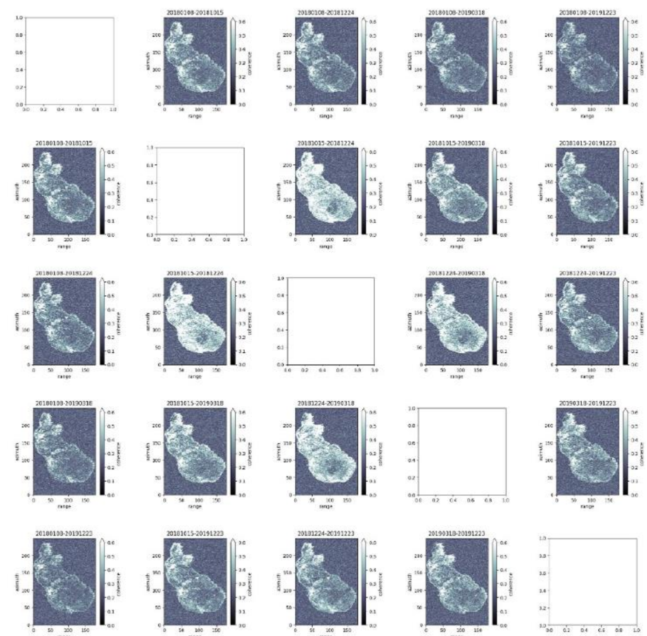

Figure 2 Example of efficiently generated coherence estimates for part of a Sentinel-1 data stack over St. Eustatius in the Caribbean [7]. The figure shows the coherence estimates between five SAR acquisitions ordered in time along the axis. The grayscale indicates the coherence level (white for high coherence, black for low coherence). The coherence matrix is generated by *SARXarray*, which effectively use Dask to read larger-than-memory data stacks.

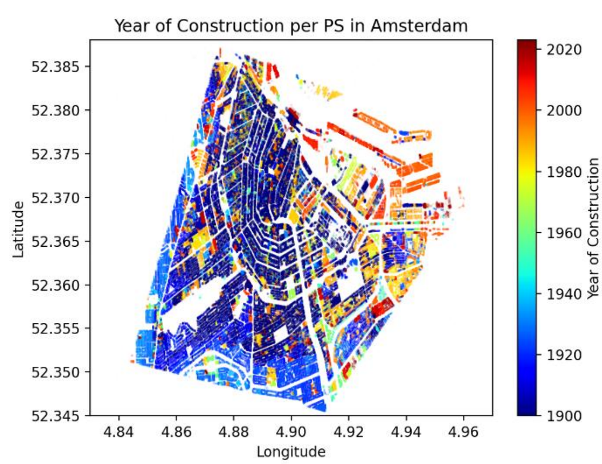

Figure 3 A visualization of Point Scatterers (PS) and the year of construction of the corresponding building where the PS points are located. The horizontal and vertical coordinates are respectively longitude and latitude, both in WGS84 reference. The color scheme indicates the year of construction. The PS dataset is selected from a TerraSAR-X interferogram stack over the Amsterdam region, the Netherlands. The year of construction information is provided by the BAG cadaster dataset as geo-referenced polygons. The PS datasets are

enriched with contextual information from BAG polygons using *STMTools*.

Figure 3 shows an example of enriching large-volume Point Scatterers (PS) with contextual information using *STMTools*. Specifically, a PS dataset (corresponding to buildings in Amsterdam) has been enriched with the year of construction as supplied by the BAG (Dutch: *Basisregistratie Adressen en Gebouwen*, Basic Registration Addresses and Buildings) cadaster dataset [10]. The PS dataset is stored as an STM instance with InSAR-related attributes, while the BAG is provided as a Geo-Package (gpkg) file. Using the functionality provided by *STMtools*, polygons provided by BAG are loaded and efficiently queried between points and polygons. In *STMtools* this process supports chunk-wise operations and scales seamlessly to out-of-core operations and larger-than-memory datasets.

4. CONCLUSIONS

The open-source Python libraries *SARXarray* and *STMtools* provide native Python support for manipulating and operating on large InSAR data sets, regardless of data volume. Furthermore, leveraging the STM concept, the tools provide efficient and intuitive support for further processing and data fusion of InSAR data with a wide variety of other (spatio-temporal) data sources. Building on the established and community supported state-of-the-art Xarray and Dask libraries, these libraries form the basis for open, community supported InSAR processing and InSAR/data fusion based science with current and upcoming InSAR data sources.

5. REFERENCES

[1] Ku, O., Alidoost, F., Chandramouli, P., Nattino, F., & van Leijen, F. sarxarray [Computer software]. https://github.com/MotionbyLearning/sarxarray

[2] Ku, O., Alidoost, F., Chandramouli, P., Grootes, M. W., van Lankveld, T., Nattino, F., & van Leijen, F. stmtools [Computer software]. https://github.com/MotionbyLearning/stmtools.

[3] Bruna, M. F. D., van Leijen, F. J., & Hanssen, R. F. (2021). A Generic Storage Method for Coherent Scatterers and Their Contextual Attributes. In 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS: Proceedings (pp. 1970-1973). [9553453] (International Geoscience and Remote Sensing Symposium (IGARSS); Vol. 2021-July). IEEE . https://doi.org/10.1109/IGARSS47720.2021.9553453

[4] van Leijen, F. J., van der Marel, H., & Hanssen, R. F. (2021). Towards the Integrated Processing of Geodetic Data. In 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS: Proceedings (pp. 3995-3998). [9554887] IEEE . https://doi.org/10.1109/IGARSS47720.2021.9554887

[5] Hoyer, S., & Joseph, H. (2017). xarray: N-D labeled Arrays and Datasets in Python. Journal of Open Research Software, 5(1). https://doi.org/10.5334/jors.148

[6] Dask Development Team (2016). Dask: Library for dynamic task scheduling.
URL http://dask.pydata.org

[7] Bosman, Z. (2023). Towards Small-Baseline InSAR Time Series Analysis for Volcanic Monitoring on Saba and St. Eustatius. MSc thesis, Delft University of Technology, the Netherlands.

[8] Alistair Miles, "zarr-developers/zarr-python: v2.16.1". Zenodo, Aug. 18, 2023. doi: 10.5281/zenodo.8263439.

[9] Kluyver, T. et al., 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt, eds. Positioning and Power in Academic Publishing: Players, Agents and Agendas. pp. 87–90.

[10] Basisregistratie Adressen en Gebouwen (Basic Registration Addresses and Buildings) dataset:
https://data.overheid.nl/dataset/0ff83e1e-3db5-4975-b2c1-fbae6dd0d8e0