THE INSAR SCIENTIFIC COMPUTING ENVIRONMENT 3.0: A FLEXIBLE FRAMEWORK FOR NISAR OPERATIONAL AND USER-LED SCIENCE PROCESSING

Paul A. Rosen[1], Eric M. Gurrola[1], Piyush Agram[1], Joshua Cohen[1], Marco Lavalle[1], Bryan V. Riel[1], Heresh Fattahi[1], Michael A.G. Aivazis[2], Mark Simons[3], Sean M. Buckley[1]

[1] Jet Propulsion Laboratory, California Institute of Technology, Pasadena CA, 91109, USA
[2] ParaSim, Inc., Los Angeles, CA, 90021, USA
[3] Division of Geological and Planetary Sciences, California Institute of Technology, Pasadena CA, 91125, USA

ABSTRACT

The InSAR Scientific Computing Environment (ISCE) was first developed under the NASA Advanced Information Systems Technology as a flexible, extensible object-oriented framework for Interferometric Synthetic Aperture Radar (InSAR) processing. The ISCE framework uses Python 3 at the workflow level, controlling modules of compiled code for functional processing, and managing inputs, outputs, and other flow control services. The currently released version, called ISCE 2.1, is distributed to the research community through the Western North America InSAR Consortium under a research license. The ISCE team is working on the next generation of the code in order to prepare for the NASA-ISRO SAR (NISAR) mission operational processing. Innovations in this code include augmentation or conversion of the custom Python framework elements in ISCE with the Pyre framework, new workflows for interferometric and polarimetric stack processing, a more intuitive and graphically based user interface, and flow control for hybrid computing environments including CPU/GPU clusters, logging and error tracking facilities, and new more efficient computational modules that exploit graphical processor units (GPUs) when available. The ISCE 3.0 framework is designed to work in an operational environment as well as on a single user's laptop or compute cluster, with services to discover capabilities and scale computations accordingly.

Index Terms— InSAR processing, geodetic imaging, computational frameworks, Earth science informatics

1. INTRODUCTION

During the 1990's, in early days of synthetic aperture radar interferometry for geodetic imaging, researchers at the Jet Propulsion Laboratory and the California Institute of Technology (JPL/Caltech) developed the Repeat Orbit Interferometry Package, known as ROIPAC, to allow scientists to generate their own interferograms from the available radar data sets at the time – ERS, RADARSAT-1 and JERS [1]. ROIPAC was a set of executables for image formation, offset estimation, resampling, interferogram formation, topographic correction, filtering, unwrapping and geocoding. Over the years, sensors were added, including ALOS and ENVISAT, and the source code was distributed for research by the Open Channel Foundation for NASA.

ROIPAC served an important role for the community, offering a free and adaptable software suite for both using in research and for showing the fundamental methods of the techniques through the source code. A number of other software packages were developed based on ROIPAC's code, including extensions of the code itself [e.g., 2,3].

In the 2007 timeframe, it was becoming clear that a new generation of community code would be needed to support the burgeoning InSAR community. Not only were the number of users growing exponentially, but the number of sensors, the availability of large quantities of data, and the number of InSAR-related workflows and techniques were all growing rapidly. ROIPAC – a set of compiled executable programs tied together with Perl scripts – was limited in its flexibility and extensibility to adapt to the rapidly changing environment. In 2008, NASA conducted a workshop to define the attributes of the next generation InSAR software. From these recommendations, NASA's Advanced Information Systems Technology program funded a prototype known as the InSAR Scientific Computing Environment (ISCE), which was architected to be a flexible, extensible framework for controlling processing modules through Python object oriented formalisms [4,5,6,7]. ISCE supports many more sensors and modalities, and because the modules are componentized, science users have been able to create custom Python scripts for their research purposes without any modifications to the core processing routines themselves. ISCE 2.1 is available by download for research through the Western North America InSAR (WInSAR) Consortium [8], and is in broad distribution.

NASA/JPL, in partnership with the Indian Space Research Organisation (ISRO) is currently developing the NASA-ISRO SAR (NISAR) Mission for launch in 2021

[9,10]. With an intrinsic 240 km swath width and high capacity storage and downlink system, NISAR will image all of Earth's land and ice-covered surfaces from both ascending and descending portions of the orbit every 12 days. The mission will produce polarimetric imagery and interferometric products over the life of the mission. To meet the demands of production for a petabyte scale raw data set, the project is re-architecting ISCE to serve production requirements in addition to the user community's needs. To accomplish this, the framework for ISCE is being extended to include additional services and facilities, and the core processing routines are being reviewed and updated for efficiency and throughput. This paper describes this third generation ISCE software – ISCE 3.0.

2. ISCE ARCHITECTURAL ELEMENTS

ISCE is constructed as a Python 3 framework controlling the flow of data and computation, and keeping track of the computing environment. Intensive computations are performed in compiled object code that are bound to Python-level components. The Python bindings determine which local variables and arrays in the object code are exposed to users and developers. The formalism of communicating between Python and compiled code through bindings adds an extra layer of work, but enforces a structure that is systematic and straightforward to implement, and provides a degree of flexibility to reuse components at the Python level in new ways without touching the compiled code. The bindings are implemented using Cython [11], which allows a straightforward, recipe-driven approach to their development. However, custom bindings using native Python APIs are also possible, and provide an even greater level of control and generality for developers.

Many of the components in original versions of ISCE were written in Fortran, deriving from ROIPAC. In ISCE 3.0, these codes are translated to C++. This translation serves two purposes: it simplifies the interfaces to Cython and Python binding APIs, and it facilitates the conversion of the code to a GPU implementation.

In ISCE 2.1, we developed a custom Python 3 framework based on concepts from another prototype framework known as Pyre [12]. While ISCE 1 and 2 were being developed, Pyre also advanced significantly, to the extent that for ISCE 3.0, blending elements of ISCE 2.1 with Pyre quickly provides most of the facilities and services for an operational version of ISCE.

The underlying software technologies employed in ISCE 3.0, in addition to Python and Pyre, include GDAL for coordinate transformations and file format conversions, Armadillo for matrix manipulation, Cython for bindings, CUDA for GPU coding, FFTW for fast fourier transform calculations, ImageMagick, grace, and Motif for image display support, HDF5 for hdf file format support, and RelaxIV and Pulp to support phase unwrapping of connected components. ISCE has a custom image API on top of the GDAL interface, and custom tensor array manipulation technology is being prototyped to replace Armadillo.

3. ISCE WORKFLOWS

ISCE 2.1 provides several standard applications workflows, including two-pass interferometry from raw or single-look complex (SLC) radar image pairs through to geocoded unwrapped interferograms, and two-pass dense offset estimation for fast motion tracking, e.g. on ice sheets or glaciers. Several of the most computationally intensive algorithms in components have been migrated to GPUs, including offset estimation and radar-to-cartographic geometry conversions. The code is uses the CUDA code library as a means of mapping ISCE algorithms to the GPU architecture.

In addition to the fully configured ISCE workflows just mentioned, ISCE 2.1 has several serialized workflows for processing Sentinel-1 TOPS-mode interferometric pairs, SLC stack coregistration, ionospheric correction, and others [13,14]. These contributed workflows take advantage of the ISCE components directly, but do not use the workflow services such as message logging that ISCE provides. ISCE 3.0 provides fully configured workflows for all core and contributed workflows that are supported.

One of the key advantages of Pyre in ISCE 3.0 is its mature facilities and services for working in a production environment. Coming from a background of computational geophysics and fluid dynamics, where applications have been deployed on massively parallel machines, GPU clusters, and standard CPU clusters, Pyre has developed a set of formalisms for assessing the state of available hardware dynamically and deciding how to deploy applications or components in a suitably efficient way, for measuring performance dynamically and adapting to changes in the environment, and for logging state information suitable for robust provenance tracking. These features are fundamental to the production workflows to be developed for NISAR.

4. ISCE ALGORITHMS

The key algorithms for NISAR production processing take data from raw radar pulse echoes to complex images, interferograms, polarimetric covariance products, and a number of related products such as unwrapped phase. Each of these products is calculated in radar coordinates as well as in geocoded coordinates tied to the reference digital elevation model. These products, shown in Table 1, dictate the basic algorithmic functionality in ISCE 3.0.

For image formation, ISCE has relied on traditional range-Doppler processing with secondary range migration corrections, but with the wide swath and bandwidths expected for NISAR, the image formation processor may need to be enhanced. The image formation processor will be

Table 1. Proposed NISAR L-band L0-L2 products

Product	Description
Incoming Data (L0A)	Raw data with basic metadata added by SDS
Radar Signal Data (L0B)	Corrected, aligned radar pulse data. L0B product is used to derive L1 products.
Range-Doppler Single Look Complex (SLC)	Standard L1 product that will be used to generate all higher level products
Multi-Look Detected	Multi-looked amplitude product in ground range coordinates.
Geocoded SLC (GSLC)	Geocoded L1 SLC product using precise orbits and a DEM.
Nearest-Time Interferogram (IFG)	Multi-looked flattened (WGS84 ellipsoid) Interferogram with topographic fringes in Range-Doppler coordinates.
Nearest-Time Unwrapped Interferogram (UNW)	Multi-looked, unwrapped differential Interferogram in Range-Doppler coordinates.
Geocoded Nearest-Time Unwrapped Interferogram (GUNW)	Geocoded multi-looked unwrapped differential Interferogram. Same as UNW but resampled onto a UTM grid.
Geocoded Polarimetric Scattering Vector (GSLC)	Geocoded L1 SLC product (2 or 4 layers) using precise orbits and a DEM.
Polarimetric Covariance Matrix (COV)	Polarimetric covariance matrix (3, 6 or 10 layers) in Range-Doppler coordinates.
Geocoded Polarimetric Covariance Matrix (GCOV)	Geocoded polarimetric covariance matrix (3 or 6 layers) using precise orbits and a DEM.

a component or collection of components implemented using multiple algorithms for NISAR, selectable by the framework for testing and cross comparison.

One of the innovations for NISAR is the production of a geocoded SLC product. With the availability of accurate orbits and atmospheric delay models, and using either resampling of the conventionally produced image or direct back-projection calculation, depending on the speed of the available hardware, we expected all geocoded SLC images to be well aligned for direct inter-comparison. Topographically induced phase variations can also be removed at the time of image formation, such that subsequent cross-multiplication of images yields interferograms that do not have a significant topographic phase component.

Key to these algorithms is a fast way to map the radar data from radar coordinates to cartographic coordinates. ISCE has GPU accelerated forward and back mapping calculations, and stores the results in tables for applying to common-referenced stacks of images.

ISCE also has a number of phase unwrapping algorithms, including the traditional branch cut technique found in ROIPAC, as well as SNAPHU and secondary arc estimation methods to connect isolated connected components.

5. USER INTERFACE

The XML input files that are used to control ISCE 2.1 are difficult for many users to interpret and edit. ISCE 3.0 will address this in two ways. First, there will be an option to use a control file with a more straightforward design with minimal tagging. While less general than XML, it will be easier to read and edit for typical users. There will be a one-to-one correspondence between XML and these inputs, so either will work in any given workflow. Second, we are developing a web-based graphical user interface (GUI) that will obviate the need for many users to touch control files. The user interface will expose control elements in an intuitive way to users, and control files will be generated in the background. The GUI graphical objects focus on the products, which ultimately are what users desire, with options for workflow processes and functions presented to users based on the products specified.

6. FUTURE PLANS FOR ISCE 3.0

ISCE 3.0 is currently under development. It is anticipated that the first functional version will be available as open source in the Fall 2018 timeframe. This first release should have all the basic functional elements of traditional InSAR processing, as well as some basic stack-processing capability, supporting Sentinel-1, ALOS-2, and the NASA/JPL UAVSAR airborne system. The release mechanism will be through GITlab source management, which will then allow the community of developers to examine the code structure, coding and data formatting conventions, and documentation, and download the code as source or virtual machines. ISCE 3.0 will be tested on cloud instances, allowing users to deploy it themselves on the cloud.

The NISAR production team will begin NISAR-specific components development around the same timeframe, including the NISAR SAR processor, in appropriate data formats using UAVSAR data as simulated data sets. As the science team develops algorithms for higher level products, these will be adapted to the framework and added as applications. Community members will be encouraged to contribute their algorithms using framework formalisms.

One example of an algorithm suite that uses the ISCE core functionalities and is inspired by the framework formalism is PLAnT [15], the Polarimetric-interferometric Lab and Analysis Tool developed at the Jet Propulsion Laboratory to support processing and analysis of SAR data for ecosystem and land-cover/land-use change science and applications. PLAnT inherits from ISCE the low-level code elements, including data structures, serialization interfaces and Python/C++ binding mechanisms. PLAnT also adopts the base ISCE L1/L2 products definition and metadata formats, which allows the user to focus on the development of ecosystem algorithms and customization of associated L2/L3 products.

For NISAR geodetic imaging requirements, the science team plans to use capabilities in the GIAnT time-series tool suite [16]. The authors of GIAnT intend to create a new version in the coming years to support NISAR specifically, with an opportunity to use ISCE 3.0 components and its framework formalism in a similar fashion to PLAnT.

WInSAR has conducted annual summer training on ISCE 2.1, and it is anticipated that the training will transition to ISCE 3.0 in 2019. In addition, the NISAR project in the US and India will be conducting science workshops as launch approaches, and it is anticipated that the product and processing training will be a key element of the activities.

7. CONCLUSIONS

ISCE 3.0 will form the core framework for NISAR production processing. At the same time, the code is being developed as a tool suite for scientists to develop their own scientific workflows with relatively light, recipe-driven development methods. With attention to efficiency in the compiled code-level elements, and an overarching framework that has been designed for scientific computing and scalability, the same code that functions as a "plug-in" to the production system can also be provided to individual science users. In this way, the scientist can compare their results directly to the official products distributed by NASA to build confidence in their use for custom processing.

8. ACKNOWLEDGMENTS

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA.

9. REFERENCES

[1] Rosen, P. A., S. Hensley, and G. Peltzer (2004), Updated Repeat Orbit Interferometry Package released, Eos Trans. AGU, 85(5).

[2] Doin, M. P., Guillaso, S., Jolivet, R., Lasserre, C., Lodge, F., Ducret, G., & Grandin, R. (2011, September). Presentation of the small baseline NSBAS processing chain on a case example: the Etna deformation monitoring from 2003 to 2010 using Envisat data. In Proceedings of the Fringe Symposium (pp. 3434-3437). ES.

[3] Hooper, A., K. Spaans, D. Bekaert, M. Caro Cuenca, M. Arıkan, and A. Oyen. "StaMPS/MTI manual." Delft Institute of Earth Observation and Space Systems Delft University of Technology, Kluyverweg 1 (2010): 2629.

[4] Gurrola, E., G.-F. Sacco, P. A. Rosen, and H. Zebker (2010), InSAR Scientific Computing Environment. Earth Science Technology Forum.

[5] Zebker, H., S. Hensley, P. Shanker, C. Wortham (2010). Geodetically Accurate InSAR Data Processor. *IEEE Transactions on Geoscience and Remote Sensing*, 48(12).

[6] Rosen, P.A., E. Gurrola, G. F. Sacco and H. Zebker, "The InSAR scientific computing environment," *EUSAR 2012; 9th European Conference on Synthetic Aperture Radar*, Nuremberg, Germany, 2012, pp. 730-733.

[7] Gurrola, E., P. Agram, M. Lavalle, P. A. Rosen, G.- F. Sacco, (2016) The InSAR Scientific Computing Environment (ISCCE): An Earth Science SAR Processing Framework, Toolbox, and Foundry, AGU Fall Conference.

[8] https://winsar.unavco.org/portal/wiki/WikiIndex/.

[9] Rosen P., *et al*. (2016) "An update on the NASA-ISRO dual-frequency DBF SAR (NISAR) mission," *2016 IEEE Intl. Geosci. and Rem. Sens. Symp. (IGARSS)*, Beijing, 2016, pp. 2106-2108. doi: 10.1109/IGARSS.2016.7729543.

[10] http://nisar.jpl.nasa.gov.

[11] Behnel, S., R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn and K. Smith, "Cython: The Best of Both Worlds," in *Computing in Science & Engineering*, vol. 13, no. 2, pp. 31-39, March-April 2011. doi: 10.1109/MCSE.2010.118.

[12] Aivazis, M., (2017) "Pyre bzr respository," http://pyre.orthologue.com.

[13] Fattahi, Heresh, Piyush Agram, and Mark Simons (2017) "A network-based enhanced spectral diversity approach for TOPS time-series analysis." IEEE Transactions on Geoscience and Remote Sensing 55, no. 2: 777-786.

[14] Fattahi, Heresh, Mark Simons, and Piyush Agram (2017) "InSAR Time-Series Estimation of the Ionospheric Phase Delay: An Extension of the Split Range-Spectrum Technique." IEEE Transactions on Geoscience and Remote Sensing 55, no. 10, 5984-5996.

[15] Lavalle, M., G. H. X Shiroma, P. Agram, E. Gurrola, G. Sacco, and P. A. Rosen (2016) PLANT: Polarimetric-interferometric lab and analysis tools for ecosystem and land-cover science and applications, in *2016 IEEE IGARSS*, pages 5354–5357, July.

[16] Agram, P. S., R. Jolivet, B. Riel, Y. N. Lin, M. Simons, E. Hetland, M.-P. Doin, and C. Lasserre (2013), New Radar Interferometric Time Series Analysis Toolbox Released, Eos Trans. AGU, 94(7), 69.