ISCE: A modular library

ISCE Developer team
Jet Propulsion Laboratory

Jun 29, 2015
@UNAVCO
What is ISCE?

- Python-based SAR/InSAR processing software
  - Modular library
  - No commercial software needed
  - Includes some pre-defined workflows

What are we going to talk about?
- ISCE directory structure
- Available functionality
- Newer functions not available in ROI_PAC / other software
Directory structure

- **ISCE Components**
  - Iscesys (Framework elements)
  - Isceobj (Data containers and building blocks)
  - Mroipac (Functionality from ROI_PAC)
  - Stdproc (stdproc library from Stanford University)
  - Contrib (User supplied functionality)
    - Snaphu
    - DEM utilities etcs

- **ISCE Applications**
  - Apps (Pre-defined workflows)
  - Utilities (Assisting users)
Iscesys

• Framework elements of ISCE
• Allows for configurable workflows
  – Defines Components and Applications

• Components
  – Any unit (data / processing) in ISCE

• Applications
  – Pre-defined workflows in ISCE

• Includes low level ImageAPI for handing images

• Typically, not directly used by users
• Short for ISCE objects

• Useful containers and building blocks for SAR/InSAR processing

• Includes most operations that don’t need geometry information
  - Polynomial fitting
  - Orbit interpolation
  - Offset outliers estimation etc
Data Containers in ISCE

- **Image**
  - Wrapper class for dealing with images

Loading an ISCE image

```python
>> import isceobj
>> img = isceobj.createImage()
>> img.load('data.xml')
```

```plaintext
Img.width, img.length, img.dataType, img.bands,
img.scheme, img.imageType
```
Data Containers ....

- Orbit
  - StateVector
    - Time, position, velocity
  - self.interpolateOrbit(time, method='hermite')
  - self.addStateVector(sv)
  - self.unpackOrbit()
  - self.pointonground(azimuthtime, range)
  - self.geo2rdr([lat,lon,hgt])
  - self.exportToC()

- Planet
  - Ellipsoid representation (Default WGS84)
Data Containers

• OffsetField
  – Offset
  – self.plot()
  – self.getFitPolynomials()
  – self.cull(snrthreshold)

• Platform
  – Antenna length, pointing direction
  – Planet

• Radar
  – PRF, wavelength, bias, sampling frequency, chirp slope, pulse length,
Frame Data Container

• Frame (Represents a SAR acquisition)
  – Orbit
  – Radar
    • Platform
  – Image

• Keeping track of the Frame object will let one keep track of most variables used during processing
Polynomials in ISCE

• Polynomials
  – Doppler polynomials
  – Offset polynomials
  – FM rate polynomials
  – Range / azimuth carriers

• isceobj.Util.Poly1D / Poly2D
  – Simple list of coefficients
  – Mean and norm applied in each direction
  – Include exportToC functionality
  – Can make a polynomial looks like an image
  – Used extensively in all processing modules
Isceobj processing modules

- **Isceobj.Util.estimateoffsets**
  - Default offset estimation in insarApp.py
  - Parallel FFT-based ampcor-like algorithm from Stanford University
  - Returns a list of offsets

- **Isceobj.Util.denseoffsets**
  - Dense offset estimation module
  - Parallel FFT-based algorithm from Stanford University
  - Returns offset images
Isceobj modules

• Isceobj.util.simamplitude
  – Simulate radar amplitude image from outputs of topo

• Isceobj.util.offoutliers
  – Identifies outliers in the offset field by fitting an affine transform

• Isceobj.util.ImageUtil
  – Library to interface ISCE products to numpy
  – Used extensively by imageMath.py

• Isceobj.XmlUtil
  – XML reading and writing utilities
• Functionality that was available in ROI_PAC
  – Bugs fixed and improved

• Ampcor
  – Classic amplitude correlation
  – Time domain convolution

• Variants of ampcor
  – DenseAmpcor (similar to denseoffsets for speckle tracking)
  – Nstage (multi-stage image match for data with bad orbits)
Mroipac modules

• Baseline
  - Approximate baseline computation from ROI_PAC

• Filter
  - Adaptive filtering of interferograms

• Dopiq
  - Doppler Centroid estimation from RAW data

• Grass
  - Residue-cut phase unwrapping algorithm
Mroipac modules

- **Icu**
  - Phase unwrapping module from SRTM

- **Fitoff**
  - Return offset fit polynomials
  - Only works up to order 3

- **Correlation**
  - Coherence estimation from amplitude files and interferogram
Stdproc

• Modules developed by Howard Zebker @ Stanford University

• insarApp.py built primarily on stdproc

• Stdproc.stdproc.formslc
  – SAR focusing module
  – Range doppler approach

• Stdproc.stdproc.estamb
  – Ambiguity estimator
  – Not really needed for modern sensors
• **Stdproc.stdproc.mocompTSX**
  - Resample SLC data delivered by other missions to an ideal mocomp geometry
  - Not accurate as no topography is taken into account

• **Stdproc.stdproc.resamp**
  - Resamples slave and cross multiplies with master
  - Simple Prati filter based on polynomial offsets
  - Sinc interpolation, can optionally flatten in range

• **Stdproc.stdproc.resamp_slc (JPL)**
  - Generalized SLC resampling
  - Polynomials + pixel-by-pixel offset files
  - Multiple interpolation methods – sinc, nearest, bilinear
• Stdproc.orbit.fdmocomp
  - Adjust doppler centroid for mocomp processing
  - Accounts for $V_h$ and reduces doppler to a function of squint only

• Stdproc.orbit.pulsetiming
  - Line-by-line orbit interpolation
  - Hermite polynomials (WGS84 system)

• Stdproc.orbit.orbit2sch
  - Transform WGS84 orbit to SCH system
  - Uses a peg point for transforming orbits
• **Stdproc.orbit.setmocomppath**
  - Given two orbits, return the best ideal mocomp orbit
  - Average of individual peg points

• **Stdproc.orbit.mocompbaseline**
  - 3 baselines computed and stored
  - Master vs ideal mocomp orbit
  - Slave vs ideal mocomp orbit
  - Master vs slave
• **Stdproc.stdproc.topo**
  - Simulate DEM in radar coordinates
  - Ideal mocomp orbit assumed
  - Pixel-by-pixel lat, lon, z
  - Line of sight file, los.rdr
  - Local incidence angle, inc.rdr
  - DEM interpolation – nearest, bilinear, bicubic, sinc, biquintic

• **Stdproc.stdproc.correct**
  - Simulate topophase with outputs of topo
  - Ideal mocomp orbit assumed
  - Line-by-line baselines from stdproc.orbit.mocompbaseline
• Stdproc.rectify.geocode
  – Generalized geocoding
  – Assumes ideal mocomp orbit
  – Multiple interpolation techniques
    • Nearest neighbor
    • Bilinear
    • Bicubic
    • Sinc
  – Pixel-by-pixel solutions
  – Parallelized and optimized for performance
• **Stdproc.model.enu2los**
  - Take 3 channel geocoded displacements and project it into radar LOS
  - Simple bilinear interpolation

• **Stdproc.model.zenith2los**
  - Take geocoded zenith wet delay and project it into radar LOS
  - Simple bilinear interpolation
Zerodop (JPL)

- Developed at JPL

- Based on algorithms published in literature over the last decade

- Handles traditional zero doppler and native doppler geometries accurately

- Works for ideal mocomp orbit as well as traditional focusing using the actual orbit
Zerodop.topozero

• Simulate DEM in radar coordinates

• DEM interpolation techniques
  – Nearest, bilinear, bicubic, sinc, biquintic

• Pixel-by-pixel solutions
  – Modified Newton-Raphson method

• Parallelized for performance

• LOS angles, Local incidence angles, Radar shadow and layover mask are optional outputs
Zerodop.geo2rdr

- Project any lat, lon, z data into radar coordinates
- Reverse geocoding
- Outputs offset files for direct use with resampling routines
- Optionally, also outputs slant range and azimuth time
• Generalized geocoding module

• Similar to geo2rdr, but produces geocoded output

• Multiple interpolation methods
  – Nearest, bilinear, bicubic, sinc
Hands-on Demo

• Using these modules together to build custom workflows

• Simple interferogram formation
  - Unpack preprocessed SLCS
  - Lat,lon,z for each pixel of master image
  - Geo2rdr using these lat,lon,z for slave image
  - Resample slave image using output of geo2rdr to generate coregistered SLC
  - Crossmultiply master and coregistered slave to generate interferogram