Single Interferogram Processing

(Geometric approach)

Piyush Agram
Jet Propulsion Laboratory

Aug 1, 2016
@UNAVCO

Thanks to my colleagues from JPL, Caltech, Stanford University and from all over the world for providing images and material for this talk.

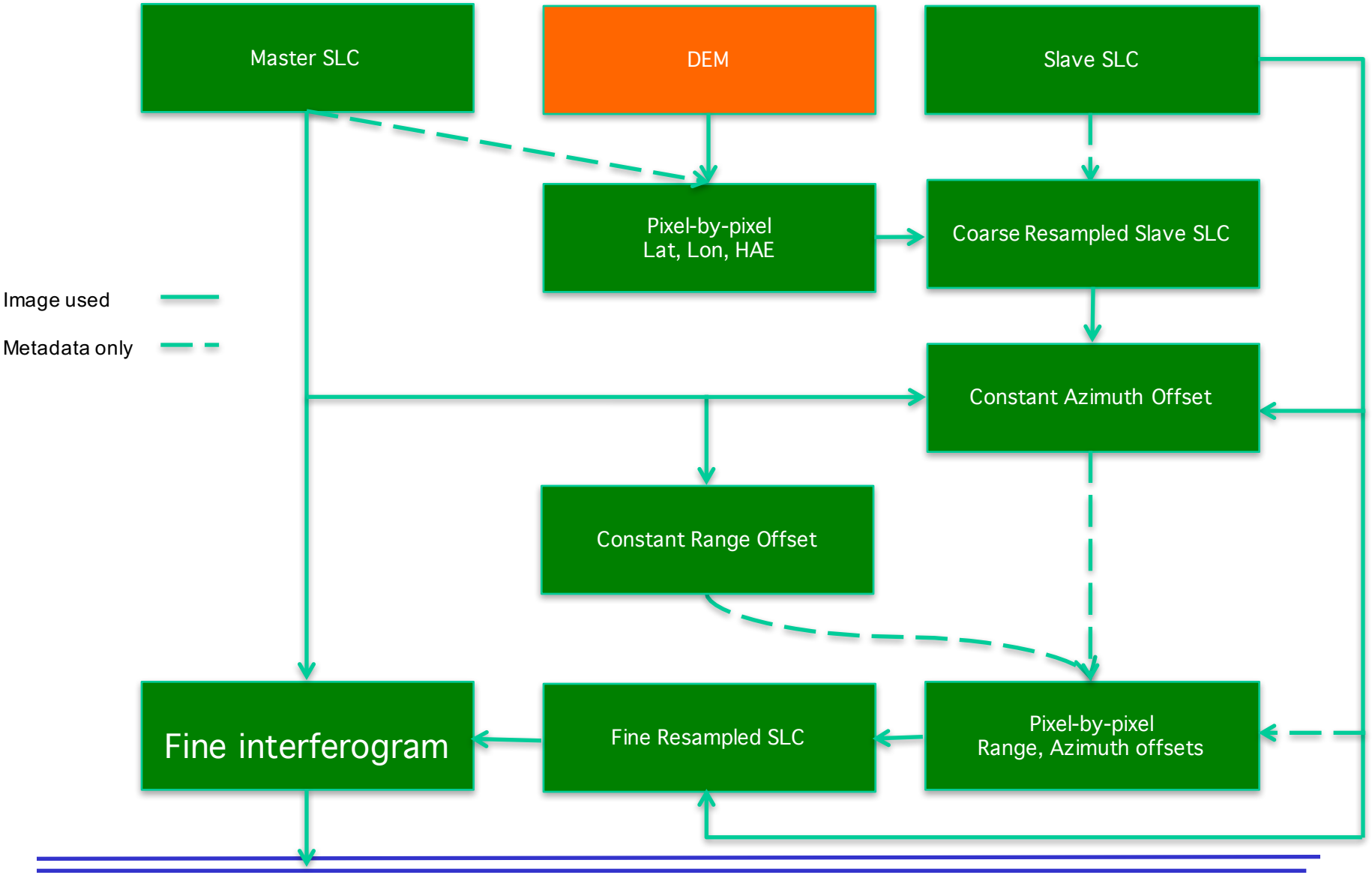
Overview

- Detailed look at geometric workflows
  - topsApp from ISCE
  - Preliminary support within DORIS

- For each processing step
  - Implementation details

- Disclaimer: Modules needed for geometric processing have been part of ISCE public release since May 2015. Currently, only topsApp.py uses these. These were demoed during UNAVCO workshop 2015.

Two-pass InSAR – geometric approach



Step 1: slcs/ formslc

- The most critical step in the workflow.
- Assumptions and parameters used during this stage, determines quality of end products
  - Resolution and bandwidths
  - Geometry system
  - Antenna patterns
- Geometry system determines the rest of the workflow
  - For pre-focused SLCs the rest of the workflow should match the geometry system used to generated the SLCs

Step 2: topo / topozero

- Creates pixel-by-pixel Lat, Lon, Hgt using Orbit and DEM
- Represents radar geometry for the master acquisition
- Depends on geometry system
  - Zero doppler / native doppler
  - Solves standard range-doppler equation
- Zerodop.topozero
  - Can handle both geometry systems
  - Orbits always in WGS84 ECEF system
- Simulated amplitude easily created from z.rdr

Step 3: Master Timing Error (Optional)

- If Master SLC time-tags don't match
  - Shift between simulated amplitude and master SLC
- Not needed for most modern sensors
  - Definitely needed for ERS/ Envisat/ Radarsat-1
- Estimate shift between simamp and SLC
  - Azimuth error corresponds to time-tag error in SLC metadata
  - Range error is less likely
- If master SLC timing is changed, go back and rerun step 2 (not needed for topsApp)

Step 4: Pixel-by-pixel offsets

- For every pixel (Lat,Lon, Hgt) in master image estimate its location in slave image
- Create pixel-by-pixel range and azimuth offset to be used for resampling
- Can inherently handle different PRFs, range sampling rates etc
  - E.g; no need for FBD2FBS etc
- Zerodop.geo2rdr
  - Can handle both native doppler / zero doppler

Step 5: Coarse Resampling slave SLC

- Stdproc.stdproc.resamp_slc
  - Can handle all modes of SAR data – stripmap, spotlight, SCANSAR, TOPS
  - Can handle doppler polynomial + azimuth carrier polynomial
  - Can handle traditional morph polynomial + pixel-by-pixel offsets

- Use pixel-by-pixel offsets from previous step to create coarsely resampled SLC

Step 6: Slave Timing error

- Estimate constant shift between master SLC and coarsely resampled SLC
- Constant azimuth shift represents differential timing error
- Constant range shift represents timing error + constant atmospheric delay
- Conditions under which constant shifts don't work – Ionosphere, fast motion like ice-sheets
  - Lets you isolate problems with specific acquisitions

Step 7: geo2rdr again

- Regenerate pixel-by-pixel offsets
- Apply constant shift in azimuth to sensingStart
- Apply constant range shift to starting Range

Step 8: Finely resampled SLC

- Use updated offset fields to regenerate coregistered SLC
- This coregistered SLC can be reused for other pairs in the stack as well
  - Not possible with the traditional approach

Step 9: Filtering of SLCs (optional)

- Now that you have the master SLC and coregistered SLC
  - Filter for common azimuth bandwidth
  - Apply range spectral filter

- This steps depends on the application
  - For PS applications, no filtering should be applied
  - For SBAS applications, if you want to track DEM error no filtering should be applied

Step 10: Crossmul and flattening

- Crossmultiply the SLCs and use the range offsets for flattening
  - Offsets used for resampling are exactly the same information needed for flattening
  - True pixel-by-pixel baselines
- Consistent use of image metadata
  - Not the case in traditional approach
  - Same metadata gets used for generating interferograms from given SAR scene in a stack as we generate coregistered SLCs first
- Rest of the processing as usual – coherence, filtering, unwrapping, geocoding etc