1. Stack Processing for GIAnT

In the previous lab sessions, we learned to use insarApp.py to generate individual interferograms. In this lab, we will explore the possibility of processing interferogram stacks i.e, multiple interferograms that are co-located on a common image grid using ISCE. The process of resampling multiple interferograms to a common grid is called “stacking”.

“Stacking” can be performed in radar image coordinates (range and azimuth axes) or in geo coordinates (lat and lon axes). In this lab session, we demonstrate a simple stacking approach in geo-coordinates. To generate our interferograms stacks, we will combine the following features of ISCE that were discussed in the previous sessions:

1. Using insarApp.py with the --steps option
2. Modifying the input XML files to customize the output geocoded grid
3. Modifying the input XML files to customize the set of geocoded output products

In this lab session, we explain the various steps involved in generating stacks with ISCE. We will not be processing any actual interferogram stacks from scratch due to space and time restrictions, but will walk through the various steps in sequence. No advanced Python programming skills are assumed for this tutorial. Users can implement these steps with the scripting language of their choice - Python, bash etc. Future versions of ISCE will include an application called isceApp.py which will automate the processing of intereferogram stacks.

2. Organizing the data

We demonstrate our approach to stack processing using an hypothetical COSMO SkyMed dataset (Note again that we won't be processing any real data). We organize the COSMO-SkyMed data corresponding to our stack in the following directory structure:

```
lab8
|-- dem   (Directory for storing DEM)
|   |-- demLat_N36_N38_Lon_W123_W121.dem
|   `-- demLat_N36_N38_Lon_W123_W121.dem.xml
|-- raw      (Directory for storing sensor data)
|   |-- 20130531 (Directory for data acquired on a particular date)
|   |-- 20130616
|   |-- 20130702
|   |-- 20130718
|   |-- 20130803
|   |-- 20130819
|   `-- 20130904
`-- insar   (Directory to store the processed interferograms)
```

Each of the date directories could again contain multiple files corresponding to consecutive frames. The XML configuration files shown in this walk through will automatically be combine the multiple frames to create a single raw image file. An example date directory is shown below:

```
20130616/
|-- CSKS1_RAW_B_HI_09_HH_RA_SF_20130616135654_20130616135701.h5
|-- CSKS1_RAW_B_HI_09_HH_RA_SF_20130616135659_20130616135706.h5
`-- CSKS1_RAW_B_HI_09_HH_RA_SF_20130616135704_20130616135710.h5
```

3. Determining viable interferogram pairs

In this section, we demonstrate the simplest method of generating the baseline plot for our stack of acquisitions. Choose one particular date (say the earliest acquisition as reference). For our hypothetical example, that would be “20030531”.

Step 1:

In the insar directory create directories named 20030531_date2, where date2 corresponds to all the other acquisition dates in the stack.

Step 2:

Populate each of the InSAR pair directories with the minimal insarApp.xml and SAR catalog files following instructions in Lab 6.

The insarApp.xml file would look like:

```
<insarApp>
    <component name="insar">
        <component name="Dem">
<catalog>../dem/demLat_N36_N38_Lon_W123_W121.dem.xml</catalog>
        </component>
        <component name="slave">
            <catalog>20130531.xml</catalog>
        </component>
        <property name="Sensor name">
            <value>COSMO_SKYMED</value>
        </property>
        <component name="master">
            <catalog>20130616.xml</catalog>
        </component>
    </component>
</insarApp>
```

SAR catalog file for the 20130531 acquisition would look like:

```
<component name="sar">
    <property name="OUTPUT">
        <value>20130531.raw</value>
    </property>
    <property name="HDF5">
      <value>['../../h5/20130531/CSKS1_RAW_B_HI_01_HH_RD_SF_20130531021035_2013
0531021042.h5','../../h5/20130531/CSKS1_RAW_B_HI_01_HH_RD_SF_20130531021030_201
30531021037.h5','../../h5/20130531/CSKS1_RAW_B_HI_01_HH_RD_SF_20130531021040_20
```

```
130531021047.h5']</value>
    </property>
</component>
```

Step 3:

In each of the directories, run insarApp.py till the step named “preprocess”. This will add required baseline information to the insarProc.xml in each directory.

```
> insarApp.py --end=preprocess
> tail -n 12 insarProc.xml
    </slave>
    <baseline>
            <horizontal_baseline_top>96.7928771266</horizontal_baseline_top>
            <horizontal_baseline_rate>-1.73848950218e-05</horizontal_baseline_r
            ate>
            <horizontal_baseline_acc>8.52094426395e-11</horizontal_baseline_acc
      >
            <vertical_baseline_top>29.755105435</vertical_baseline_top>
            <vertical_baseline_rate>8.06177637777e-07</vertical_baseline_rate>
            <vertical_baseline_acc>-4.07016302367e-12</vertical_baseline_acc>
            <perp_baseline_top>-64.1586857667</perp_baseline_top>
            <perp_baseline_bottom>-63.4138138722</perp_baseline_bottom>
    </baseline>
</insarProc>
```

Use the perpendicular baseline information and acquisition dates to determine the interferogram pairs that you would like to process. Save the dates corresponding to the pairs that you want to process to a text file in the parent directory (“lab8”). Clear out the “insar” directory.

Note:
ISCE is a modular toolbox, and users familiar with Python programming can directly use ISCE modules to generate baseline plots. Use isce/components/isceobj/InsarProc/runPreprocessor.py as a template.

4. Determine common output grid

Before processing all the viable interferograms, we need to determine the common output grid. We will do so by processing one pair from the stack (typically a pair with short spatial and temporal baselines). To save time, we will turn off phase unwrapping and only geocode the wrapped interferogram. Lets say we process the pair 20130616_20130531 in the "insar/20130616_20130531" directory.

Modify the insarApp.xml file for the chosen pair to look like this:

```
<insarApp>
    <component name="insar">
        <component name="Dem">
<catalog>../dem/demLat_N36_N38_Lon_W123_W121.dem.xml</catalo
g>
        </component>
        <component name="slave">
            <catalog>20130531.xml</catalog>
        </component>
        <property name="unwrap">
            <value>False</value>
        </property>
        <property name="geocode list">
            <value>['filt_topophase.flat']</value>
        </property>
        <property name="Sensor name">
            <value>COSMO_SKYMED</value>
        </property>
        <component name="master">
            <catalog>20130616.xml</catalog>
        </component>
    </component>
</insarApp>
```

Now run insarApp.py in the example directory.

```
> insarApp.py --steps
```

To determine the bounding box of the processed interferogram, scroll down to the end of insarProc.xml

```
> tail -n 12 insarProc.xml
     <outputs>
     <GEO_WIDTH>3853</GEO_WIDTH>
```

```
            <MAXIMUM_GEO_LONGITUDE>-121.64999999999999</MAXIMUM_GEO_LONGITUDE>
       <MINIMUM_GEO_LONGITUDE>-122.72</MINIMUM_GEO_LONGITUDE>
       <LATITUDE_SPACING>-0.00027777777777777778</LATITUDE_SPACING>
       <GEO_LENGTH>4864</GEO_LENGTH>
       <LONGITUDE_SPACING>0.00027777777777777778</LONGITUDE_SPACING>
       <MAXIMUM_GEO_LATITUDE>36.899444444444434</MAXIMUM_GEO_LATITUDE>
       <MINIMUM_GEO_LATITUDE>38.250277777777775</MINIMUM_GEO_LATITUDE>
      </outputs>
   </runGeocode>
</insarProc>
```

We now have the information needed to set up the bounding box ([South, North, West, East]) in the insarApp.xml files. You may add a little padding around this estimated bounding box if needed. Clear out the “insar” directory.

5. Preparing directories for interferogram generation

In this step, we will set up the directories for all viable interferograms and the insarApp.xml files with correct values for generating stacks.

Step 1:

Create a new directory for each of the viable interferograms under the “insar” directory and create corresponding SAR catalog XML files as well in each sub directory.

Step2:

Determine common processing parameters. Setup insarApp.xml and SAR catalog XML files accordingly. Some of the commonly manipulated parameters for stack processing are:

Processing parameter	**Value**	**Description**
Doppler method	useDEFAULT	COSMO SkyMed metadata includes doppler information
Unwrap method	snaphu_mcf	Snaphu unwrapper using the MCF algorithm
Geocode bounding box	[36.85, 38.3, -122.75, -122.6]	SNWE determined from Step 4 above.
Geocode list	[‘filt_topophase.unw’, ‘phsig.cor’, ‘topophase.cor’, ‘los.rdr’]	Geocode only the files needed by GIAnT or for modeling.
Filter strength	0.5	Goldstein filter strength
Posting	20	Choose a posting comparable but less than DEM posting for best performance.

The corresponding insarApp.xml file looks like:

```
<insarApp>
    <component name="insar">
        <property name="posting">
            <value>20</value>
        </property>
         <property name="doppler method">
```

```
            <value>useDEFAULT</value>
        </property>
        <property name="unwrap">
            <value>True</value>
        </property>
        <property name="unwrapper name">
            <value>snaphu_mcf</value>
        </property>
         <property name="geocode bounding box">
              <value>[36.85,38.3,-122.75,-122.6]</value>
         </property>
         <property name="geocode list">
              <value>['filt_topophase.unw', 'phsig.cor',
'topophase.cor','los.rdr']</value>
         </property>
         <property name="filter strength">
             <value>0.5</value>
        </property>
        <property name="Sensor name">
            <value>COSMO_SKYMED</value>
        </property>
        <component name="master">
            <catalog>20130616.xml</catalog>
        </component>
        <component name="slave">
            <catalog>20130531.xml</catalog>
        </component>
        <component name="Dem">
            <catalog>../../dem/demLat_N36_N38_Lon_W123_W121.dem.xml</catalog>
        </component>
    </component>
</insarApp>
```

Each interferogram directory is now ready to support an independent “insarApp.py” processing run.

Step 3:

Run “insarApp.py” in individual directories and the output geocoded products are ready to be directly ingested into GIAnT. The presented approach has following advantages:

1. Allows for customized processing of individual interferograms. Each interferogram processing is localized to a single directory. Add all configuration XML files to the processing directory. Can modify individual insarApp.xml files.
2. Independent processing of interferograms. Once the structure of the input XML files are known, interferograms can be processed independently on different machines / networks/ cloud instances.