1. Processing COSMO-SkyMed data from raw data files

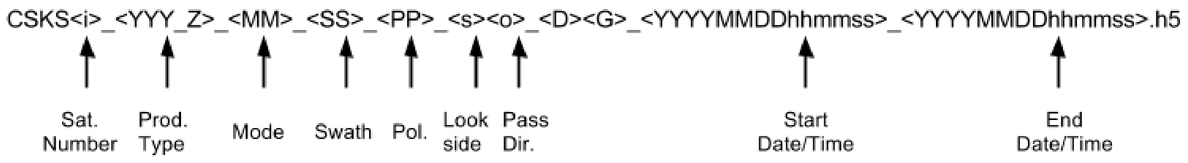
In this lab, you will learn how to process COSMO-SkyMed data from raw data files, while exploring some of the configurability capabilities of ISCE. As we've seen in previous labs, InsarApp.py is set up with sensor-dependent default parameters for all the processing steps. For many data sets, all the user needs to do is set up the input data file names in the master and slave XML files and update insarApp.py's control file (typically named insar.xml) to reference them. Then the command

```
insarApp.py insar.xml
```

will process the data from raw to geocoded interferograms. There are times however when the default parameters are not optimal. We saw a simple example of this in lab 3.3 where the default settings did not unwrap the interferogram. We could set the unwrap flag in the control file to cause unwrapping to occur.

In this lab we will demonstrate how to affect control parameters through configurability files associated with individual processing components. The README file at the top level of the ISCE distribution describes the details of the configurability options and hierarchy of where ISCE looks to find parameters it needs to process. Here we will first process a patch of data to completion to allow us to see just the beginning of the file. We will examine the outputs and decide that we want to change a particular processing option, create a configuration file for the appropriate component, and reprocess with that new option. We will then look at the output again to see how things have changed.

2. Understanding CSK Data Set Names


CSKS<i>_<YYY_Z>_<MM>_<SS>_<PP>_<s><o>_<D><G>_<YYYYMMDDhhmmss>_<YYYYMMDDhhmmss>.h5
Sat. Number
Prod. Type
Mode
Swath
Pol.
Look side
Pass Dir.
Start Date/Time
End Date/Time

The graphic above shows the standard naming conventions for COSMO-SkyMed (CSK) data files. CSK data comes with all of the data and metadata, including the orbit, in a single HDF5 (.h5) file. Sometimes the data is delivered with some additional files, but these are not used in the ISCE processing.

3. How to insert CSK filenames into the ISCE xml input files

The file names can be inserted into a master and slave component through configuration files as shown previously:

```
> cd
> cd data
> cd lab6
> cat Master.xml
<component name="Master">
    <property name="HDF5">
     <value>data/CSKS4_RAW_B_HI_08_HH_RA_SF_20110327162502_201103271
62510.h5</value>
    </property>
    <property name="OUTPUT">
        <value>20130327.raw</value>
    </property>
</component>

> cat Slave.xml
<component name="Slave">
    <property name="HDF5">
          <value>data/CSKS4_RAW_B_HI_08_HH_RA_FF_20110311162513_2011
     0311162521.h5</value>
    </property>
    <property name="OUTPUT">
        <value>20061231.raw</value>
    </property>
</component>
```

Note that in this case, the `h5` data files reside in a subdirectory called `data`, so the filename is prepended with the relative path `data/`. These lines can also be directly inserted into the insarApp input file. For this lab, the input file is named `insar_130327_130311.xml`. If you print that file to the screen, you will see these lines verbatim inline in the file.

```
> cat insar_130327_130311.xml
```

(shows the same as above text in Master.xml and Slave.xml)

4. Running a patch of data using component configurability and examining the results

Let's start with an input file that has some particular values set : `insar_130327_130311.xml`. In this input file, the only specified parameters are the posting of the output grid and the master and slave data files, and of course the sensor name. All other parameters are set to their defaults that are deemed appropriate for this sensor.

```
> cat insar_130327_130311.xml
<insarApp>
    <component name="insar">
        <property name="posting">
            <value>20</value>
        </property>
        <property name="Sensor Name">
            <value>COSMO_SKYMED</value>
        </property>
        <component name="master">
            <property name="HDF5">
<value>data/CSKS4_RAW_B_HI_08_HH_RA_SF_20110327162502_20110327162510.
h5</value>
            </property>
            <property name="OUTPUT">
                <value>20130327.raw</value>
            </property>
        </component>
        <component name="slave">
            <property name="HDF5">
<value>data/CSKS4_RAW_B_HI_08_HH_RA_FF_20110311162513_20110311162521.
h5</value>
            </property>
            <property name="OUTPUT">
                <value>20130311.raw</value>
            </property>
        </component>
    </component>
</insarApp>
```

Sometimes it is convenient to process only the beginning portion of data set the first time through to get a feel for the data - examine its correlation and fringe quality, check the focus of the image, etc. There are several ways to accomplish this; we saw in Lab 3 that the number of patches can be set in the input file directly. But suppose we don't want to or do not have permission to alter the input file, or we want to apply the same parameter modifications to a number of interferogram processing runs. We can take advantage of component configurability to accomplish the same thing.

The ISCE architecture has mechanisms to allow each parameter in a workflow component in `insarApp.py` to be configurable. This feature was added recently so not all components are yet configurable, but certain key components are. The image formation component, known as `formslc,` is one such configurable component. `formslc` is a patch-based focusing system, performing convolution through FFT-based circular convolution one chunk of pulses at a time. This is the module where specifying a single patch for quick examination takes place. To set parameters specific to `formslc` in a configuration file, we create a file called `formslc.xml` and place it in the directory where we are running `insarApp.py`. In this case, formslc is the f*amily name* of the what would be a possible set of image formation modules. Thus the name *formslc* in `formslc.xml` refers to the family name. Let's set the parameter for the number of patches, called `NUMBER_PATCHES`, to 1 in `formslc.xml`:

```
> cat > formslc.xml
<dummy>
<component name="formslc">
    <property name="NUMBER_PATCHES">1</property>
</component>
</dummy>

Press Ctrl-D to exit cat
```

For configuration files, the bounding xml construct we've seen before in `insar.xml` is not needed, so we see above the use of `<dummy>.` This could be any string.

Now let's run one patch, with `formslc.xml` present in the current working directory.

```
> insarApp.py insar_130327_130311.xml --steps
```

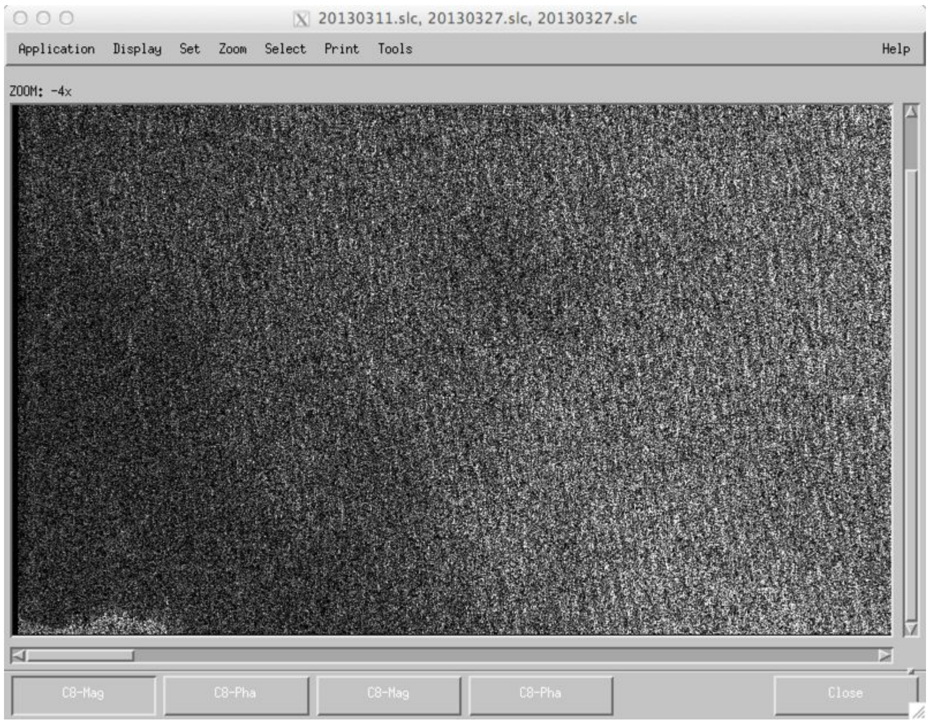
After waiting a while, you will find that the run ends in an error at the end of a long traceback:

```
"/Users/parosen/Applications/Installs/isce_py33/isce/components/isceobj/Util/EstimateOffsets.p
y", line 347, in checkImageLimits
    raise ValueError('Too small a reference image in the height direction')
ValueError: Too small a reference image in the height direction
```

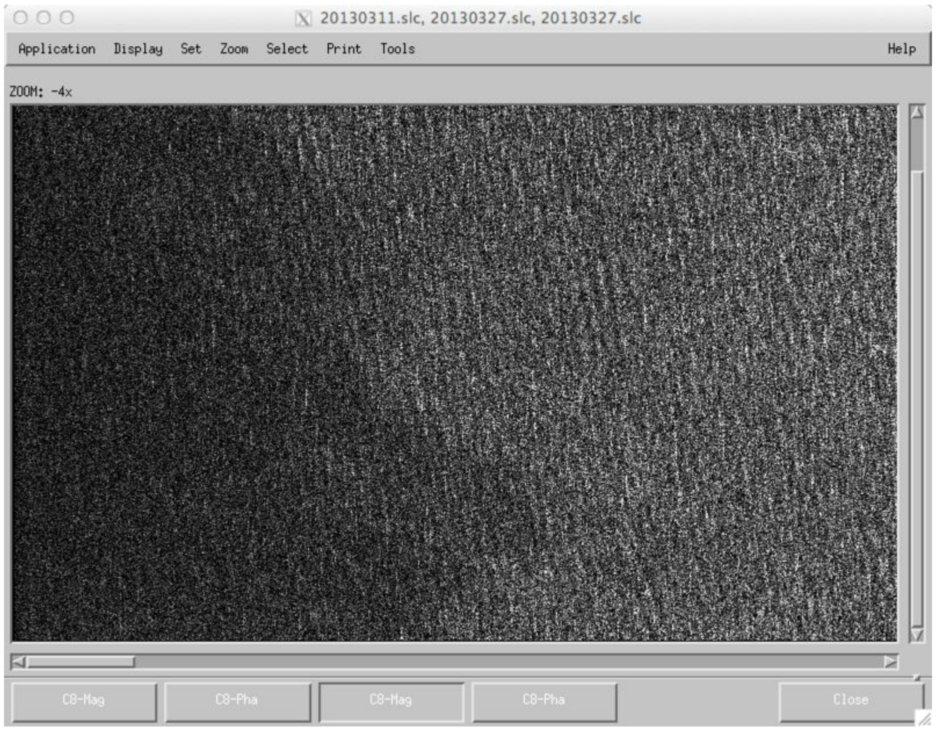
Is there some fundamental problem with the data set? Typically at this point, it is good to look at the images (slcs) to see if there is an issue.

```
> mdx.py *.slc
```

shows something interesting. If you zoom out by a factor of 4, then scroll to the bottom left, then click on C8-Mag left most button, you see the amplitude of the master image, as follows.



If you do the same but click on the other C8-Mag button, you see the amplitude of the slave image.



These images look similarly non-descript except for a small feature in the bottom left corner of the master image. This is a piece of land from Hawaii, and the rest of the image is just water. So in this particular case, processing 1 patch was not particularly useful because most of the image is water in this area, and clearly the ability to estimate offsets, both between images and between interferogram and topography, is compromised with only a patch of ocean coverage. So we should really try more patches.

5. Running more patches of data using component configurability

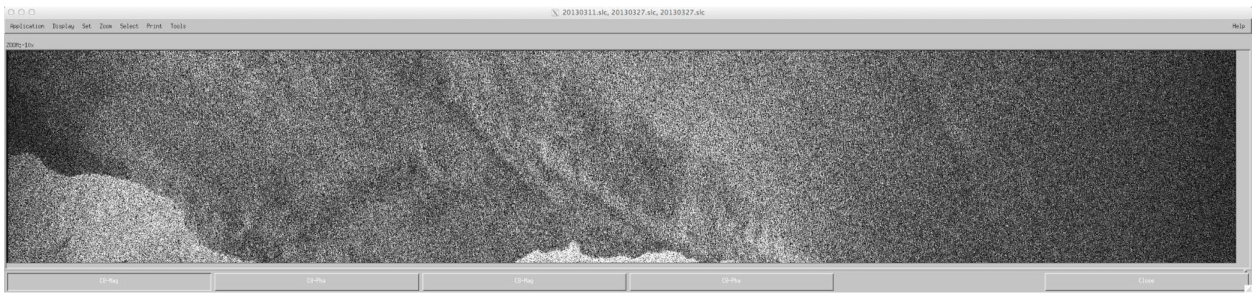
Let’s just try 2. Modify `formslc.xml` to set `NUMBER_PATCHES` to 2 and run it again.

```
> insarApp.py insar_130327_130311.xml --start=formslc
.
.
.
  File
"/Users/parosen/Applications/Installs/isce_py33/isce/components/isceobj/InsarProc/runOffoutlie
rs.py", line 60, in runOffoutliers
    raise Exception('Offset estimation Failed.')
Exception: Offset estimation Failed.
```

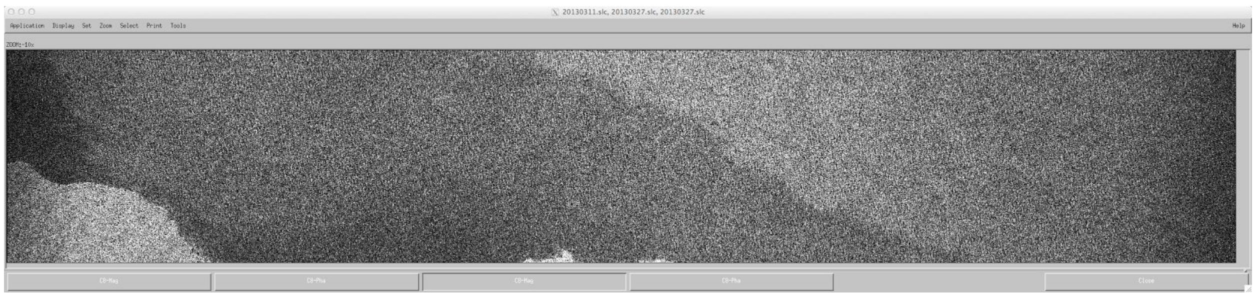
We’ve failed again, with a different error message! What could have gone wrong this time?

```
> mdx.py *.slc
```

This time, zoom out by a factor of 10 and stretch the window to be able to see the entire processed region. (If your screen is too small to do this, try zooming out even more until it fits.) The master looks as follows:


20130311.slc, 20130327.slc, 20130327.slc
Application Display Set Zoom Select Print Tools
Help
C8-Mag
C8-Pha
C8-Mag
C8-Pha
Close

And the slave as so:


20130311.slc, 20130327.slc, 20130327.slc
Application Display Set Zoom Select Print Tools
Help
C8-Mag
C8-Pha
C8-Mag
C8-Pha
Close

The bright region in the lower left is land, The other brightness features are water backscatter which varies from time to time and does not correlate. So clearly in this case, there are not enough points of common correlation over land to estimate the alignment of the data, so the offset estimation failed.

This is a piece of land from Hawaii, and the rest of the image is just water. So in this particular case, processing 1 patch was not particularly useful because most of the image is water in this area, and clearly the ability to estimate offsets, both between images and between interferogram and topography, is compromised with only a patch of ocean coverage. So we should really try more patches.

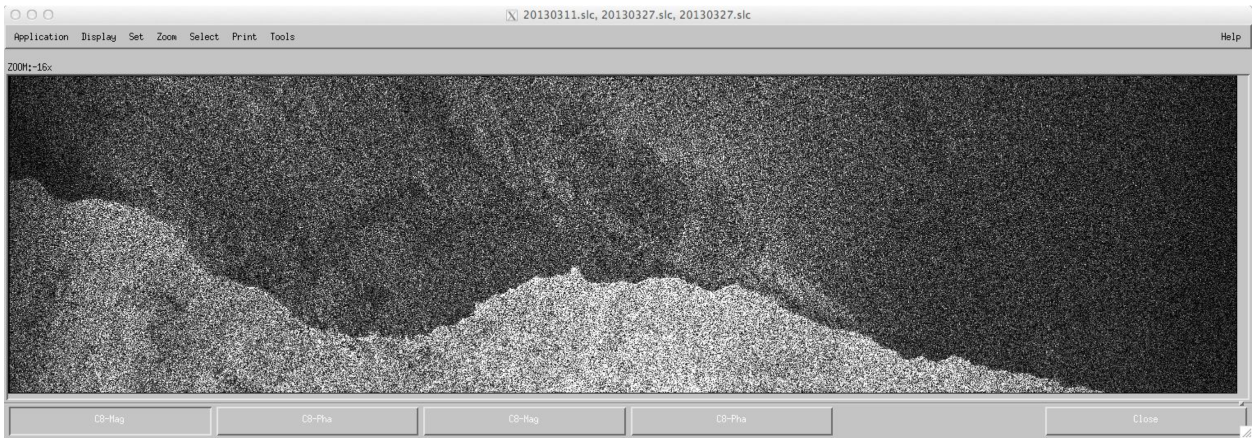
So how do we get this to work better? It looks like the data are good, but we are confounded by geography relative to our rectangular window into the world dictated by the radar imaging process. One option is to process more patches to get more land. Another is move the starting location to process the data. Another yet is to increase the size of each patch to cover more real estate with each step.

6. Running with a larger patch size using component configurability

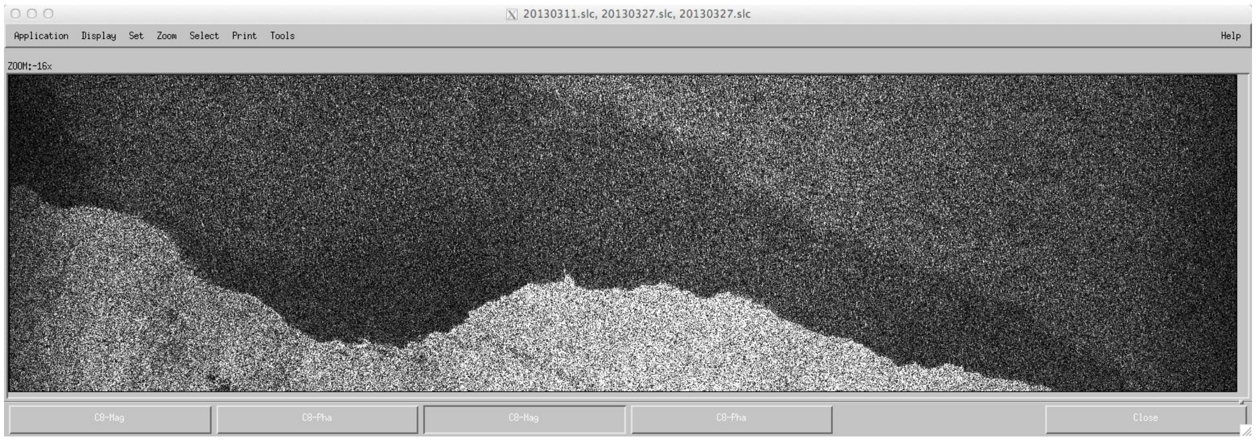
Let's try the last option, specifying 1 patch but a larger patch size. The default setting for patch size saves 2048 pulses (look at the mdx window and scroll to the bottom, then click on a pixel to see how many lines there are in the file, or look at the metadata for the slc. For 2 patches, the slc's are 4096 lines, so each patch is 2048 lines). Let's try a patch size of 8192 pulses `formslc.xml` should now look like:

```
<dummy>
<component name="formslc">
    <property name="NUMBER_PATCHES">1</property>
    <property name="AZIMUTH_PATCH_SIZE">8192</property>
</component>
</dummy>
```

We've increased the size of the patch by a factor of 4, so we should see more land. Indeed, looking at the slcs, we see for the master:



and for the slave:

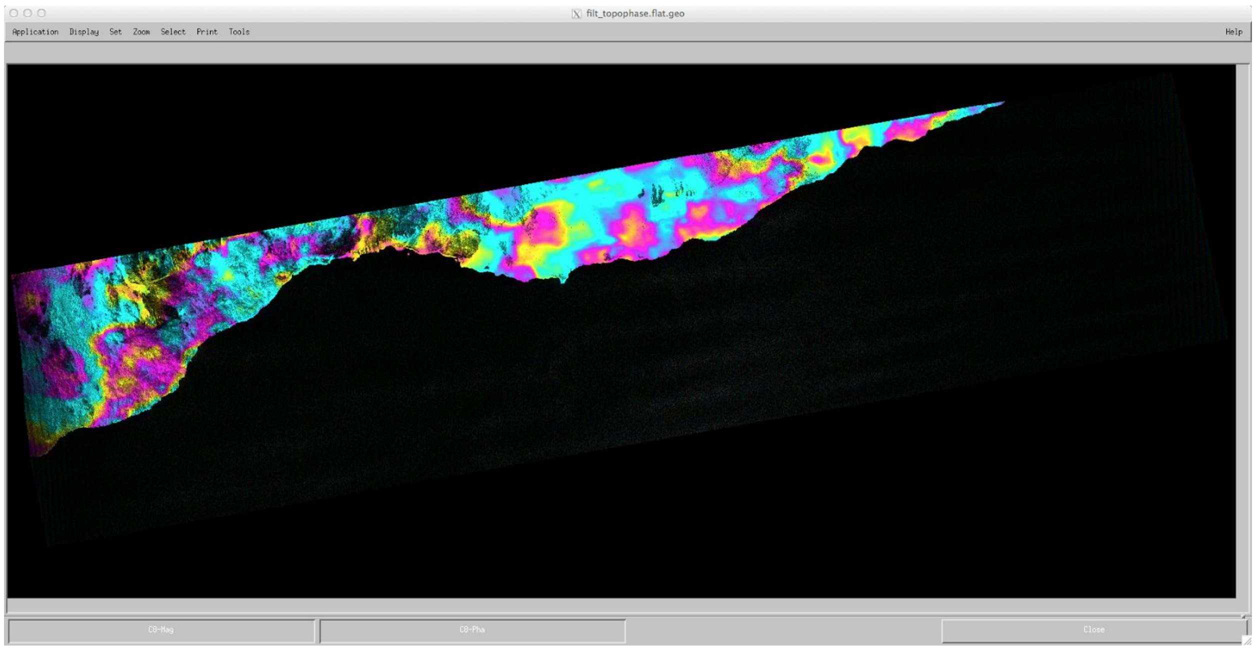


Note also that the run processed all the way through to the end, as evidenced by the screen output:

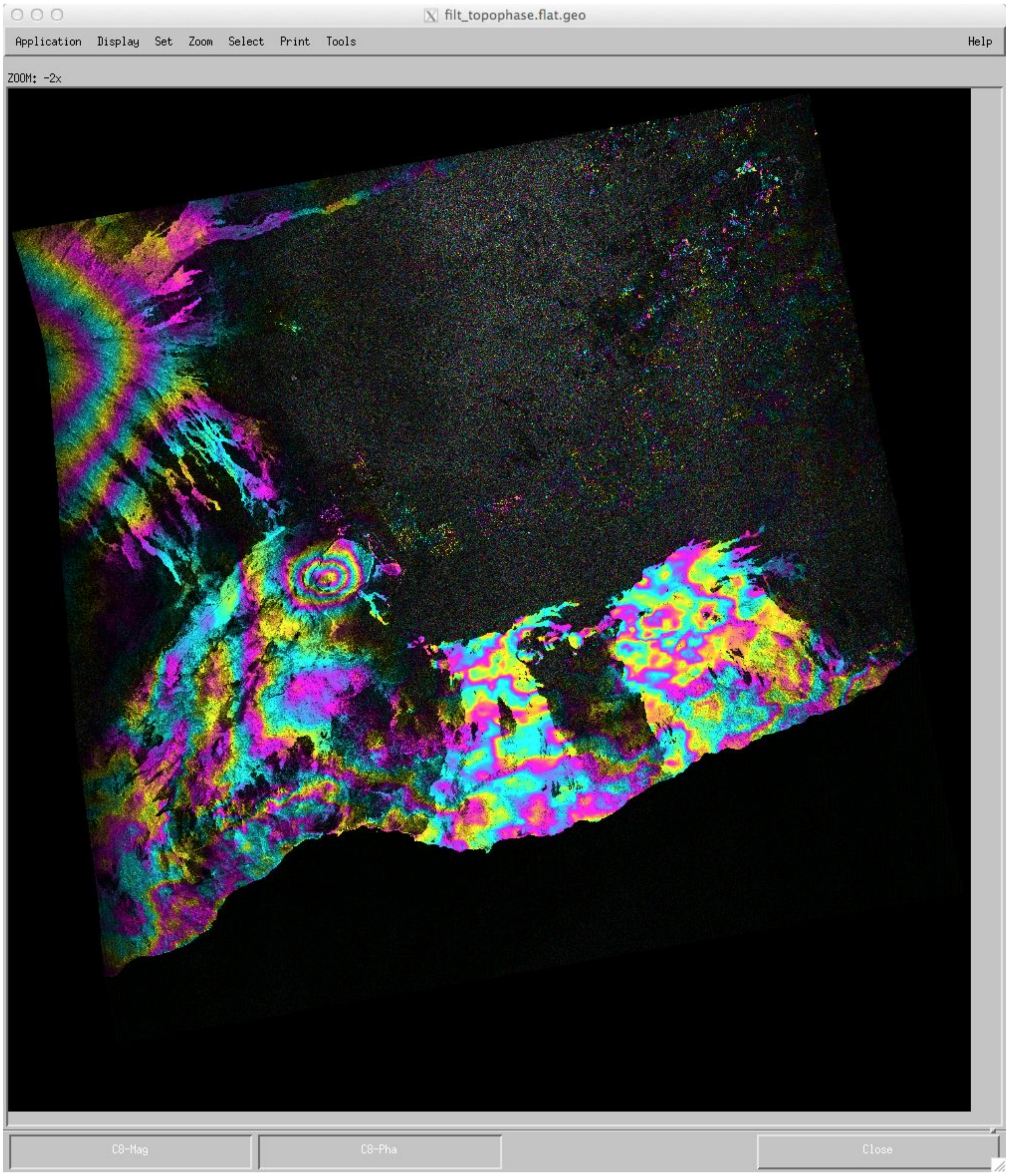
```
     runGeocode - Outputs
------------------------------------------------------------------------------------------
------
runGeocode.outputs.LATITUDE_SPACING = -0.0002777777777777778
runGeocode.outputs.GEO_WIDTH = 1770
runGeocode.outputs.MAXIMUM_GEO_LATITUDE = 19.130277777777778
runGeocode.outputs.MINIMUM_GEO_LATITUDE = 19.343333333333334
runGeocode.outputs.LONGITUDE_SPACING = 0.0002777777777777778
runGeocode.outputs.MAXIMUM_GEO_LONGITUDE = -154.92333333333332
runGeocode.outputs.GEO_LENGTH = 768
runGeocode.outputs.MINIMUM_GEO_LONGITUDE = -155.41472222222222
##########################################################################################
######
2014-07-28 12:26:34,690 - isce.insar - INFO - Total Time: 214 seconds
```

It is instructive to look at the final decoded interferogram `filt_topophase.flat.geo`. This is a geocoded interferogram with the topography removed and a smoothing filter applied to reduce the phase noise. Even with a fairly narrow sliver of land, the end-to-end processing works quite well, as seen in this screen shot of this image

```
> mdx.py filt_topophase.flat.geo
```



Encouraged by this, we can process the entire image by deleting the `formslc.xml` entirely, or if we want to keep the larger azimuth patch (which is more efficient in general), we can just delete the `NUMBER_PATCHES` property in the file. The result is the image below. Note the bulls-eye around Kilauea crater and the atmosphere-related fringes in the top right over Mauna Loa. Note also the decorrelation at X-band in COSMO-SkyMed data is high over vegetation, so there are few fringes visible north east in the forested areas.


filt_topophase.flat.geo
Application
Display
Set
Zoom
Select
Print
Tools
Help
ZOOM: -2x
C8-Mag
C8-Pha
Close

7. Notes about component configurability

This lab has used a new sensor type - COSMO-SkyMed - to illustrate a few of the configurability options available to users. Configurability is a potentially powerful way to control the behavior of the workflow components. However, users should be aware of some of the limitations and features that comes along with configurability, particularly at this early stage of development.

1. Some modules may not have not been architected to be configurable. Not all are expected to need it, so the developers have focused on those that are most commonly configured.
2. For those that have been architected appropriately, all nominal input parameters that are computed by the control scripts and passed into a compute module are made configurable, but some don't make sense to change. For example, you would not want to change the radar wavelength under any normal circumstance (though it there are cases where it might make sense!) or any of the other radar specific parameters.
3. Some parameters that you might want to change have interactions with other parts of the workflow, and though you would expect them to work, they don't. For example, in the example above, changing the starting pulse actually does not work robustly (try it!). It does actually process the data starting at the specified location, but things break further downstream because other parameters were set up outside `formslc` that assumed starting at the beginning. This needs to be improved in subsequent iterations of configurability.
4. It is non-trivial to discover the names of the configurable parameters if you are not a developer. There are a lot of them, and the developers have not yet documented them all.
5. Some parameters can be specified in the input file as well as in a configuration file. They do not have the same names, which is confusing. This will be improved in subsequent releases.