Using "pixelTrack.py" to Generate Offsets from Radar/Optical Image Pairs

Andrew K. Melkonian (akm26@cornell.edu)
M.S. Student, Cornell University

September 4, 2009

Note: **Beta** version.

RADAR: Full SAR pixel-tracking processing chain.

Optical: Reads ASTER HDF-EOS file and writes specified band as flat file of 32-bit floats, can also apply 5x5 high-pass gaussian filter to data.

1 Introduction

1.1 Purpose

This python script is intended to take overlapping scene pairs of the rawest of the raw SAR or ASTER data and create pixel offsets using ROIPAC/ampcor.

1.2 Required Software

Standard UNIX/Linux C and Fortran compilers and tools (gcc 4.1.2 or later required)

ROIPAC

Download from http://www.roipac.org/installation

Python 2.4 or later, Python 2.5 or later if reading ASTER HDF files

Download from: http://www.python.org/download/

Pylab/matplotlib: http://matplotlib.sourceforge.net/

Download from: http://sourceforge.net/projects/matplotlib/

NumPy: http://numpy.scipy.org/

Download from: http://sourceforge.net/projects/numpy/files/NumPy/

SciPy: http://www.scipy.org/

Download from: http://sourceforge.net/projects/scipy/files/

Please read documentation accompanying SciPy for information on downloading and installing ATLAS, BLAS, and LAPACK libraries. These are necessary for SciPy to function effectively.

OPTICAL ONLY: pyhdf

Please review requirements for installing pyhdf, requires szip, zlib, and HDF4 libraries in addition to Python 2.5 and the numpy module.

Download from: http://pysclint.sourceforge.net/pyhdf/

1.3 Setup

Download "pixelTrack.tar" and untar it using the command "tar -xvf pixelTrack.tar".

This will create the folder "PixelTracking", containing "pixelTrack.py", these instructions, and a folder called "Python" containing "generic_load_make_azo.py".

1.4 Run

Create a parameter file (see **Section 2**)

"cd" to the folder containing "pixelTrack.py"

Command to run "pixelTrack.py": "python pixelTrack.py parameterFile startStep [endStep]"

If "endStep" is not provided, "pixelTrack.py" will run from "startStep" to the last step (See **Section 3** to determine appropriate start and end steps).

IMPORTANT: "pixelTrack.py" should only be run on data that overlaps.

1.5 Output

"pixelTrack.py" generate a file called "pixelTrackLog.txt" that records, for each run, the start step, end step and time elapsed.

2 Parameters File

2.1 Parameters File Template

Note: An entry in the parameters file must have the following format: "name=value".

Template Parameters File

RawRawFolder=/path/to/data

DataType=seebelow

ROIPAC=/location/of/ROIPAC/scripts

PyPacks=/location/of/numpy/scipy/pylab/and/pyhdf/packages/site-packages

PYTHON=/location/of/python/scripts/that/come/with/this

RADAR parameters

Angle=Look angle of instrument (should be number)

DEM=/path/to/dem/file.dem

MaxBaseline=Maximum baseline distance between scenes in meters(for radar data), this should be a number

MinDateInterval=Minimum separation in days between scenes (for radar data), this should be an integer

MaxDateInterval=Maximum separation in days between scenes (for radar data), this should be an integer

awin=Window size in pixels along azimuth axis, this should be an integer

rwin=Window size in pixels along range axis, this should be an integer

wsamp=Sampling rate, this should be an integer

numproc=Definitely should be an integer

CPXorRMG="CPX" or "RMG". Use both phase and amplitude or just amplitude for pixel-tracking, respectively

ASTER parameters

Band=Band number of band to be used for pixel-tracking

2.2 Parameters

RawRawFolder

"RawRawFolder" should be the path to the folder containing the data to be processed. "RawRawFolder" is the folder where "pixelTrack.py" looks for all files (except those included in "pixelTrack.tar". "pixelTrack.py" also uses this folder for processing, creating all necessary files and folders here.

DataType

Name of the instrument used to acquire the data. Valid names include "ERS", "RADARSAT". Default is "ERS".

Angle

Look-angle of the instrument. Default is "23" (look-angle for ERS).

MaxDateInterval

Integer indicating the maximum separation (in days) between two scenes for scene pair to be processed.

MinDateInterval

Integer indicating the minimum separation (in days) between two scenes for scene pair to be processed.

PyPacks

Location of scipy, numpy, pylab, and pyhdf on machine, specifically the "site-packages" folder, e.g. "/location/of/numpy/site-packages".

PYTHON

Location of PYTHON scripts that come with "pixelTrack.py", such as "generic_load_make_azo.py", NOT the location of python on the machine.

ROIPAC

Location of ROIPAC scripts, i.e. /here/is/ROIPAC, where "roipac" is a folder, and contains a bunch of perl scripts (such as "process_2pass.pl").

DEM

Location of ".dem" file that will be used in processing, file should have ".dem" extension, e.g. "/here/is/the/dem.dem".

Note: DEM downloaded to Linux machine may need to be byte-swapped to produce a DEM with two-byte integers. If the DEM contains erroneous values (far too high or too low), then this step is probably necessary. The command is: "byte-swap 2 ¡ input_file ¿ output_file". "byte-swap" is available online.

MaxBaseline

The maximum distance (in meters) of "P_BASELINE" that will be allowed to continue processing a particular pair of scenes.

awin

The integer "awin" is the size in pixels of the ampcor search window along the azimuth axis. The value of awin multiplied by the size of a pixel (in meters) along the azimuth axis should be twice the maximum expected movement along the azimuth axis (in meters). If the ration (awin/rwin) is not the same as the pixel ratio used by ROIPAC, than the pixel-tracking results will have to be "looked down" (i.e. reduced in resolution).

rwin

The integer "rwin" is the size in pixels of the ampcor search window along the range axis. The value of rwin multiplied by the size of a pixel (in meters) along the range axis should be twice the maximum expected movement along the range axis (in meters). If the ration (awin/rwin) is not the same as the pixel ratio used by ROIPAC, than the pixel-tracking results will have to be "looked down" (i.e. reduced in resolution).

wsamp

This is the sampling rate, and it should be an integer. “wsamp” determines how often a result is generated. For example, if wsamp is 2, awin is 100 and rwin is 20, then a result will be generated every 100/2=50 azimuth pixels, and 20/2=10 range pixels.

numproc

However many processors you want to use on your system. This should definitely be an integer. Check how many processors you have (on linux/unix) with the following command: ”ls /proc/acpi/processor/”. However many CPU folders there are that’s (probably) how many processors you have. If that command doesn’t work there are lots of other, more ”robust” ways to find out (just look online).

CPXorRMG

If this is set to ”CPX”, then both the phase and amplitude will be used to generate pixel-tracking offsets. If it is set to ”RMG”, then only the amplitude will be used. ”RMG” is the default.

ASTER Parameters

Band

Recommended value for this parameter is “3N”. Value should correspond to dimension names for “ImageData” dataset in ASTER hdf file.

3 Steps

Steps for ”pixelTrack.py”, in order

Radar

uncompress_raw_raw

setup

make_raw

make_int_dirs_and_procs

baselines

offsets

ampcor

make_unw

affine

geocode

ASTER

optical_raw

These steps can (theoretically) be run independently. Please read the description of each step to get a sense of the files it needs. There are many files that ROIPAC builds, if these get moved around or changed too much then problems will arise when trying to "automatically" process data. Here are the steps layed out "step-by-step", addressing what each step does and what files it needs to perform its intended function.

uncompress_raw_raw

Required files: Compressed (really raw) raw data.

Function: Uncompresses the rawest of the raw data. Recognized extensions: ".tar", ".gz", ".bz2", ".gzip", ".bzip2", and ".zip". Uses "bunzip2", "gunzip" and "tar" commands.

setup

Required files: "*.ldr" files in binary format that contain a date ("pixelTrack.py" looks for at least 7 consecutive digits within the file), and ".raw" files that have the same name as the ".ldr" files except for their extension.

Function: Searches for "*.ldr" files, searches these files for a date, and makes YYMMDD date directories in "RawRawFolder" if they don't exist already. "pixelTrack.py" then makes a link to the leader file in the corresponding date directory with the name "SARLEADERDD" (unless a "SARLEADERDD" link in this date directory to the leader file exists already), where "DD" is a pair of digits to distinguish this leader file from others. Then the script tries to locate a raw file with the same name as the leader file by replacing ".ldr" with ".raw". If it cannot find this, it gives a warning, otherwise it makes a link to this raw file in the same date directory as the link to the leader file, with the name "IMAGERYDD", where "DD" is the same pair of digits as the corresponding leader file.

Note: "setup" will not overwrite existing "SARLEADER" or "IMAGERY" links/files.

make_raw

Required files: "date" directories ("YYMMDD" name) in the "RawRawFolder" that contain "SARLEADER*" links/files, and corresponding "IMAGERY*" links/files.

Function: "make_raw" runs the appropriate "make_raw" script from ROIPAC. The "DataType" parameter is used to determine the appropriate ROIPAC "make_raw" script, e.g., if your "DataType" is "ERS", then "make_raw" will use "ROIPAC/make_raw_ASF.pl", but if "DataType" is "RADARSAT" then the "make_raw" step will run "make_raw_RSAT-CEOS.pl".

If everything goes smoothly the result will be "YYMMDD.raw" files of non-zero size in the "YYMMDD" folders, made from all the "SARLEADER*" and "IMAGERY*" files in the "YYMMDD" folder.

make_int_dirs_and_procs

Required files: "YYMMDD" folders in your "RawRawFolder".

Function: This step will make "int_date2_date1" directories and "int_date2_date1.proc" files in "RawRawFolder", where "date2" (YYMMDD) is later than "date1" and the difference (in days) between the two is ¿= MinDateInterval and ¡= MaxDateInterval.

"make_int_dirs_and_procs" will attempt to generate the azimuth pixel size of one of the scenes in order to find the azimuth-to-range pixel ratio and put it in the proc file(s). It runs ROIPACs' "dopav.pl" and "roi_prep.pl" on a temporary directory to create a ".slc.rsc" file that contains the azimuth pixel size. This part of "make_int_dirs_and_procs" requires a look angle to be given in the parameters file, in addition to complete ".raw" and ".raw.rsc" files in order to run successfully.

baselines

Required files: "*.proc" files in your "RawRawFolder" (and the successful results of the "make_raw" step).

Function: Runs "ROIPAC/process_2pass.pl" from "raw" to "orbbase" for every "*.proc" file in "RawRawFolder". Generates baselines used for filtering in next step ("offsets").

offsets

Required files: "*baseline*" files (and all other files) that are the result of a successful "baselines" step. These should be in "int_date2_date1" directories, because the "offsets" step assumes the ".proc" file associated with any "*baseline*" file it finds are in the directory above the "*baseline*" file and have the same name as the directory containing their corresponding "*baseline*" file (with the addition of a ".proc" extension).

Function: Generates offsets by running "ROIPAC/process_2pass.pl" from "orbbase" to "offsets". The "offsets" step does not process any scene pair with a "P_BASELINE" whose absolute value is greater than the maximum allowed baseline ("MaxBaseline" in the parameters file, default value is "500").

ampcor

Required files: "*_cull.off" (where "*" includes two separate six-digit dates) and the ".slc" files for those two dates (the name of the slc files must be "date1.slc" and "date2.slc"). Also requires the

".rsc" files for the cull file and the two slc files. The cull file and the slc files should be in the same folder that all processing for their corresponding scene pair has taken place in.

Function: Generates pixel-tracking results using "ROIPAC/azo.pl" (ampcor) for the scene pair using the values for "rwin", "awin" and "wsamp" given in the parameters file (if those are not in the parameters file, then "pixelTrack.py" uses the default values). Generates a binary file from the resulting ".off" file using "ROIPAC/azo_merge.pl".

make_unw

Required files: "*rwinxawin.off.bin" files in "RawRawFolder", "rwin" and "awin" being parameters from the parameters file. The ".off.bin" file is the result of a successful "ampcor" step. Requires the python script "generic_load_make_azo.py" located in the "PYTHON" directory (set in parameter) file. This is packaged with "pixelTrack.py". Also requires the ".rsc" file that results from a successful ampcor step.

Function: Makes separate unw files for the azimuth offsets and the range offsets using Python script provided with "pixelTrack.py".

affine

Required files: Results of a successful "offsets" step, contained in directories that have the same name as their corresponding ".proc" files, and are in the same location. For example, if there is a "int_date2_date1.proc" file whose full path is "RawRawFolder/subfolder1/subfolder2/int_date2_date1.proc", then the results of the successful "offsets" step corresponding to this proc file should be in the "RawRawFolder/subfolder1/subfolder2/int_date2_date1/" directory.

Function: Creates an affine transformation file (used to geocode the pixel-tracking results) by running "ROIPAC/process_2pass.pl" from "offsets" (don't worry, it won't redo these so long as the input files haven't been changed) to "done_sim_removal". Future versions of this script might contain a stripped-down perl/python script that removes some of the processing done by ROIPAC that is not absolutely necessary to geocode the pixel-tracking results.

geocode

Required files: "azimuth*.unw"/"range*.unw" files (results of successful "make_unw" step), file named "log" (automatically generated by ROIPAC) containing the entry for "ROIPAC/azo.pl" (should be there if "pixelTrack.py" was used to run the "ampcor" step), affine transformation file "date2-date1_SIM.aff", with "date2" and date1" derived from the "azo.pl" entry of the log file, full-res "date2.slc" and "date1.slc", and the succesfful results of "affine", all in the same folder.

Function: Build geo-referenced unw files for range and azimuth offsets. The final result will be "geo_azimuth*.unw" and "geo_range*.unw" files, the "*" being "_Drlks", where "D" is the range look-downs on the initial unw file to achieve the same pixel ratio as the affine transformation file.

ASTER steps

optical_raw

Required files: ASTER product 1 "*.hdf" files in "RawRawFolder".

Function: Extract band with band name "Band" from each ASTER hdf file and output it as a flat file of 32-bit floats, along with a metadata file of the same name (with a ".met" extension) that contains the number of samples and lines in the image (rows and columns).

These are put into a directory in RawRaw folder named "bandBand", e.g. if "Band" is "3N" then the results of this step will be placed in a folder named "band3N".

gausshp_filt

Required files: Flat band files of 32-bit floats that have a ".b*" extension and their associated ".met" files (same file name with ".met" on then end) that list the number of samples and lines.

Function: Apply a high-pass gaussian filter to each of the data files and output the results as a flat file of 32-bit floats. The name of the output files are the same as the name of their corresponding band input files with the addition of "_ghp" before the file extension.

4 Example

4.1 Hector Mine Earthquake

The following results were generated using the SAR images from ERS track 127, frame 2907, dates 10/20/1999 and 9/15/1999.

4.1.1 Parameter File

RawRawFolder=/lr/akm26/work/TrueTest4

CPXorRMG=RMG

ROIPAC=/home/matt/insar/INT_SCR

PYTHON=/lr/akm26/work/Python

PyPacks=/lr/akm26/work/Python/Libs/lib64/python2.4/site-packages

DEM=/lr/akm26/work/TrueTest4/DEM/hector_mine.dem

MaxBaseline=500

MinDateInterval=1

MaxDateInterval=37

DataType=ERS

Angle=23

rwin=20

awin=100

wsamp=2

numproc=2

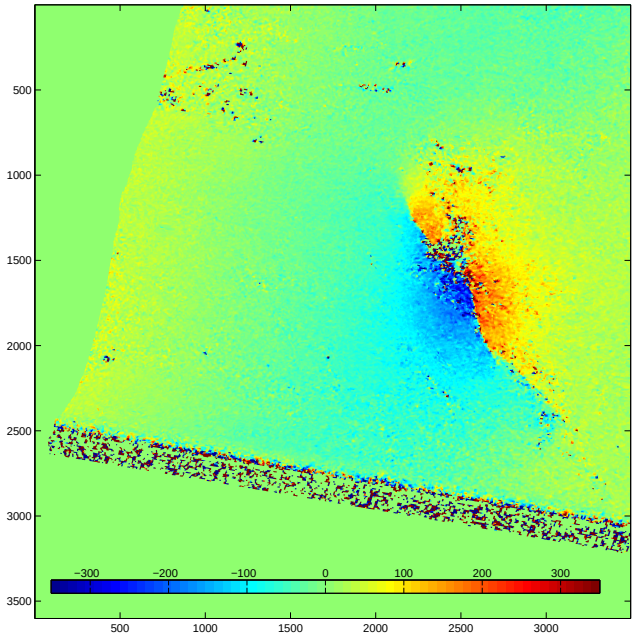

Figure 1: AZO displacements (cm) between scenes 1999-09-15 and 1999-10-20 (Hector Mine Earthquake).

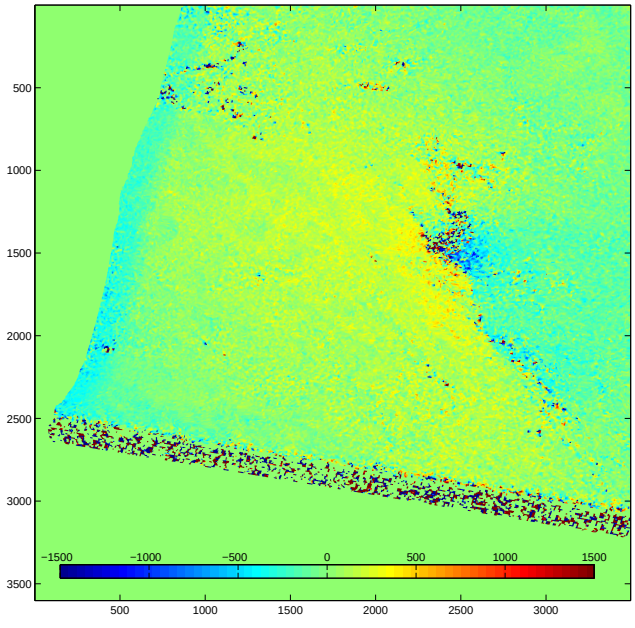

Figure 2: Range displacements (cm) between scenes 1999-09-15 and 1999-10-20 (Hector Mine Earthquake).