SAR Pixel-Tracking Using “pixelTrack.py”

Andrew Kenneth Melkonian

August 28, 2014

Prerequisites:

Python, including numpy, scipy, pylab, and matplotlib

ROI_PAC, with paths set up by “SAR_CONFIG” or equivalent file and ROI_PAC scripts and executables added to your path in the “.bashrc” and/or “.bash_profile” files (for bash users).

Perl, to run ROI_PAC scripts.

USAGE:

python pixelTrack.py params.txt start_step end_step

Input:

start_step/end_step: Valid start and end steps, can be one of the following:

uncompress/preraw/makeraw/setup/baselines/offsets/ampcor/make_unw/affine/geocode

The steps are listed in order, make sure “start_step” is BEFORE “end_step” on the above list.

params.txt: ASCII text file setting parameters used by scripts, for example:

```
::::::::::::::
params_ALOS_example.txt
::::::::::::::

WorkPath = /data/username123/ALOS
DEM = /data/username123/DEM/my_favorite_dem.dem
MaxBaseline = 2000
MinDateInterval = 45
MaxDateInterval = 47
DataType = ALOS
Angle = 34.3
rwin = 40
```

awin = 80
search_x = 16
search_y = 16
wsamp = 4
numproc = 2

WorkPath: Directory to work in, this should contain the files necessary to process from "start_step" to "end_step". The requirements will vary from step to step. NOTE: A different directory should be set up for different paths/tracks, the script will NOT distinguish between scenes from different paths/tracks when applying the criteria described below.

DEM: DEM to use, should be the same format as DEMs used for standard ROI_PAC processing (often get_SRTM3.pl can be used to obtain a properly-formatted DEM).

MaxBaseline: Pairs with an average perpendicular baseline above this value will NOT be processed further (the script checks this against the values in the *baseline*.rsc file).

MinDateInterval: Minimum date interval, pairs with a separation time BELOW this value will NOT be processed.

MaxDateInterval: Maximum date interval, pairs with a separation time ABOVE this value will NOT be processed.

DataType: The sensor/satellite that the data was acquired by, can be "ALOS", "ERS", or "TSX".

Angle: Look angle of the satellite, for "TSX" data type this value is determined from the metadata.

rwin: Reference window size in the range direction (in pixels), should reflect the "pixel_ratio" calculated for this pair. "pixel_ratio" is the ratio of the ground pixel size in the range direction to the ground pixel size in the azimuth direction.

awin: Reference window size in the azimuth direction (in pixels), should reflect the "pixel_ratio" calculated for this pair.

search_x: Search window size in the range direction (in pixels), should be set to capture the maximum expected motion.

search_y: Search window size in the range direction (in pixels), should be set to capture the maximum expected motion.

wsamp: Determines the sampling frequency, which will be rwin/wsamp in the range direction and awin/wsamp in the azimuth. For the example above, the values would result in the offset being calculated every 10 pixels in the range direction and every twenty in the azimuth direction. The pixel-tracking output would therefore be downsampled by a factor of 10 in the range and 20 in the azimuth relative to the full-resolution "*.slc" files.

numproc: Number of processors to run on, for Linux or MacOS-X use the command "top" then enter "1" to check how many "processors" there are.

Summary:

These parameters are set up to run ALOS pairs in the directory /data/username123/ALOS, using two processors for EACH pair. Only pairs with a baseline below 2000 and a separation time of 46 days are run.

EXAMPLE RUN FOR ALOS:

“/data/username123/ALOS” contains uncompressed ALOS data files of the form “ALPS*.zip” (and does NOT contain any other files, for now). Create the file “params.txt” in “/data/username123/ALOS and use the values given in the example parameters file above. Run “cd /data/username123/ALOS” then run “python /path/to/pixelTrack.py params.txt uncompress preraw”. Make sure you are ALWAYS in “/data/username123/ALOS” when running the pixel-tracking script. The result should be that the zip files are unzipped and also copied to a newly-created backup folder named “ARCHIVE”.

Next run “python /path/to/pixelTrack.py params.txt preraw makeraw” The result should be that the contents of the files unzipped in the last step are moved to their appropriately-labeled 6-digit date directories (YYMMDD), ready for the “makeraw” step.

Next run “python /path/to/pixelTrack.py params.txt makeraw setup”. The result should be that the appropriate ROI_PAC “make_raw*.pl” script is run, and *.raw files created in each 6-digit date folder.

Next run “python /path/to/pixelTrack.py params.txt setup baselines”. The result should be that “int_date1_date2” directories and “*.proc” files are set up for date pairs that meet the criteria given in “params.txt”.

Next run “python /path/to/pixelTrack.py params.txt baselines offsets”. The result should be that a “*baseline.rsc” file is created for each pair.

Next run “python /path/to/pixelTrack.py offsets ampcor”. This step NOT the pixel-tracking step, rather, it is part of the standard ROI_PAC processing that coregisters the “*.slc” files (SAR images) for each pair.

Next run “python /path/to/pixelTrack.py ampcor make_unw”. This will set up everything so that the pixel-tracking can be run, and then will output the commands necessary to perform the pixel-tracking to the screen. The user then manually runs the commands by copying the output of this step and pasting it in the command line. ALTERNATIVELY, you can run “python /path/to/pixelTrack.py ampcor make_unw ¿ run_amp.cmd”, the “chmod u+x run_amp.cmd”, then “./run_amp.cmd”.

Next run “python /path/to/pixelTrack.py make_unw affine”. The result will be “azimuth*.unw”, “range*.unw”, and “snr*.unw” files for each pair that contain the azimuth offsets, range offsets and signal-to-noise ratios, respectively.

Next run “python /path/to/pixelTrack.py affine geocode”. This will run a slightly modified version of several ROI_PAC scripts that will find and save the affine transformation parameters between the DEM and the radar to be used in geocoding. The result should be a “*.aff” file for each pair.

Next run “python /path/to/pixelTrack.py geocode”. This will create “geo_azimuth*.unw”, “geo_range*.unw” and “geo_snr*.unw” files for each pair that contain the geocoded azimuth offsets, range offsets and signal-to-noise ratios, respectively.