PySAR

A Python package for InSAR time series analysis

PySAR Training Workshop

Zhang Yunjun

Geodesy Lab, University of Miami, FL

Nov 17th, 2017

What's PySAR?

- PySAR is a open-source Python package for InSAR (Interferometric Synthetic Aperture Radar) time series analysis.
- It reads stack of coregistered, unwrapped interferograms processed by ROI_PAC, Gamma or ISCE, and produces three dimensional (2D in space and 1D in time) ground displacement.
- Download: git clone https://github.com/yunjunz/PySAR.git
- Installation: https://github.com/yunjunz/PySAR#1-installation
- Google Group: https://groups.google.com/forum/#!forum/py-sar

How to use it?

1. Generate a template file:

```
cd ~/KujuAlosAT422F650/PYSAR
pysarApp.py -g
```

2. Modify ”Step 1. Load Data” options in `pysarApp_template.txt` to add the path of input files

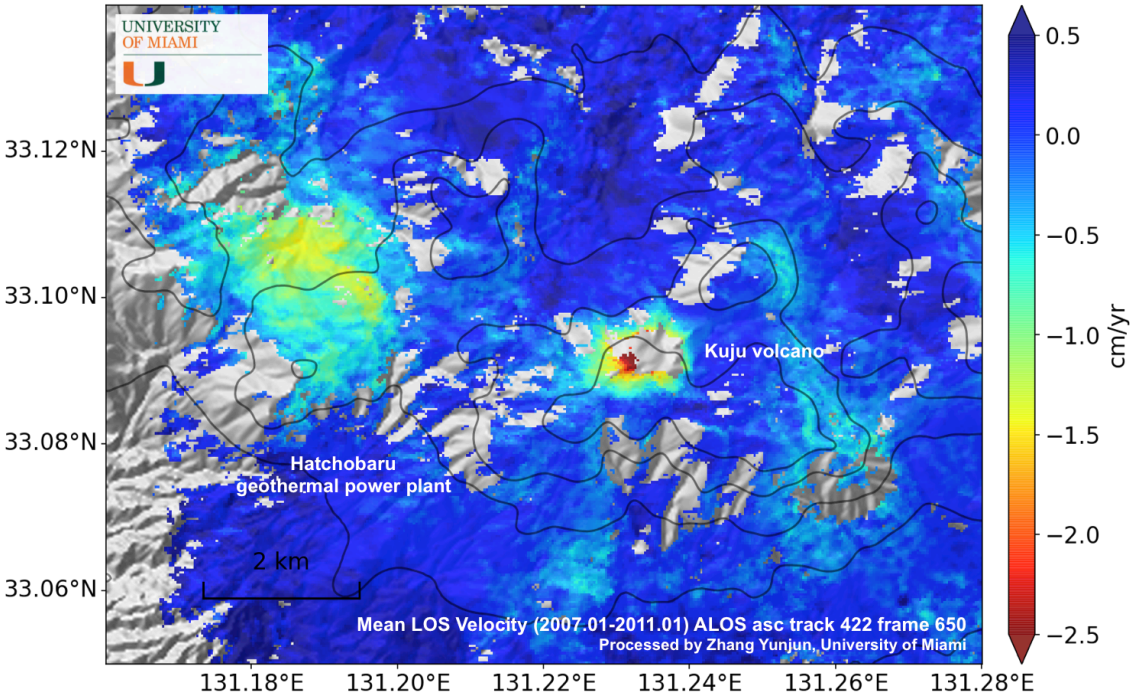
```
pysar.insarProcessor = roipac
pysar.unwrapFiles    = ~/KujuAlosAT422F650/ROIPAC/RADAR/filt_*.unw
pysar.corFiles       = ~/KujuAlosAT422F650/ROIPAC/RADAR/filt_*.cor
pysar.lookupFile     = ~/KujuAlosAT422F650/ROIPAC/RADAR/geomap*.trans
pysar.dem.radarCoord = ~/KujuAlosAT422F650/ROIPAC/RADAR/radar*.hgt
pysar.dem.geoCoord   = ~/KujuAlosAT422F650/ROIPAC/RADAR/*.dem
```

3. Run pysarApp.py and check results plot in PIC folder!

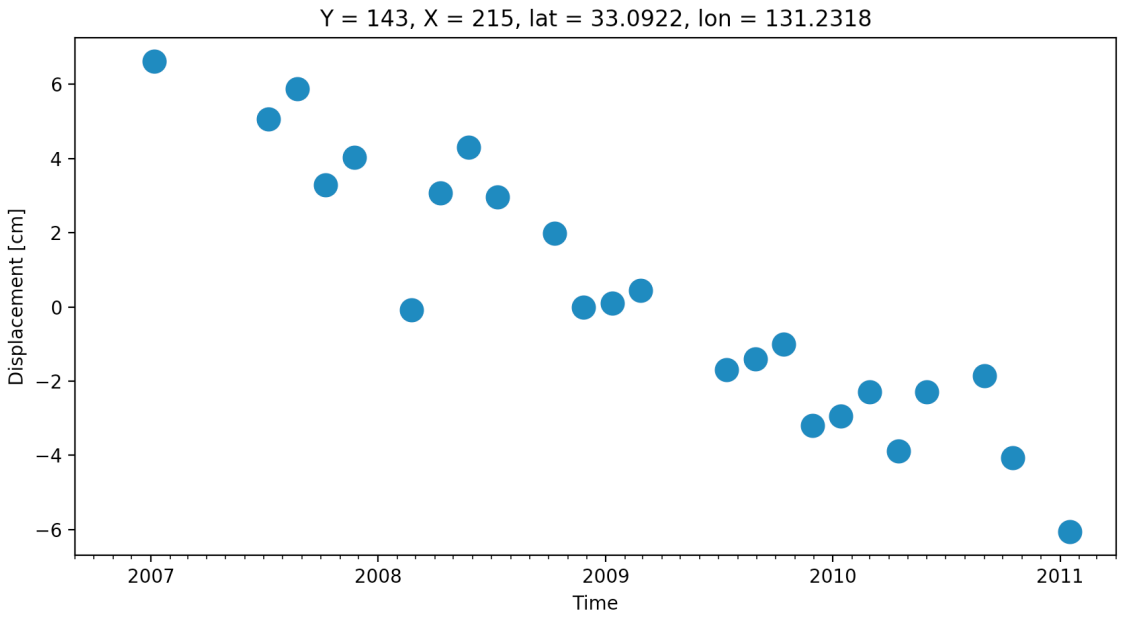
```
pyarApp.py
```

- Test data KujuAlosA:

Mean LOS Velocity



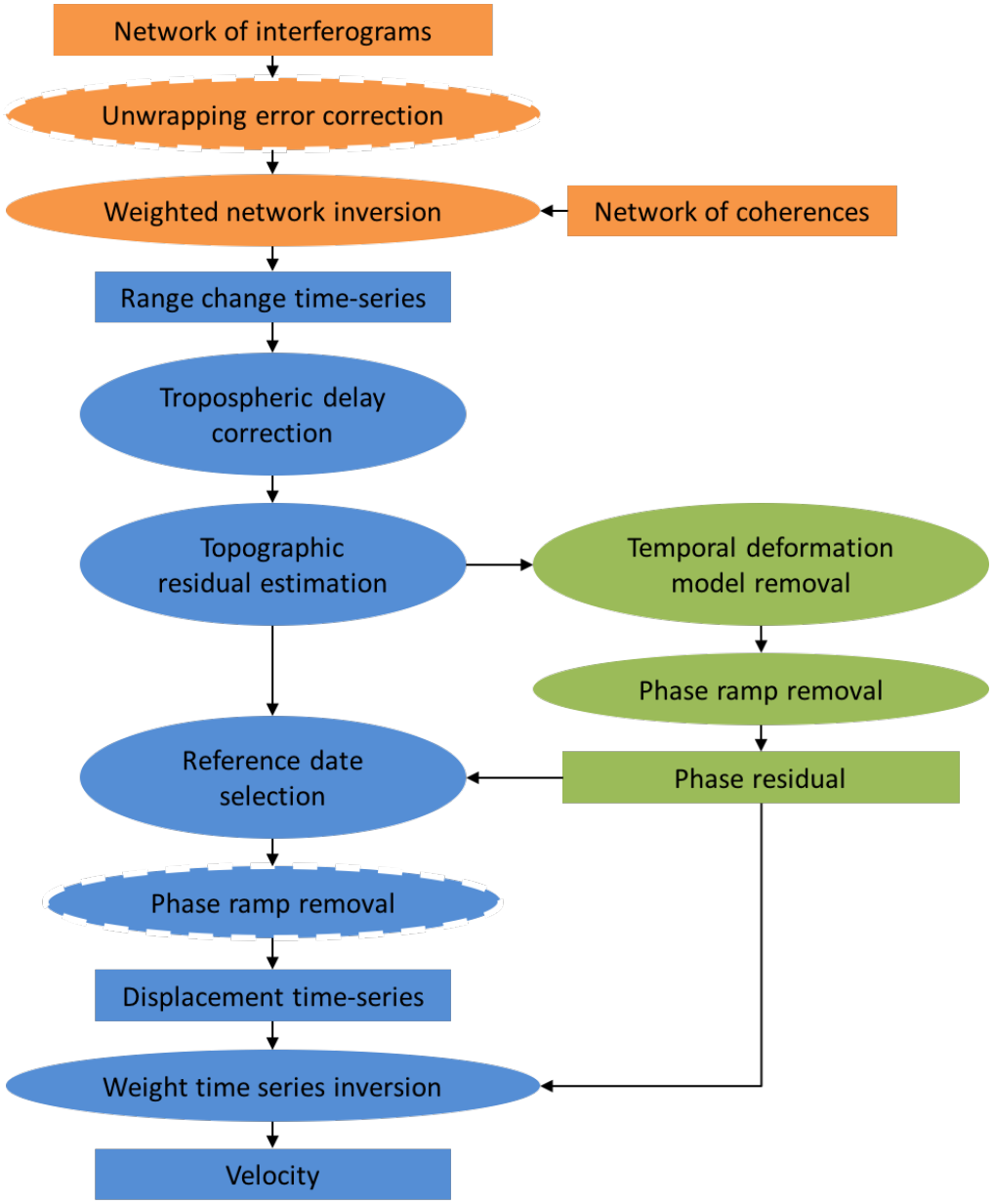
Displacement History @ Kuju Volcano



Plot result using view.py and tsviewer.py, with -h for help/usage.

pysarApp.py

Workflow



pysarApp.py - steps

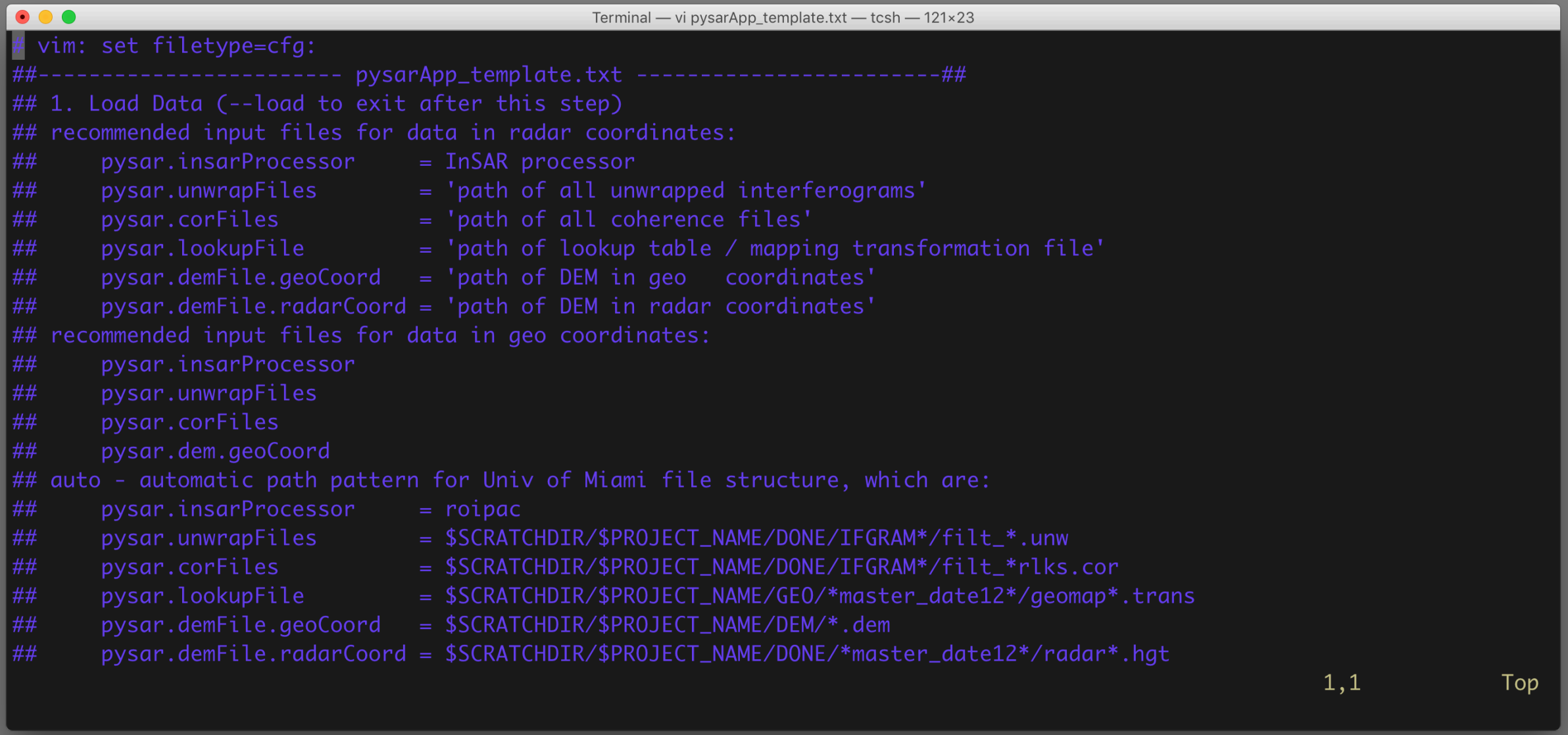
1. Load Data
    1. Subset (optional)
    2. Prepare geometry
2. Reference in Space
3. Unwrap Error Correction (optional)
4. Modify Network (optional)
5. Network Inversion

pysarApp.py - steps (cont.)

6. Local Oscillator Drift Correction (for Envisat)
7. Tropospheric Delay Correction
8. Topographic Phase Residual Correction
    1. Phase residual RMS
9. Reference in Time
10. Phase Ramp Removal
11. Velocity Estimation
12. Post-processing: geocode, export, plot

pysarApp_template.txt – basic rules

- pysarApp.py runs in the order of options listed here in the file
- To adjust your processing result, change the corresponding option value in the template txt file and re-run pysarApp.py
- “auto” for default option value, with explanation in the comment in the same line
- custom value input format/range is explained in the comment in the same line


Terminal — vi pysarApp_template.txt — tcsh — 121×23
vim: set filetype=cfg:
##------------------------ pysarApp_template.txt ------------------------##
1. Load Data (--load to exit after this step)
recommended input files for data in radar coordinates:
pysar.insarProcessor   = InSAR processor
pysar.unwrapFiles   = 'path of all unwrapped interferograms'
pysar.corFiles   = 'path of all coherence files'
pysar.lookupFile   = 'path of lookup table / mapping transformation file'
pysar.demFile.geoCoord   = 'path of DEM in geo   coordinates'
pysar.demFile.radarCoord = 'path of DEM in radar coordinates'
recommended input files for data in geo coordinates:
pysar.insarProcessor
pysar.unwrapFiles
pysar.corFiles
pysar.dem.geoCoord
auto - automatic path pattern for Univ of Miami file structure, which are:
pysar.insarProcessor   = roipac
pysar.unwrapFiles   = $SCRATCHDIR/$PROJECT_NAME/DONE/IFGRAM*/filt_*.unw
pysar.corFiles   = $SCRATCHDIR/$PROJECT_NAME/DONE/IFGRAM*/filt_*rlks.cor
pysar.lookupFile   = $SCRATCHDIR/$PROJECT_NAME/GEO/*master_date12*/geomap*.trans
pysar.demFile.geoCoord   = $SCRATCHDIR/$PROJECT_NAME/DEM/*.dem
pysar.demFile.radarCoord = $SCRATCHDIR/$PROJECT_NAME/DONE/*master_date12*/radar*.hgt
1,1 Top

Terminal — vi pysarApp_template.txt — tcsh — 121×23

```
##     pysar.lookupFile         = $SCRATCHDIR/$PROJECT_NAME/GEO/*master_date12*/geomap*.trans
##     pysar.demFile.geoCoord   = $SCRATCHDIR/$PROJECT_NAME/DEM/*.dem
##     pysar.demFile.radarCoord = $SCRATCHDIR/$PROJECT_NAME/DONE/*master_date12*/radar*.hgt
pysar.insarProcessor     = auto  #[roipac,        gamma,           isce], auto for roipac
pysar.unwrapFiles        = auto  #[filt*rlks.unw, diff_*rlks.unw,  filt*.unw]
pysar.corFiles           = auto  #[filt*rlks.cor, filt_*rlks.cor,  filt*.cor]
pysar.lookupFile         = auto  #[geomap*.trans, sim*.UTM_TO_RDC, l*.rdr]
pysar.demFile.radarCoord = auto  #[radar*.hgt,    sim*.hgt_sim,    hgt.rdr]
pysar.demFile.geoCoord   = auto  #[*.dem,         sim*.utm.dem,    demLat*.dem.wgs84]


## 1.1 Subset (optional, --subset to exit after this step)
## if both yx and lalo are specified, use lalo option unless a) no lookup file AND b) dataset is in radar coord
pysar.subset.yx       = auto    #[1800:2000,700:800 / no], auto for no
pysar.subset.lalo     = auto    #[31.5:32.5,130.5:131.0 / no], auto for no
pysar.subset.tightBox = auto    #[yes / no], auto for yes, tight bounding box for files in geo coord
pysar.multilook.yx    = auto    #[4,4 / no], auto for no [not implemented yet]

## 1.2 Prepare geometry files
## Prepare incidenceAngle.h5, rangeDistance.h5 files

                                                                    41,0-1        12%
```

Terminal — vi pysarApp_template.txt — tcsh — 121×23

```
## 2. Reference in Space
## reference all interferograms to one common point in space
## auto - randomly select a pixel with coherence > minCoherence
pysar.reference.yx            = auto   #[257,151 / auto]
pysar.reference.lalo          = auto   #[31.8,130.8 / auto]

pysar.reference.coherenceFile = auto   #[filename], auto for averageSpatialCoherence.h5
pysar.reference.minCoherence  = auto   #[0.0-1.0], auto for 0.85, minimum coherence for auto method
pysar.reference.maskFile      = auto   #[filename / no], auto for mask.h5


## 3. Unwrapping Error Correction
## unwrapping error correction based on the following two methods:
## a. phase closure (Fattahi, 2015, PhD Thesis)
## b. connecting bridge
pysar.unwrapError.method   = auto   #[bridging / phase_closure / no], auto for no
pysar.unwrapError.maskFile = auto   #[filename / no], auto for no
pysar.unwrapError.ramp     = auto   #[plane / quadratic], auto for plane
pysar.unwrapError.yx       = auto   #[y1_start,x1_start,y1_end,x1_end;y2_start,...], auto for none

                                                                40,0-1        26%
```

Terminal — vi pysarApp_template.txt — tcsh — 121×23

```
## 4. Modify Network (optional)
## Coherence-based network modification = MST + Threshold, by default
## 1) calculate a average coherence value for each interferogram using spatial coherence and input mask (with AOI)
## 2) find a minimum spanning tree (MST) network with inverse of average coherence as weight (keepMinSpanTree)
## 3) for all interferograms except for MST's, exclude those with average coherence < minCoherence.
pysar.network.coherenceBased  = auto  #[yes / no], auto for yes, exclude interferograms with coherence < minCoherence
pysar.network.keepMinSpanTree = auto  #[yes / no], auto for yes, keep interferograms in Min Span Tree network
pysar.network.coherenceFile   = auto  #[filename], auto for coherence.h5
pysar.network.minCoherence    = auto  #[0.0-1.0], auto for 0.7
pysar.network.maskFile        = auto  #[filename, no], auto for [maskLand.h5, mask.h5][0], no for all pixels
pysar.network.maskAoi.yx      = auto  #[y0:y1,x0:x1 / no], auto for no, area of interest for coherence calculation
pysar.network.maskAoi.lalo    = auto  #[lat0:lat1,lon0:lon1 / no], auto for no - use the whole area

pysar.network.tempBaseMax     = auto  #[1-inf, no], auto for no, maximum temporal baseline in days
pysar.network.perpBaseMax     = auto  #[1-inf, no], auto for no, maximum perpendicular spatial baseline in meter
pysar.network.referenceFile   = auto  #[date12_list.txt / Modified_unwrapIfgram.h5 / no], auto for no
pysar.network.excludeDate     = auto  #[20080520,20090817 / no], auto for no
pysar.network.excludeIfgIndex = auto  #[1:5,25 / no], auto for no, list of interferogram number starting from 1
pysar.network.startDate       = auto  #[20090101 / no], auto for no
pysar.network.endDate         = auto  #[20110101 / no], auto for no
                                                                              61,0-1        40%
```

```
## 5. Network Inversion
## Invert network of interferograms into time series
## Temporal coherence (weighted) is calculated using Tazzani et al. (2007, IEEE-TGRS)
## For no/uniform weight approach, use Singular-Value Decomposition (SVD) if network are not fully connected
## For weighted approach, use weighted least square (WLS) solution with the following weighting functions:
##     variance - phase variance due to temporal decorrelation (Yunjun et al., 2017, in prep)
##     no       - no weight, or ordinal inversion with uniform weight (Berardino et al., 2002, IEEE-TGRS)
##     linear   - uniform distribution CDF function (Tong et al., 2016, RSE)
pysar.timeseriesInv.weightFunc    = auto #[variance / no / linear / normal], auto for no, coherence to weight
pysar.timeseriesInv.coherenceFile = auto #[filename / no], auto for coherence.h5, file to read weight data
pysar.timeseriesInv.residualNorm  = auto #[L2 ], auto for L2, norm minimization solution
pysar.timeseriesInv.minTempCoh    = auto #[0.0-1.0], auto for 0.7, min temporal coherence for mask
```

Terminal — vi pysarApp_template.txt — tcsh — 121×23

```
## 6. Local Oscillator Drift (LOD) Correction (for Envisat only, Marinkovic and Larsen, 2013, Proc. LPS)
## correct LOD if input dataset comes from Envisat
## skip this step for all the other satellites.


## 7. Tropospheric Delay Correction
## correct tropospheric delay using the following methods:
## a. pyaps - use weather re-analysis data (Jolivet et al., 2011, GRL, need to install PyAPS; Dee et al., 2011)
## b. height_correlation - correct stratified tropospheric delay (Doin et al., 2009, J Applied Geop)
## c. base_trop_cor - (not recommend) baseline error and stratified tropo simultaneously (Jo et al., 2010, Geo J)
pysar.troposphericDelay.method       = auto  #[pyaps / height_correlation / base_trop_cor / no], auto for pyaps
pysar.troposphericDelay.polyOrder    = auto  #[1 / 2 / 3], auto for 1, for height_correlation method
pysar.troposphericDelay.weatherModel = auto  #[ERA / MERRA / NARR], auto for ECMWF, for pyaps method
```

102,0-1 62%

Terminal — vi pysarApp_template.txt — tcsh — 121×23

```
## 8. Topographic (DEM) Residual Correction (Fattahi and Amelung, 2013, IEEE-TGRS)
## Specify stepFuncDate option if you know there are sudden displacement jump in your area,
## i.e. volcanic eruption, or earthquake, and check timeseriesStepModel.h5 afterward for their estimation.
pysar.topoError              = auto  #[yes / no], auto for yes
pysar.topoError.polyOrder    = auto  #[1-inf], auto for 2, polynomial order of temporal deformation model
pysar.topoError.stepFuncDate = auto  #[20080529,20100611 / no], auto for no, date of step jump
pysar.topoError.excludeDate  = auto  #[20070321,20101120 / txtFile / no], auto for no, date exlcuded for error estimation


## 8.1 Phase Residual Root Mean Square
## calculate the deramped Root Mean Square (RMS) for each epoch of timeseries residual from DEM error inversion
## To get rid of long wavelength component in space, a ramp is removed for each epoch.
## Recommendation: quadratic for whole image, plane for local/small area
pysar.residualRms.maskFile        = auto  #[filename / no], auto for maskTempCoh.h5, mask for ramp estimation
pysar.residualRms.ramp            = auto  #[quadratic / plane / no], auto for quadratic
pysar.residualRms.threshold       = auto  #[0.0-inf], auto for 0.02, minimum RMS in meter for exclude date(s)
pysar.residualRms.saveRefDate     = auto  #[yes / no], auto for yes, save date with min RMS to txt/pdf file.
pysar.residualRms.saveExcludeDate = auto  #[yes / no], auto for yes, save date(s) with RMS > threshold to txt/pdf file.

                                                                    144,0-1       75%
```

Terminal — vi pysarApp_template.txt — tcsh — 121×23

```
## 9. Reference in Time
## reference all timeseries to one date in time
## auto - choose date with minimum residual RMS using value from step 8.1
## no   - do not change reference date, keep the defaut one (1st date usually) and skip this step
pysar.reference.date = auto   #[auto / reference_date.txt / 20090214 / no]


## 10. Phase Ramp Removal (optional)
## remove phase ramp for each epoch, useful to check localized deformation, i.e. volcanic, land subsidence, etc.
## [plane, quadratic, plane_range, plane_azimuth, quadratic_range, quadratic_azimuth, baseline_cor, base_trop_cor]
pysar.deramp          = auto  #[no / plane / quadratic], auto for no - no ramp will be removed
pysar.deramp.maskFile = auto  #[filename / no], auto for maskTempCoh.h5, mask file for ramp estimation


## 11. Velocity Inversion
## estimate linear velocity from timeseries, and from tropospheric delay file if exists.
pysar.velocity.excludeDate = auto   #[exclude_date.txt / 20080520,20090817 / no], auto for exclude_date.txt
pysar.velocity.startDate   = auto   #[20070101 / no], auto for no
pysar.velocity.endDate     = auto   #[20101230 / no], auto for no

                                                                143,0-1       88%
```

Terminal — vi pysarApp_template.txt — tcsh — 121×23

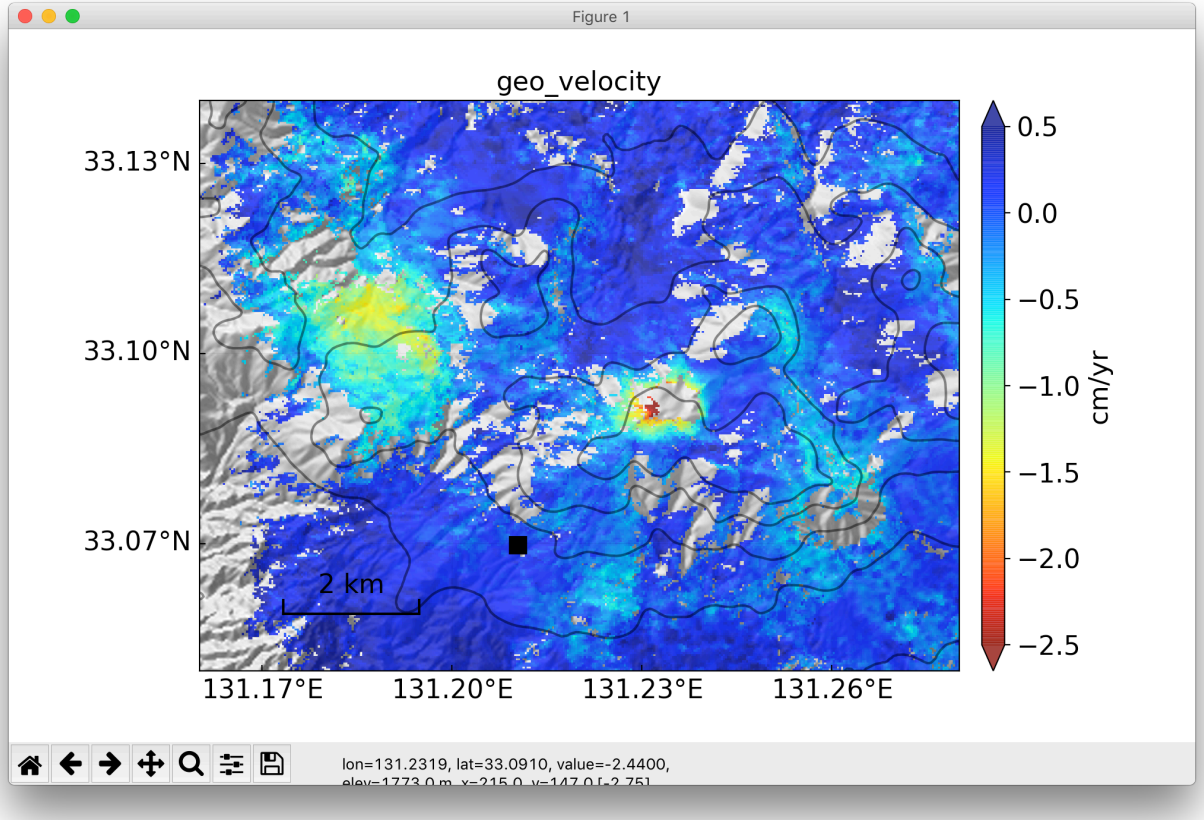
```
## 12. Post-processing (geocode, output to Google Earth, UNAVCO, etc.)
## 12.1 Geocode
## For data processed by ROI_PAC/Gamma, output resolution for geocoded file is the same as their lookup table file.
## For data processed by ISCE/Doris, output resolution is assign by user with resolution option:
## 1) float number - resolution in degree, 0.001 by default, around 100 m on equator
## 2) file name    - use the resolution from a file in geo coordinates, e.g. demGeo.h5
pysar.geocode            = auto  #[yes / no], auto for yes
pysar.geocode.resolution = auto  #[0.0-inf / filename], auto for 0.001 (~100 m), output resolution for ISCE processor

## 12.2 Export to other formats
## To update UNAVCO file with new acquisitions, enabling update mode, a.k.a. put XXXXXXXX as endDate
## in filename if endDate < 1 year
pysar.save.unavco  = auto   #[yes / update / no], auto for no, save timeseries to UNAVCO InSAR Archive format
pysar.save.kml     = auto   #[yes / no], auto for yes, save geocoded velocity to Google Earth KMZ file
pysar.save.geotiff = auto   #[yes / no], auto for no, save geocoded velocity to Geotiff format [not implemented yet]

## 12.3 Plot
pysar.plot = auto   #[yes / no], auto for yes, plot files generated by pysarApp default processing to PIC folder


                                                                                164,0-1       Bot
```

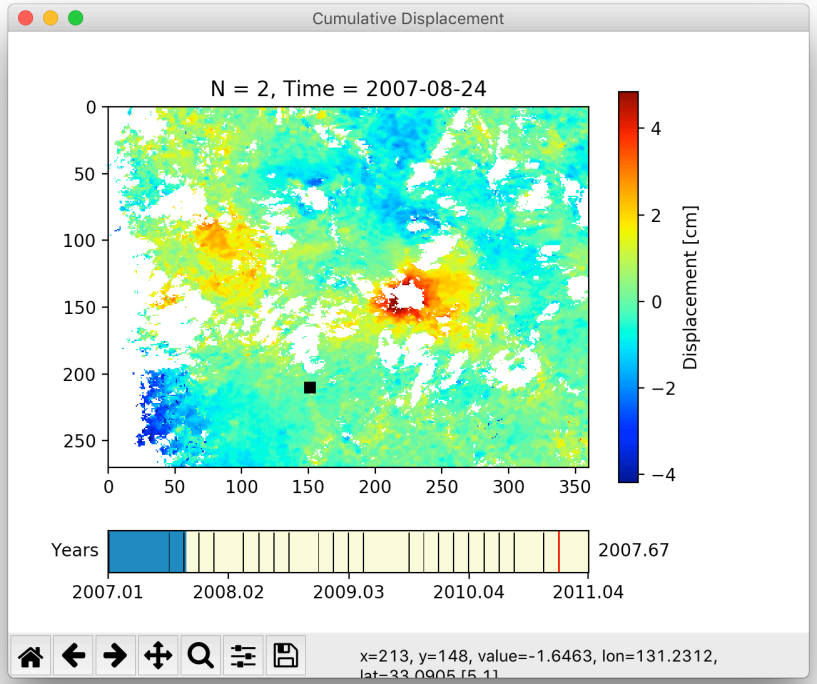
Play with plotting scripts – view.py

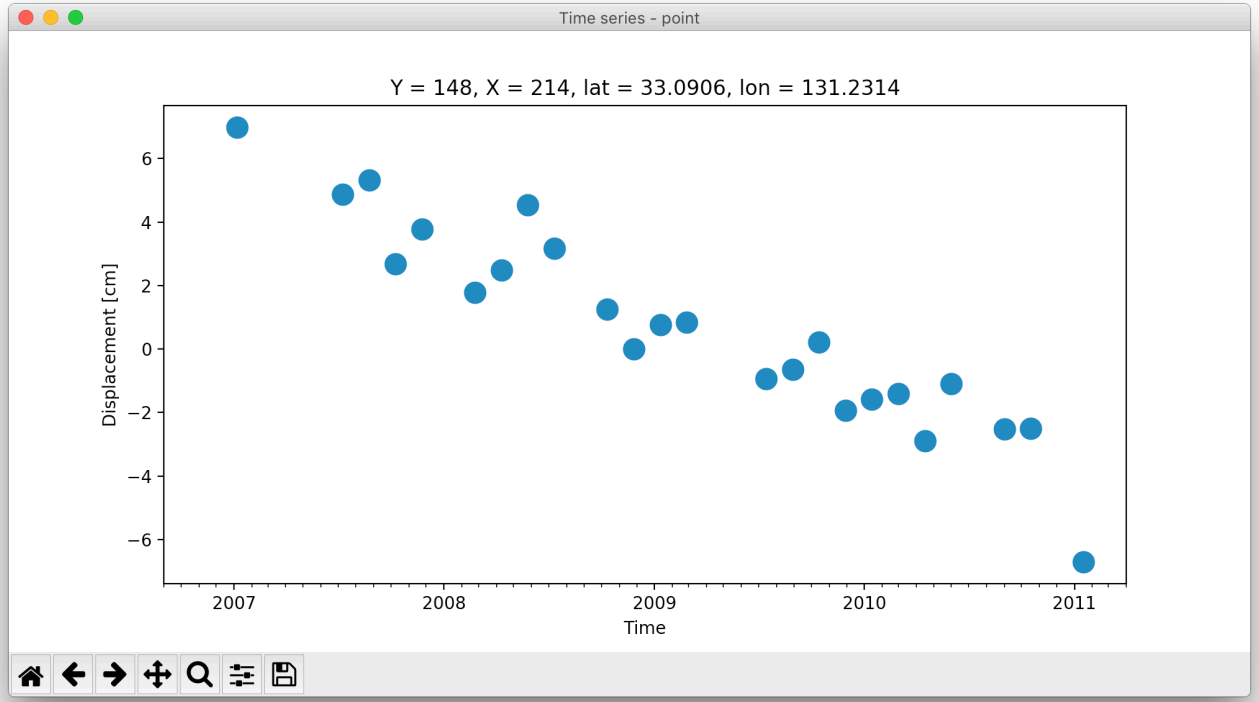


- Feature Highlights
  - DEM shaded relief background
  - Lat/lon label; scale bar
  - All colormaps from matplotlib
  - --save/--nodisplay to save w/n display
  - Custom figure/font size, output DPI
  - Display with different refence pixel/date on the fly
  - Check the current pixel coordinate / value on the status bar

```
view.py geo_velocity.h5 --mask geo_maskTempCoh.h5 -u cm
-m -2.5 -M 0.5 -c jet_r --lalo-label -d demGeo.h5
```

Play with plotting scripts – tsviewer.py





Feature Highlights:

1. Click on the left panel to plot its displacement history on the right
2. Drag “Time Slider” on the bottom of left panel to see the animation!
3. --save / --nodisplay to save the plot data on text file

```
tsviewer.py geo_timeseries_ECMWF_demErr_refDate_plane.h5 --mask geo_maskTempCoh.h5
```

Play with plotting scripts – plot_pysarApp.sh

- Check the default plotting results in PYSAR/PIC folder.
- plot_pysarApp.sh calls view.py to plot most of h5 file generated by pysarApp.py; change the plot setting within the file to adjust and save your plots in the style you like!

Re-run pysarApp.py

- After initial running and checking your results, you may want to adjust some parameters and re-run to update your result, PySAR makes it very easy.
- Change the option value in pysarApp_template.txt, then run pysarApp.py directly. You don’t need to delete nor remove files, and pysarApp.py will start from the changed step and continue.

What in PySAR but not in pysarApp.py

- PySAR is a toolbox. There are a lot of other scripts besides pysarApp.py. Check them in $PYSAR_HOME/pysar director. All scripts can be ran individually, except for scripts starts with _, e.g. _readfile.py.
- You can build you own processing recipe with it, check the example in $PYSAR_HOME/shellscripts
- You can import pysar as a python module below and develop your own program based on it, check the jupyter notebook example in $PYSAR_HOME/examples:

```
import pysar
```

- Check API Document for complete list of scripts and modules/functions

Have fun to play the InSAR time series!