CK Fusion Software User’s Guide

Author: Robert Granat

Introduction and Background

The *clustered kriging fusion* method was developed in order to combine remote sensing observations from two different instruments with potentially widely different instrument characteristics and sampling in time and space. This particular software uses that method to combine Earth surface displacement observations from two sources. One source is a network of fixed site stations utilizing the Global Navigation Satellite System (GNSS) to measure displacements at particular locations around a region of interest. Raw GNSS measurements are sampled at 1-2Hz depending on instrument, but are processed to produce more accurate displacement time series at each location on a daily basis. The other source is the UAVSAR mission instrument, an Interferometric Synthetic Aperture Radar device mounted on an airplane, which measures displacements between the time of two flight passes over some geographical region. UAVSAR observations are not made on any fixed schedule, and the coverage regions are determined by vehicle availability and targeting decisions made by the mission science team. Importantly, the GNSS observations are *three dimensional* displacement measurements in the East-West, North-South, and Vertical directions, while the UAVSAR displacement measurements are one dimensional along the *line of sight* direction from the instrument to the measured section of the Earth’s surface. The UAVSAR mission has carried out observations primarily in California, where there is a high density of GNSS measurement stations relative to the world average. Nevertheless, the spatial density of the measurements differ by orders of magnitude, posing a challenge to combining the two sources. The observational data from these two sources is publicly available, both through the individual mission data archives and via the GeoGateway science portal at https://geo-gateway.org/.

Kriging is a standard statistical approach to interpolating geospatial observations [ref], and has been widely used across many scientific fields. However, it operates on the fundamental assumption that the underlying physical state is continuous across the region of interest. In the case of Earth surface displacements, we know this assumption is violated as there are numerous discontinuities in the surface (e.g., earthquake fault boundaries). *Clustered kriging* [ref] is an approach that attempts to address this issue by automatically subdividing the observations into locally continuous clusters and then performing kriging based interpolation on each cluster separately. This approach has been applied successfully to GNSS displacement observations in the western United States. *Clustered kriging fusion* takes this one step further, using the clustered kriging GNSS results as input to a fusion method that combines GNSS and InSAR observations.

This software is the implementation of this approach for GNSS and UAVSAR observations. While it can be operated as stand alone software, it was designed to be integrated into the GeoGateway science portal with a web based user interface.

Software Components, Authorship, and Licensing

The software consists of several components, with a number of contributing authors. The components of this software are released under the Apache Software License V2, with **one important exception** (see below). The main contributing author is Robert Granat, who was responsible for integrating all the software components into a unified program. The software also makes use of open source software libraries from various authors external to the project. The components of the software and primary contributors are as follows.

- Clustered kriging for GNSS observations: Robert Granat
- InSAR and GNSS fusion: Robert Granat
- UAVSAR parsing and pre-processing: Jay Parker
- GNSS displacement and velocity processing: Michael Heflin
- Constrained clustering: a python implementation of the *COP K-means* algorithm was written by Behrouz Babaki (2017) [ref] and distributed publicly on Github under the MIT license. However, this package is no longer supported and can no longer be installed as a stand alone python package. As a consequence, the source code for this method was slightly modified and included as a part of the source of this package.
- UAVSAR line of sight calculation: developed by various authors in the Jet Propulsion Laboratory's radar section and provided courtesy of Brian Hawkins for use solely by the Quakes-A project. **This component is not released to the public under any sort of license and should not be used or redistributed without the explicit permission of its authors.** Persons wishing to utilize this component should contact Brian Hawkins directly at brian.p.hawkins@jpl.nasa.gov.

Any questions about the software should be directed to Robert Granat at granat@gmail.com.

Method Overview

Global GNSS sites provide daily three-dimensional position measurements which are precise at the few-mm level when data is processed using JPL's GipsyX software (Bertiger et al., 2020). This information can be used to study crustal deformation directly, help calibrate SAR baseline corrections, or be integrated with InSAR interferograms to improve their ability to resolve subtle, long-wavelength (e.g., tectonic) movements. This method takes processed GNSS displacements and clusters them into geophysically distinct groups using a Baysian gaussian mixture model clustering approach (Figure 1). Clustering of GNSS data is done based on the 3D displacement measurements at each station, thereby grouping stations on the basis of similarity of motion rather than their geographical distribution. As shown in (Granat et. al., 2021), natural geophysical boundaries (e.g., earthquake faults) between different clusters fall naturally out of this process (Figure 2). The number of clusters / sub-regions can be estimated from the data using Bayesian gaussian mixture modeling (Bishop, 2006; Attias, 2000; Blei, 2006). The clustered GNSS measurements are then interpolated with kriging algorithms (Heflin et al., 2020; Granat et al., 2021).

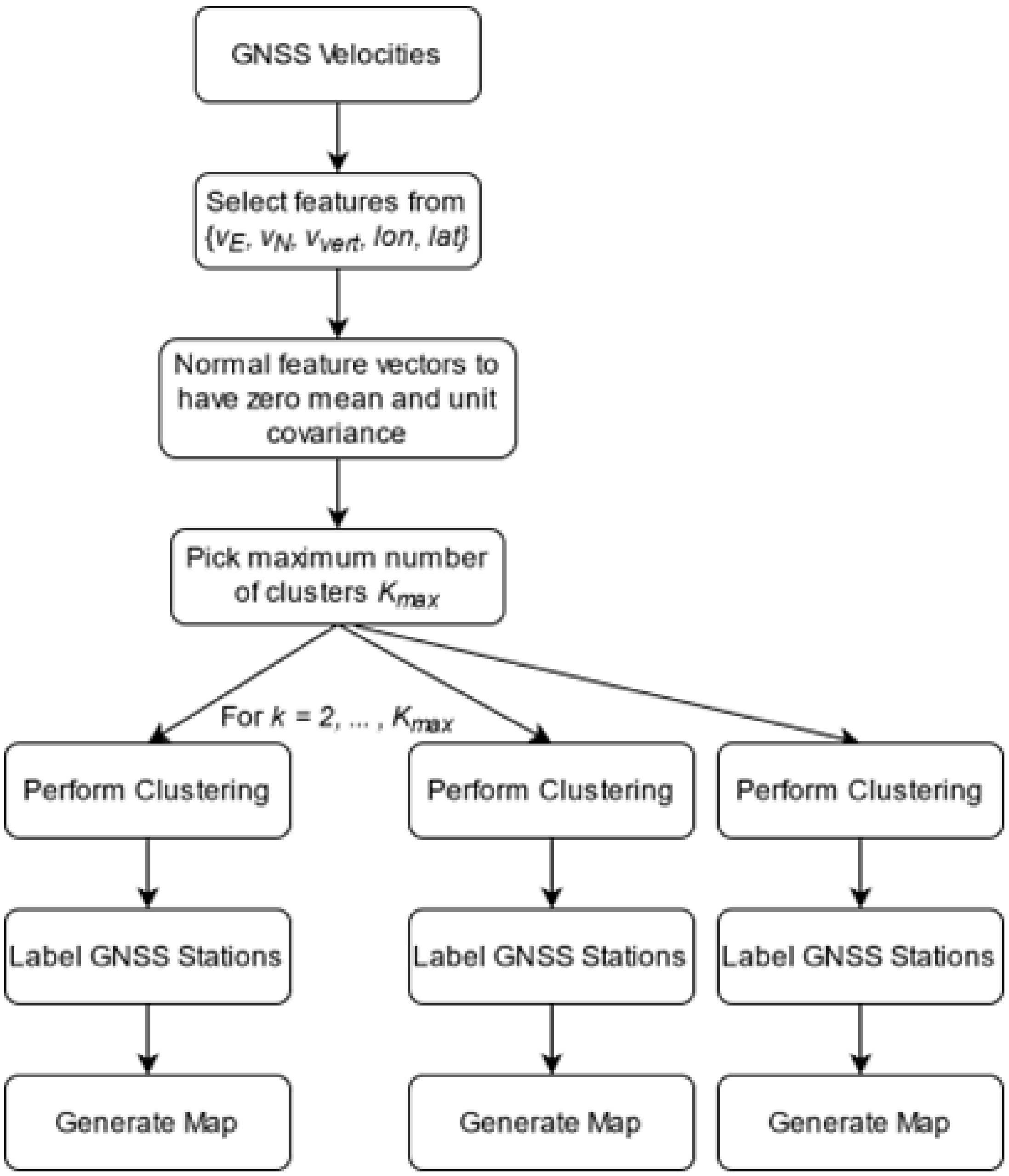
GNSS Velocities
Select features from $\{v_E, v_N, v_{vert}, lon, lat\}$
Normal feature vectors to have zero mean and unit covariance
Pick maximum number of clusters K_{max}
For $k = 2, \ldots, K_{max}$
Perform Clustering
Label GNSS Stations
Generate Map
Perform Clustering
Label GNSS Stations
Generate Map
Perform Clustering
Label GNSS Stations
Generate Map

Figure 1. Flow chart of the GNSS clustering process.

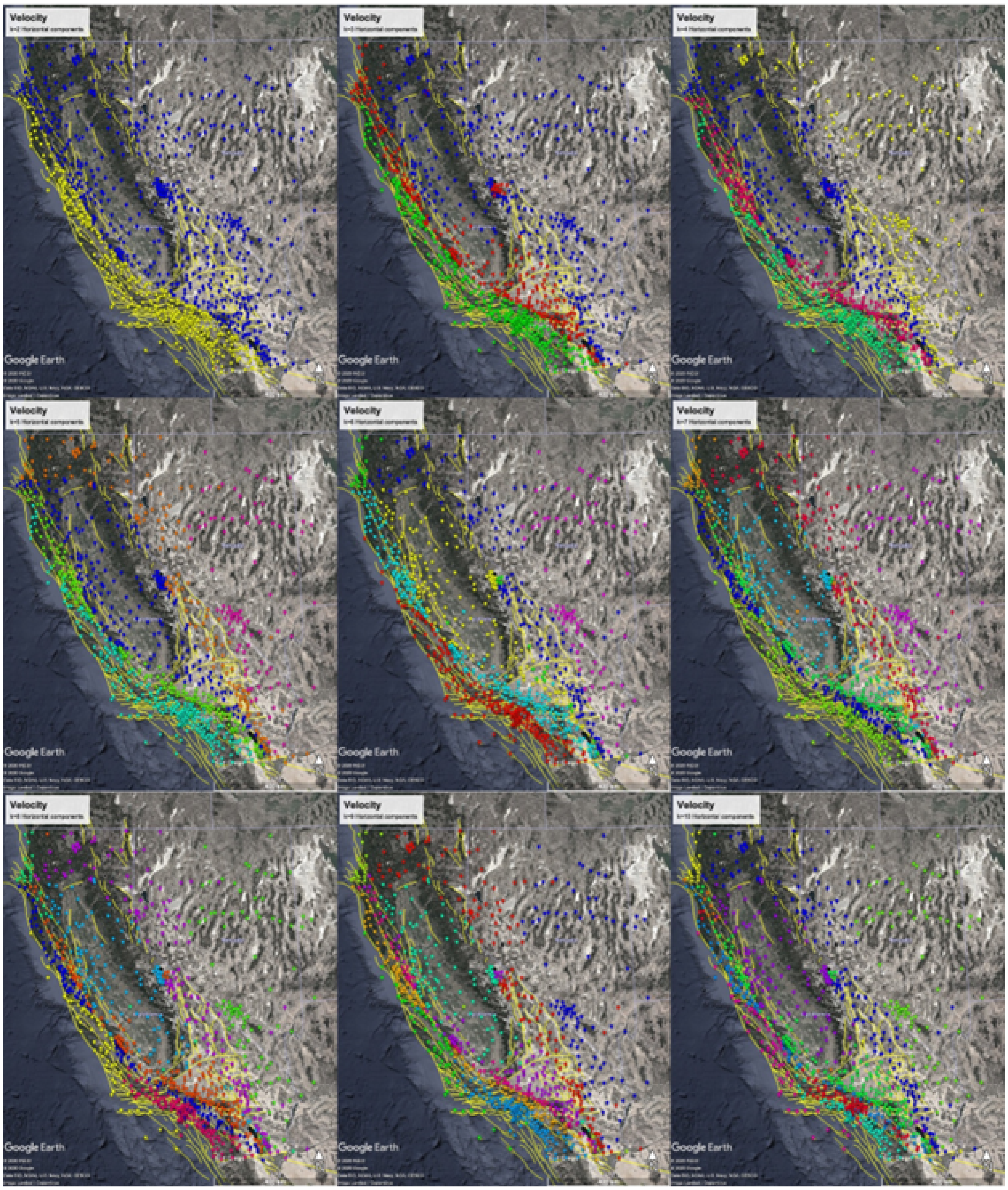

Figure 2. GNSS stations in California divided into color coded classes for two to ten classes.

GNSS stations are usually relatively spatially separated (~10 km on average, even in well-instrumented California). As a result, the mathematical boundaries between clusters as defined by the clustering method (e.g., Voronoi boundaries for a simple clustering method like *k-means*) sometimes fail to exactly match the natural geophysical boundaries due to the lack of spatial resolution. We can address this issue through a process of clustering refinement. If provided additional information in the form of a fault database, edges detected from satellite imagery, or similar sources, we can generate constraints on the clustering result that force the result to align with the geophysical boundaries as much as possible (Figure 3). Constraints take the form of “cannot link” point pairs, which must be placed in different clusters (e.g., because they are points on either side of a known fault), or “must link” point pairs, which must be placed in the same cluster (e.g., because we know they both belong on the same side of a known fault). In addition, we can impose constraints on the number of GNSS stations that must belong to each cluster, so that each sub-region contains sufficient station information to perform a valid kriging estimate of the displacement field. Constrained clustering (Basu, 2008) allows us to take these kinds of constraints and produce a refined clustering result that generates more accurate sub-regions.

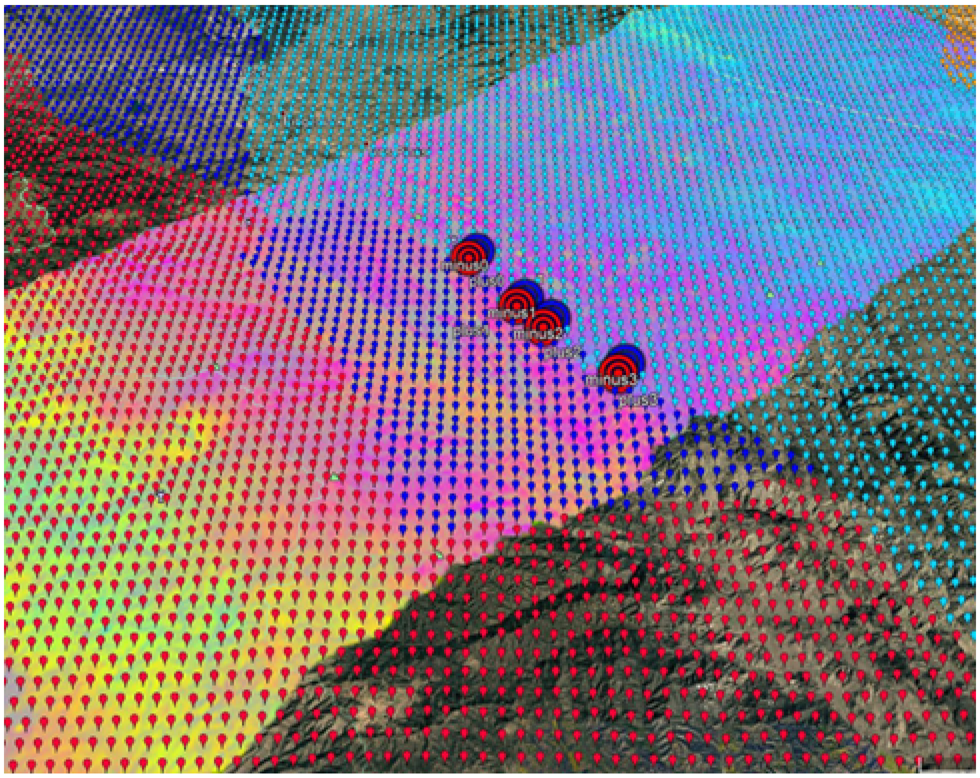

Figure 3: InSAR coordinate grid points classified according to regional GNSS observations. Boundary refinement achieved through application of “must link” (red/red or blue/blue) or “cannot link” (red/blue) constraints derived from image edge detection.

Once we have identified the sub-regions via clustering, we use ordinary kriging methods to take spatially sparse GNSS displacement measurements within each cluster and estimate the displacement field over a desired grid. The results from each sub-region are combined to create a complete estimate. For purposes of fusion with InSAR data, the desired grid is composed of the latitude, longitude coordinates of pixels in the InSAR swath, but we can also use this approach to generate interpolated displacement fields in areas and times not covered by InSAR measurement

Integration of GNSS and InSAR observations involves the projection of 3D measurements into the line of sight (LOS) direction of the InSAR measurement and inference out of the LOS direction into the 3D space. Unlike GNSS measurements, which provide a straightforward observation of displacement over a given time period in three directions of motion, an InSAR instrument only provides 1D displacement measurements in the LOS direction. The LOS direction is dependent on the location and orientation of the instrument at the time of collection as well as the surface topography, and is different for each pixel in the InSAR image. Fusion therefore requires that the GNSS measurements be projected into the varying LOS directions at each point in the final displacement field grid. Once the GNSS measurements are projected, the InSAR and GNSS measurements are placed in the same reference frame by selecting a stable reference point in the grid that has good coherence in the InSAR image and low estimate uncertainty in the GNSS interpolation. The aligned measurements are then combined by a weighted sum. These weights can be determined in an automated fashion by utilizing the relative uncertainties of each information source, or can be set as an input parameter to the algorithm. The combined LOS measurements are then projected back out of the LOS directions and into the standard 3D displacement coordinate system. Filtering steps are taken to remove noisy or otherwise faulty data from both the GNSS and InSAR sources. A flow chart of the fusion process can be seen in Figure 4.

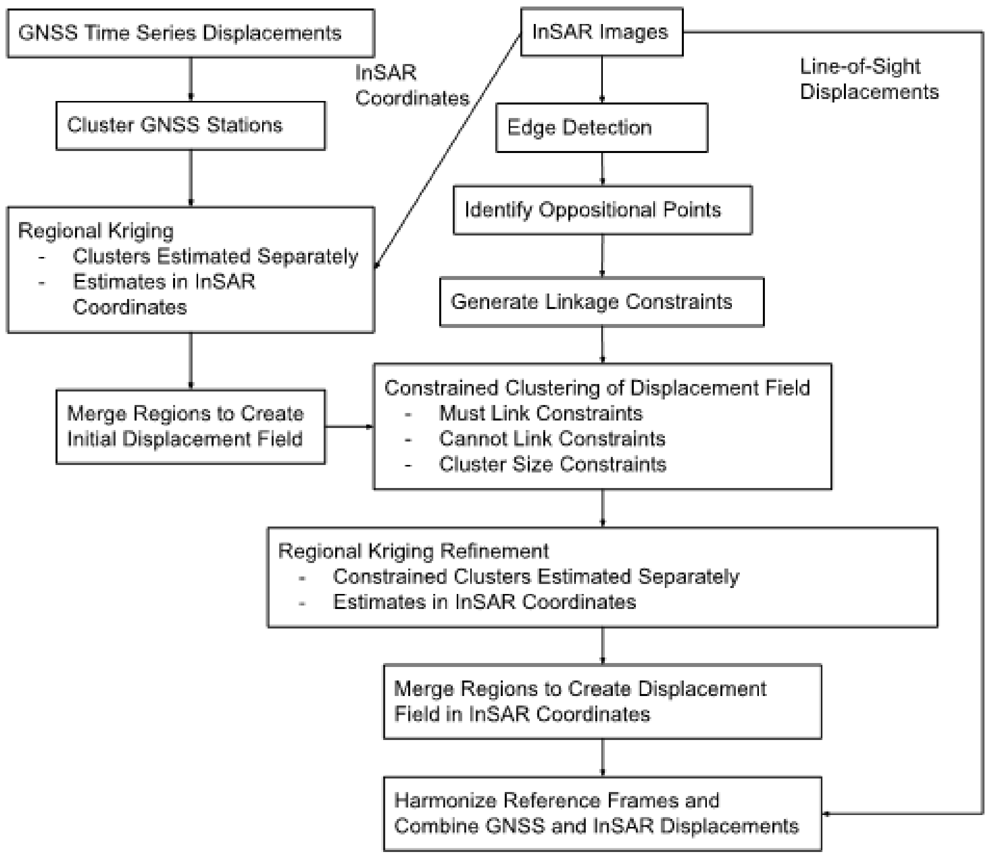
GNSS Time Series Displacements
Cluster GNSS Stations
Regional Kriging
- Clusters Estimated Separately
- Estimates in InSAR Coordinates
Merge Regions to Create Initial Displacement Field
InSAR Coordinates
InSAR Images
Line-of-Sight Displacements
Edge Detection
Identify Oppositional Points
Generate Linkage Constraints
Constrained Clustering of Displacement Field
- Must Link Constraints
- Cannot Link Constraints
- Cluster Size Constraints
Regional Kriging Refinement
- Constrained Clusters Estimated Separately
- Estimates in InSAR Coordinates
Merge Regions to Create Displacement Field in InSAR Coordinates
Harmonize Reference Frames and Combine GNSS and InSAR Displacements

Figure 4: A flow chart showing the steps needed to generate the fused GNSS / InSAR data product.

A full description of the algorithm can be found in a separate accompanying document.

Installation and Compilation

To install the python components, you first need to install Anaconda (https://anaconda.org/anaconda/python). Then, from a Unix or MacOS command line use the commands:

```
conda create -n geogateway_env
conda activate geogateway_env
conda config --env --add channels conda-forge
conda config --env --set channel_priority strict
conda install python=3 geopandas
conda install python=3 geopy
conda install python=3 pyKrige
```

To install the FORTRAN components needed for the line of sight computations, you first need to set your `ARCH` environment variable as appropriate for your OS. For example:

```
export ARCH=linux
```

If you don't have gfortran on your system you will need to install it:

```
apt-get install gfortran
```

Then, in the `make_geolook` parent directory, you need to use the command appropriate for your OS. For example:

```
make linux/dev
```

Please note that the `make_geolook` code base dates from 1997 and was not written with modern Fortran standards in mind. Depending on your OS and gfortran version, it may require modifications to the `Make.inc` file to successfully compile the executable. This release includes several example `Make.inc` files in the `COMPILE_OPTIONS` subdirectories with different configuration options that may help if the code does not compile on the first try.

As previously noted, the `make_geolook` software is not publicly licensed and should not be used without express permission of the appropriate group at JPL. It is intended that future versions of the CK Fusion code will remove any explicit dependency on this code base and instead take line of sight data files as input instead.

Usage

There are three stand alone components to software each of which can be run separately as a python executable. These are as follows.

- `get_uavsar_displacements:` Retrieves GNSS displacements over an InSAR epoch defined by the input UAVSAR annotation file. The use of this command is entirely optional, as GNSS displacements can be obtained in other ways. This command only retrieves measurements from a user specified geographical bounding box around the InSAR swath, and so is often unsuitable for use in areas with a relatively low density of GNSS stations (in which case you will need the information from more remote stations in order to estimate the GNSS displacement field with any degree of accuracy or certainty).
- `clusterKrigeVelocities:` A stand alone implementation of the clustered kriging algorithm that uses only the GNSS observations to generate an estimated field. Despite the name, this component can calculate displacement **or** velocity fields over the specified grid, depending on whether GNSS displacement or velocity observations are provided as input. It utilizes the clustered kriging algorithm described above.

- `CK_Fusion_GNSS_InSAR`: Implements the full clustered kriging fusion method for GNSS and UAVSAR displacement observations as described above. Internally, it makes a command line call to the separate FORTRAN executable `make_geolook`, but that should be transparent to the user.

Inputs and Outputs to `CK_Fusion_GNSS_InSAR`

```
usage: CK_Fusion_GNSS_SAR.py [-h] -input INPUT -output OUTPUT [-k K] -feature_name
NAME
                           [NAME ...] [-method METHOD] [--scale] [-lat_max LAT_MAX]
                           [-lat_min LAT_MIN] [-lon_max LON_MAX] [-lon_min LON_MIN]
                           [-lat_grid_space LAT_GRID_SPACE]
                           [-lon_grid_space LON_GRID_SPACE] [-lon_lat_file
LON_LAT_FILE]
                           [-constraint_file CONSTRAINT_FILE] [-insar_file
INSAR_FILE]
                           [-downsample DOWNSAMPLE] [-fullgrid]
```

`-input`

The name of the GNSS displacement observations file as produced by Mike Helfin’s various GNSS utilities.

`-output`

The name of the labeled output fused displacement file in KML format. The portion of the file name before the .kml extension will be used as the base name for other output files.

`-k`

The number of clusters used in the clustering step. For some clustering methods, such as Baysian Gaussian mixture modeling, this may be interpreted as the *maximum number* of clusters, and the final result may be less than this.

`-feature_name`

The GNSS features used in the clustering. This must be some subset of `{"Lon", "Lat", "Delta E", "Delta N", "Delta V", "Sigma E", "Sigma N", "Sigma V"}`, but the use of only `“Delta E” “Delta N” and “Delta V”` is recommended in most cases.

`-method`

The clustering method to be used. Must be one of `{k-means, affinity, meanshift, spectral, agglomerative, bdscan, gmm, bgmm}`. The use of `bgmm` is recommended. For more information about these options, refer to the Scikit-Learn documentation.

`--scale`

Boolean variable that determines whether to scale in the inputs to the clustering method to lie between 0 and 1. Recommended.

`-lat_min, -lon_min, -lat_max, -lon_max`

The maximum values of the corner points of the target displacement field. Only relevant if not interpolating in the reference frame of the InSAR image.

`-lat_grid_space, -lon_grid_space`

Spacing of the rectangular lat, lon target grid used for the target displacement field. Only relevant if not interpolating in the reference frame of the InSAR image.

`-constraint_file`

The name of the file containing pairs of (lat, lon) coordinates that define oppositional points. Only relevant if using boundary refinement via constrained clustering.

`-insar_file`

The name of the InSAR annotation file for the InSAR observation being used as input to the fusion. The name of the annotation file (`.ann` extension) is used to derive the names of other necessary InSAR input, which are the associated `.hgt.grd`, `.unw`, and `.unw.grd` files from that observation.

`-downsample`

The factor by which to downsample the InSAR swath target grid when creating the output. Downsampling improves computational performance and can be very useful for testing and visualization purposes. Recommended values of 10+ when doing any operation other than production runs.

`-fullgrid`

Calculate the displacement estimate over the full InSAR rectangle defined by the minimum and maximum (lat, lon) corner points rather than just the InSAR swath. More computationally intensive but provides estimates over a larger region.

Example commands:

Get Displacements:

```
./get_uavsar_displacements.py --lat 33 --lon 115 --width 1 --height 1
-insar_file SanAnd_05020_09007-014_10082-000_0636d_s01_L090HH_01.ann -o
displacement.kml
```

Clustered Kriging:

```
./clusterKrigeVelocities.py -input displacement_table.txt -output
labeled_velocity_05020_636d.kml -feature_name "Delta E" "Delta N" "Delta V"
-k 8 --scale -method bgmm -constraint_file oppoPairs05020.636d.txt
-insar_file SanAnd_05020_09007-014_10082-000_0636d_s01_L090HH_01.unw.grd
-downsample 100 -fullgrid
```

Fusion:

```
./CK_Fusion_GNSS_SAR.py -input displacement.kml_table.txt -output
labeled_velocity_05020_636d.kml -feature_name "Delta E" "Delta N" "Delta V"
-k 8 --scale -method bgmm -constraint_file oppoPairs05020.636d.txt
-insar_file SanAnd_05020_09007-014_10082-000_0636d_s01_L090HH_01.unw.grd
-downsample 1000 -fullgrid
```

Acknowledgements

This work was carried out at City College of New York and at the Jet Propulsion Laboratory, California Institute of Technology under contract with NASA. The work was funded by NASA's ACCESS, Advanced Information Systems Technology (AIST), Geodetic Imaging, and Earth Surface and Interior (ESI) programs.

References