GEFOLKI
Instruction manual

Object
Instruction Manual for using GeFolki in order to ensure remote sensing image coregistration
Author
Elise Koeniguer
Last Updated
2016, December

License and the respective conditions of working
The GeFolki module is distributed under the GPL license.
GeFolki is the subject of two different publications. If you use this code in a publication or scientific work, please quote one of the two publications below.

- In the context of the coregistration of two remote sensing images of the same nature (same band of the electromagnetic spectrum: SAR / SAR of the same band, optical / optical in the same channel, etc.), even if the resolutions are different: Images are said to be homogeneous. The publication to be quoted is then:


- In the context of the coregistration of two images of different natures (SAR / LIDAR, SAR-X / SAR band L, SAR / optical, two different optical bands, etc.), the images are considered to be heterogeneous. The publication to be quoted is then:


The contents of the instructions
GeFolki is a module allowing carrying out the coregistration of two images of remote sensing.
In this manual, a description is given about:

- Various modules to ensure the entire co-registration of two images: initialization, flow calculation (assured by GeFolki) and resampling.

- Recommendations on the choice of the different parameters involved in GeFolki according to the scenario envisaged.
In the most general way, the pixel matching of two images can be decomposed into three steps:

1. An initialization step
2. The flow calculation step (GeFolki)
3. A final resampling step

These three steps and their input-outputs are described in the diagram above.

1. The initialization step takes as input the two images of different sizes to be co-registered. The output is the second slave image transformed into an image of the same size as the master image: \( n_{x1} \times n_{y1} \).

2. The flow calculation step is ensured by the code GeFolki.m. This step takes as input two images of the same dimensions. The output \( W \) is a three-dimensional matrix corresponding to the flow with size \( (n_{x1} \times n_{y1} \times 2) \).
   - \( W(:, :, 1) \) corresponds to the component of the flow computed on the horizontal axis.
   - \( W(:, :, 2) \) corresponds to the component of the flow computed on the vertical axis.

3. The third step consists in the re-sampling of the slave image following the flow matrix previously calculated. This step has two inputs: the image \( I_2' \) and the flow matrix \( W \). The output is the resampled image \( I_2'_{\text{resampled}} \) that is co-registered to the image \( I_1 \).
**DESCRIPTION OF THE FIRST STEP : INITIALIZATION**

Consider that I1 et I2 are the two images to be registered with respective dimensions (number of rows x number of columns) : \((n_{x1} \times n_{y1})\) et \((n_{x2} \times n_{y2})\).

The initialization step consists in putting the images roughly in the same geometry, with pixels of the same size, and the same number of pixels \((n_{x1} \times n_{y1})\).

Image I₁ is considered as the master image and image I₂ as the slave image.

The first operation is to apply a matlab command that defines the matrices X and Y where X gives the row number and Y the column number:

\[
[Y,X]=\text{meshgrid}((1:n_{y1}),(1:n_{x1}));
\]

With this convention, and for the slave image presented above with size 7 × 9, the matrices X and Y are:
The initialization consists in applying simple transformations (rotation, homothety) to image $I_2$ or general transformations, generally determined by the taking of tie-points.

The result is to have a slave image that has passed from the coordinate system $(X_2, Y_2)$ to the coordinate system $(X'_2, Y'_2)$ with the same dimensions as the master image. Therefore, this step is to have two images of same size and same geometry which corresponds to the requirements for the flow calculation.

There are two mainly different configurations:

- Either the images were acquired in the same geometry (two SAR images in interferometric conditions, VNIR and SWIR images). In this case, the transformation to be applied is reduced to a simple "crop" and / or translation. The slave image is not re-sampled.

- Or the images were acquired by sensors with different headings / resolutions. Then the transformation gives rise to a resampling of the slave image.

For the second case, the desired transformation is much more complex and it can be obtained by different methods whose choice depends of the configuration:

**First case: simple transformation**

In some cases, the transformation may correspond to known simple transformations: a homothety (only the sampling intervals of images are different) or a rotation (headings of the images differ).
A scaling can be done with the imresize matlab function, in case I1 and I2 are already in the same geometry:

```matlab
Nx1=size(I1,1);
Nx2=size(I1,2);
l2prime = imresize(I2, [nx1,ny1]);
```

For a rotation, we apply the matlab function: imrotate

**Second case: a complex transformation**

For a more complex transformation, an initialization step is based on the selection of corresponding points between the two images and after on the estimation of a global transformation.

The selection of points into the images is done with the matlab function: cpselect

```matlab
cpselect(I1, I2);
```

This operation leads to two lists of coordinates, those of the first image (master image) stored in the variable "input_points" and those of the corresponding points in the second image (slave image) stored in the variable "base_points".

From these corresponding coordinates, the matlab command « cp2tform » is used to calculate the transformation that does the initial correspondence between pixels from one image to the other:

```matlab
t = cp2tform(base_points,input_points,'affine');
```

Then, image I2 is transformed into an image I2' with the matlab command "imstransform" in according to the transformation previously estimated:

```matlab
Nx1=size(I1,1);
Nx2=size(I1,2);
I2p = imtransform(I2, t,'nearest','XYScale',1,...
    'XData', [1,dimy],...
    'YData', [1,dimx],...
    'FillValues', eps);
```

**Third case: initialization by geocoding**

Finally, it is possible to assign to each image a grid of latitudes and longitudes for each pixel, obtained from all the georeferencing data (sensor trajectories and DTM) and then to resample the image I2 in the referential of image I1.

Because these operations are specific to each sensor, they are not described in this document.
DESCRIPTION OF THE MAIN STEP : FLOW CALCULATION BY GEFOULKI

This step takes as input two images $I_1$ and $I_2'$ of the same size.

The flow describes the transfer from coordinates $(X_2', Y_2')$ to $(X_1, Y_1)$ by a matrix $W$ of size $(n_{x1}, n_{y1}, 2)$.

The first component of $W$ is the y-displacement (column) and the second one is the x-displacement (row) to apply on the slave image.

![Diagram showing the flow of coordinates](image)

The GeFolki function is applied on two images $I_1$ and $I_2'$ of the same size. The output is a matrix $W$ corresponding to the flow matrix.

Below, you have a structure “para” that contains the parameters of the algorithm. This input is optional. If it is not specified, a default setting is provided.

```matlab
type = struct('radius', 32:-4:8, ...
              'levels', 6, ...
              'iter', 2, ...
              'contrast_adapt', false, ...
              'rank', 4);
```
1. The first parameter $r = \text{para}.\text{radius}$ is the radius parameter. The algorithm can be launched for several radius sizes in an iterative manner. We usually start from the biggest radius to the smallest one. For this reason, it is defined as a decreasing vector.

This radius $r$ determines the size of the window $(2r + 1) \times (2r + 1)$ on which the correlation between the two images is maximized. Its choice results from a compromise between robustness and the level of detail for the flow:

- A large radius makes the algorithm more robust
- When the flow rapidly changes on the image, the radius must be chosen small enough to estimate these variations

2. The second parameter $L = \text{para}.\text{levels}$ is the number of levels in the scale pyramid. It must be conditioned by the maximum size of the displacement to be estimated: $W_{\text{max}} < 2L$.

3. The third parameter $K = \text{para}.\text{iter}$ is the number of iterations used in the gradient method dedicated to the minimum search. Typically, this parameter varies between $K = 2$ and $K = 10$.

4. The parameter $\text{para}.\text{contrast}_\text{adapt}$ allows the algorithm to look for a contrast inversion. It must be active for heterogeneous images, and set to “false” for homogeneous images (SAR interferometry, etc.)

5. Parameter $R = \text{para}.\text{rank}$ is the parameter used in the rank filter. A typical value of the parameter is $R = 4$. (9x9 window)

Once the structure has been defined, the algorithm can be launched with the command:

$W = \text{GeFolki}(I_1, I_2, \text{para})$;
LAST STEP : RESAMPLING

Last operation is to transform the slave image to superpose it to the master image.

From the flow matrix $W$, this operation consists in resampling the images on the new grid of coordinates. Several methods of interpolation are possible: bilinear, linear, nearest neighbors, etc.

data1=W(:,1); 
data2=W(:,2); 
I3=(interp2(x',y',I2p',x'+data2',y'+data1','nearest'))';