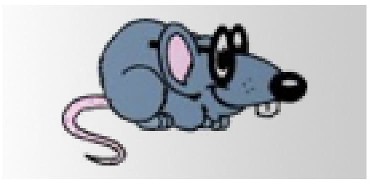
RAT User Guide

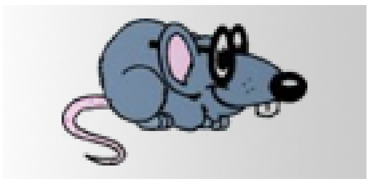
Version 0.18
18. February 2007



RAT User Guide

Version 0.18

18. February 2007



Contents

Chapter 1

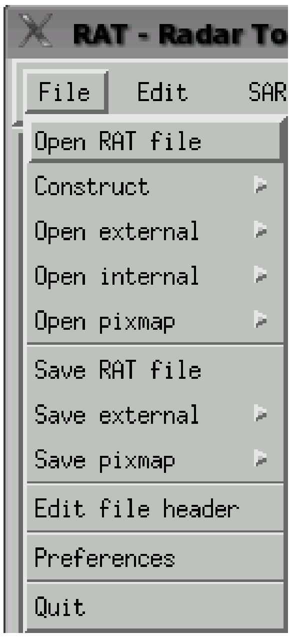
Overview

RAT is developed in IDL [?].

Official publications about RAT [?, ?, ?].

Chapter 2

The FILE menu



Use the ***File*** menu on the RAT menu bar to read files into RAT, save file, construct polarimetric or interferometric images, import or export into different data formats, set preferences, edit the file header of the RAT file format and exit RAT.

2.1 Open RAT file

Use the “Open RAT file” to open data saved in RAT format.

2.1.1 RAT file format

RAT uses a special file format, containing information about data type, size and representation. Until some import filters are available, you’ll have to convert your data to RAT format before beeing able to work with it. This is quite simple if you know how to use IDL. If not, have a look at the format description. In fact, the RAT format is just a simple header plus the data in binary representation.

2.1.2 How to generate a file in RAT format with IDL

Quite easy:

- Enter IDL
- Compile RAT or restore the IDL save-file. This makes a routine for writing RAT files available (srat).
- Read your data into an IDL variable using your own rotines. You should know how....
- Write the RAT file with the command `srat,"file.rat",var` on the IDL command line. ’file.rat’ denotes the RAT file to be generated, ‘var’ the variable in which contains your data.
- Now you can start RAT and load the content of the file with File>Open RAT file.

2.1.3 RAT format description

A RAT-file is structured as in the following:

Element	**type**	**Description**
DIM	1*long integer	Number of dimensions of the array
SIZE	DIM*long integer	Array sizes for each dimension
VAR	1*long integer	Type of Array (1=byte, 2=int, 3=long, 4=float, 5=double, 6=complex, 9=double complex)
TYPE	1*long integer	Type of Data (100 = SAR amplitude image, 101 = SAR complex image, etc.... (see definitions.pro)
DUMMY	3*long integer	Reserved
INFO	80*byte	Comment string
ARRAY	Array type	The array itself

2.2 Construct

2.2.1 InSAR pair

2.2.2 PolSAR pair

2.3 Open external

2.3.1 E-SAR (DLR)

2.3.2 ENVISAT-ASAR

2.3.3 Radarsat-2

This routine is made for importing RADARSAT-2 data in Geo-TIFF format into RAT. Up to now, it has been only tested using simulated data - RADARSAT-2 is not yet in space!

Basically, this routine has only two options: Reading quad-pol data, i.e. data composed out of the four polarisation HH,VV,HV and VH, or single-pol data. Since RAT does not support partial polarimetry, dual-pol data are threated as two individual single-pol files. When reading quad-pol data, only one of the four files has to be selected in the fileselector. RAT will automatically find the 3 others. When not all the 4 files are found, RAT switches back to single-pol.

Many thanks to Daniel De Lisle from CSA for providing a CD with simulated sample data.

2.3.4 PolSARPro

2.3.5 Generic binary

2.3.6 ENVI standard

2.4 Open internal

2.4.1 rarr / sarr

2.4.2 2*long + complex

2.4.3 E-SAR-RK (Rolf)

2.5 Open pixmap

2.5.1 Open PNG

2.5.2 Open JPEG

2.5.3 Open TIFF

2.6 Save RAT file

2.7 Save external

2.7.1 ENVI Standard

2.7.2 Generic binary

2.8 Save pixmap

2.8.1 Save PNG

2.8.2 Save JPEG

2.8.3 Save TIFF

2.9 Edit file header

2.10 Preferences

2.11 Quit

Chapter 3

Single-channel SAR

3.1 Textures

3.1.1 Variation coeffiecient

This function calculates the variation coefficient of an amplitude or intensity SAR image. The variation coefficient is defined as

$$var = \frac{variance(x)}{mean(x)^2} \tag{3.1}$$

and is a measure of the amount of local gray-value variation. In homogeneous regions, its value is idependent of the local image amplitude; edges appear emphasized.

RAT allows to set the size in x and y of the box used for estimation of variance and mean.

3.1.2 RFI filter

This routines (tries) to filter out so-called radio frequency interferences (RFI). RFI come from received energy, which was not emitted by the SAR (mobile phones, radio beacons, etc).

You have to set precisely some system parameters. RAT assumes the near range to be on the left side of the displayed image! That's important. If the system parameters are not precise, you can expect very stange filtering results.

After starting the detection and waiting a bit, an image is shown, where potential RFI appear as "white" spots. You can now use the sliders to adjust the notch filter. Ideally, it should cancel out all the white spots (but not the rest). There are two filter parameters: The detection threshold sets the sensitivity of detecting interferences. A value of 2.0 means that all RFI having an energy of larger then 2.0 times the background are detected. The filter strength sets the radius of the notch filter used for removing the detected RFI. A value of 3 means that a circle of radius 3 around the RFI peak is removed. You'll have to press 'update window' if you want to see what'll happen when using the actual filter parameters.

When you think your settings are fine, press "start filtering"....

3.1.3 Co-occurence features

Chapter 4

Polarimetric Interferometry (PolInSAR)

4.1 Data Types

RAT can handle polarimetric interferometric images. A POLInSAR image pair is stored in a single file, in pixel interleave format. The dimensions of data are: N_x, N_y — image dimensions; N_{pol} — number of polarimetric channels (3 or 4), N_{intrf} — number of baselines (only 2 baselines are supported).

- *PolInSAR scattering vector*
$$data = [N_{intrf}; N_{pol}; N_x; N_y]$$
- *PolInSAR covariance matrix*
$$data = [N_{2pol}; N_{2pol}; N_x; N_y] \qquad N_{2pol} = 2 * N_{pol} = N_{intrf} * N_{pol}$$

4.2 Inspect

4.2.1 Coherence Analysis

A variety of methods is used to visualize the POLInSAR coherence behavior in dependence of averaging kind and quantity, computation methods, optimization parameters, random coherence projections, etc. Compare [?, ?, ?, ?, ?].

Chapter 5

SAR Interferometry (InSAR)

RAT can handle interferometric image pairs. An interferometric image pair is stored in a single file, in pixel interleave format.

5.1 Coregistration

Interferometric image pairs are usually not coregistered. Reasons can be the baseline between the images, different range delays, shifted image frames etc. As even a small misregistration drastically degrade the coherence of an interferogram, before further processing a precise coregistration has to be performed.

5.1.1 Global offset

This routine can shift the whole slave image by a constant offset. The offset has to be an integer number, i.e. subpixel shifts can not be performed with this routine. Enter the offset values in the upper box and press 'OK' to shift the slave image.

If you don't know the image offset, you can use the "automatic offset estimation" to estimate it. The default values in the boxes should be o.k. for most cases. You have to change them only if you have images with very large offsets. Shortly after "start estimation" has been pressed, the estimated values appear in the upper box. Press "OK" to shift the slave.

For offset estimation, the method of amplitude correlation is used.

5.1.2 Subpixel offset

Very much the same as function 5.1.1. A constant shift of the entire slave image is performed, this time allowing floating point values as offsets. For shifting, cubic interpolations are used. It is not necessary to use this function when it is planned to use "array of patches" or similar afterwards.

The estimation is based on oversampled amplitude correlation is used. It is not possible to estimate very large offsets. In this case use function 5.1.1 before.

5.1.3 Array of patches

This function estimates image offsets on a grid of small patches distributed over the image. Then, a 2D-warping function is applied on the slave image for coregistration. This function is useful if the offsets are not constant over the image, which is often the case in airborne repeat-pass images or spaceborne images with strong topography.

In the first box, the number of patches and their size can be specified. Don't select a too small size of the patches, since this will make the results unreliable. Press "OK" to start the estimation.

After a while, a second box will appear showinh the vector field of offsets. With the 2 edit buttons you can eliminate by hand obviously wrong estimates. When everything looks good, press "Start coregistration" to warp the slave image (time consuming!). Oversampling factors of 2 or 4 can help to better preserve image quality.

For offset estimation, oversampled amplitude correlation is used. For warping, cubic convolution, is desired combined with FFT based oversampling, is used.

5.2 Phase noise filter

Phase noise filtering is used to enhance the quality of the interferometric phase. It is usually employed prior to phase unwrapping, but can also be useful for other purposes.

5.2.1 Boxcar

Boxcar filtering is a very simple filter: It is just a local averaging over the give rectangular box. Becuase of phase wrapping effects, averaging is performed in the complex domain, even if only a floating point phase is provided.

5.2.2 Goldstein

The Goldstein filter is a spectral filter which enhances dominant fringe components in a local box. This is achieved by multiplying the fringe spectra with its own amplitude power a certain filter parameter (spectral exponent). Goldstein filtering is very powerful in relatively high coherent regions, but might fail in low coherent areas.

In RAT, the window size is fixed to 64x64 pixels with an overlap of 32 pixels.

5.2.3 GLSME

Another InSAR specific fringe filter. GLSME filtering minimises globally the phase noise fluctuations by a least-squares approach. GLSME seems to be very powerful especially for very low coherent regions surrounded by high coherent areas.

5.3 Coherence

5.3.1 Boxcar

5.3.2 Gauss

5.4 Remove flat-earth

5.4.1 linear

5.4.2 from file

5.5 Shaded relief

5.6 Transform

5.6.1 pair to interferogram

5.6.2 extract amplitude

5.6.3 extract phase

Chapter 6

SAR Multichannel Subapertrues

6.1 Data Types

The dimensions of data are: N_x, N_y — image dimensions; N_{ap} — number of subapertures; N_p — number of vector elements in InSAR-, POLSAR-, POLInSAR- vectors.

- *Subapertures* Single channel subapertures

$$data = [N_{ap}; N_x; N_y]$$

- *Multi-channel Subapertures* Multichannel subapertures (e.g. for interferometry)

$$data = [N_p; N_{ap}; N_x; N_y] \qquad N_p = N_{intrf} = 2 \text{ for interferometry}$$

- *Polarimetric Subapertures* Multichannel subapertures for polarimetry

$$data = [N_p; N_x; N_y] \qquad N_p = N_{pol} = \{3; 4\}$$

- *POLInSAR Subapertures* Multichannel subapertures for POLInSAR

$$data = [N_p; N_x; N_y] \qquad N_p = N_{polin} = N_{pol} * N_{intrf} = \{6; 8\}$$

  POLIn-channel order = $[pol_0 intrf_0;\ pol_1 intrf_0;\ pol_2 intrf_0;\ pol_0 intrf_1;\ pol_1 intrf_1;\ pol_2 intrf_1]$

- *Covariance matrices for every Subaperture*

$$data = [N_{p^2}; N_{ap}; N_x; N_y] \qquad N_{p^2} = N_p^2 = \{N_{pol}^2; N_{polin}^2\}$$

- *Subapertures covariance matrix*

$$data = [N_{apcov}; N_{apcov}; N_x; N_y] \qquad N_{apcov} = N_p * N_{ap}$$

- *Subapertures stationarity [log(L)]* from [**?**]

$$data = [N_x; N_y]$$

6.2 Nonstationarity

Evaluates the stationarity over azimuth subapertures. The result is *Subapertures stationarity [log(L)]*. The nonstationary subapertures can be removed to generate a more stationary SAR image. Compare [?].

Chapter 7

Developer's Guide

This section is meant as a reference for the developers of RAT, as well as for those of you, who would like to understand better how RAT was programmed and how to modify it by yourself.

7.1 Developer's Conventions

Just to start to propose and to write down some conventions. At the current stage they are all open for discussions.

- The channels, as presented to the user, start with channel **1** (baseline 1, data set 1, polarization 1). Contrary to the internal array handling of IDL, where the first array index is **0**.

7.2 Import Template

7.2.1 Block Processing (Tiling)

7.3 Data Parameters Handling (Metadata)

This section answers the questions:

- How can I get access to parameters, resp. read/write them?
- How do I include new parameters.
- Where should I get the real parameters from?

1. Define the parameter in the parameter structure **parstruct** in **definitions.pro**. The name of the entry is also the name of the parameter, followed by an NIL–Pointer type. Don't forget to add some comments about your parameter. Also think of a good name!

2. For changing a parameter use the function **set_par()**. You have to provide the name as a string and the value(can also be an array). As a result you get a status flag, which you should check. If you get 0, then everything is ok.

3. For getting a parameter use the function **get_par()**. See details above.

7.4 Speckle Filter Wizard

This section answers the questions:

- How do I include my newly written speckle filter into the Wizard.
- I have changed some parameters in my speckle filter. What do I have to modify in the Wizard?

Following modifications need to be considered:

1. Put the name of the filter into the string–array **filter_type**. Remember the position of your filter (index in the array).
2. In dependence of the data type (POLINSAR, POLSAR, others) you should either *offer* or *not offer* this speckle filter to use. Set the right flag in the byte–array **offer**.
3. Write generic fields for input in the widget at the right position.
4. Call you speckle filter with the right parameters. Compere the fields you defined for the widget.

Example

1.
```
filter_type = [...,'MY-FILTER',...]  ; at position Z
```

2.
```
POLINSAR --> offer = [...,1,...]  ; at position Z
POLSAR   --> offer = [...,0,...]  ; at position Z
OTHERS   --> offer = [...,0,...]  ; at position Z
```
with it this filter is allowed only for POLINSAR data

3.
```
Z: begin                 ; MY-FILTER
   field[0,i]=CW_FIELD(filter_grp[i],VALUE=50,/integer,TITLE='Parameter1 : ',XSIZE=3)
   field[1,i]=CW_FIELD(filter_grp[i],VALUE=25,/integer,TITLE='Parameter2 : ',XSIZE=3)
```

4.
```
;; MY-FILTER
   Z: my-filter,/called,par1=*field_value[0],par2=*field_value[1]
```

And that's it!

7.5 Batch processing with RAT

This section describes the possibilities of batch processing with RAT. The batch processing should provide the possibilities for

- fast data processing from the shell without graphical output (which can be very time consuming in case of big data sets).
- writing of IDL batch programs for automatic and fully operational SAR data processing with RAT procedures.
- shell-only radar data processing, without any mouse.

7.5.1 Basic Concepts

General processing chain:

1. Open IDL shell
2. Compile RAT

   ```
   .compile rat
   ```

3. Start RAT in the batch mode with

   ```
   rat, /nw ;;   nw for NoWindow
   ```

   - As usual with RAT, if a data set with the name ”default.rat” can be found in the working directory, RAT opens it automatically.
   - Alternatively, one can provide a file name to open at the start with

     ```
     rat, /nw, startfile="my_data.rat"
     ```

4. One can open a new data set with

   ```
   open_rat, inputfile="my_data.rat"
   ```

5. And, similarly, one can save the current working data set with

   ```
   save_rat, outputfile="my_modified_data.rat"
   ```

6. The current data set can also be exported to an IDL variable, and used further without interaction with RAT:

   ```
   export_var, variable_name
   ```

7. At any moment, the GUI can be shown, and hidden again, which makes it possible to switch between batch processing, and the graphical representation:

   ```
   show_gui
   hide_gui
   ```

8. To exit RAT (batch processing or GUI), write

   ```
   exit_rat
   ```

In a similar fashion, one can import data from external file formats, apply various processing functions, and export it again.

It is of importance to provide all the required parameters to the functions. To find out, which parameters these are, please see this documentation, or consult the IDL source codes of the functions.

It is also important to use the keyword "/CALLED" when calling procedures from the shell, which act on the data.

Note: This mode is still at an experimental stage. Therefore, it is possible that for some untested procedures, some windows will pop up. Also, some functions are not recommened to use in the batch mode, for instance, there is no reason to call a zoom procedure within the batch processing.

In the following, two examples of RAT batch scripts are presented:

1) Simple steps to open a PolSAR data, to do speckle filtering, and to decompose into Entropy/Alpha/Anisotropy. The data is then exported to an IDL variable, which can be further examined, even after exiting RAT.

```
.compile rat ;; compile RAT
rat, /nw, startfile="some_polarimetric_C_or_T_matrix_data.rat"
speck_polreflee, smm=7, looks=9, /CALLED
decomp_entalpani, /CALLED
show_gui ;; just to see, what did come out
hide_gui ;; hide the GUI again
export_var, HAA   ;; export the data into the given variable "haa"
exit_rat ;; quit rat batch processing
help,HAA   ;; shows the structure of the still existent variable "haa"
```

2) Advanced automatic processing: A function to construct a MB PolInSAR data set from a given set of PolSAR data files. Chain flow: construct the MB data set, remove the flat earth phase, conduct adaptive range spectral filtering, construct a coherency matrix, apply simultaneously a specified presuming, and do speckle filtering with the Refind-Lee filter. The file is finally saved in the rat format.

```
pro mbpolinsar_data_processing, polsar_files, fe_file, outputfile, $
                                smmx=smmx, smmy=smmy, Lee_box=box
  rat,/nw
```

```
    construct_polinsar, /CALLED, FILES=polsar_files
    rrat, fe_file, fe
    polin_rgflt_adaptive, /CALLED, fe
    polin_k2m, /CALLED, SMMX=smmx,SMMY=smmy
    polin_c2t
    speck_polreflee, /CALLED, SMM=Lee_box, LOOKS=smmx*smmy
    save_rat,outputfile=outputfile
    exit_rat
  end
```

This function can then be used for automatic data processing, e.g. with:

```
  .compile rat
  mbpolinsar_data_processing, ["track1.rat","track2.rat","track3.rat"], $
      "fe_1x2x3.rat", "mb_tracks_1x2x3.rat", smmx=3, smmy=9, Lee=7
```

These two examples should only show the possibilities of batch processing with RAT.

7.6 Mini Howto on RAT-SVN via Linux shell

Attention! To have a direct access to the subversion repository, as desribed in the following, you need the status of a RAT–developer on the server.

Checkout of the whole trunk from the server to the current directory:

```
  svn checkout https://YOUR_NICK@svn.berlios.de/svnroot/repos/radartools/trunk
```

The following operations can be executed in the trunk directory or any subdirectory.

Current status:

```
  svn status -u
```

Update the local copy from the server:

```
  svn update
```

Commit your changes (do ALWAYS add useful comments to the changes after $-m$):

```
  svn commit -m "Change Log: ..."
```

Add a new file to the repository (only locally, to upload it to the server, use the commit command!)

```
svn add new_file_name.pro
```

Further repository file manipulation commands:

```
svn delete ...
svn copy ...
svn move ...
svn mkdir ...
```

7.7 Building a Binary for the Release

IDL:

```
.compile rat
resolve_all
save,filename="rat.sav",/routines
```

Pack together with *icons* and *preferences* into a single archive.