Rack – a program for anomaly detection, product generation, and compositing

Markus Peura
Finnish Meteorological Institute, Helsinki, Finland (first.last@fmi.fi)

(Dated: May 30, 2012)

1 Introduction

Rack is a free command-line program for processing weather radar data. It reads sweeps or volumes in HDF5 file format as input and generates single-radar products as well as composite products. Rack also provides anomaly detection and removal (AnDRe) functions. In addition, Rack supports producing output in image formats and provides related utilities such as applying color palettes and transparency.

Rack is a light-weight option for operational or research environments running Linux/Unix as it only needs HDF5, PNG and PROJ.4 libraries as external dependencies. It has been implemented in C++ to replace RoPo, an anomaly detection program developed in C and operated in the Finnish Meteorological Institute since 2001 (Peura 2002). Recent development work has been also supported by the BALTRAD project, making the code freely available under the GNU Lesser General Public License. Rack uses OPERA Data Information Model in HDF5 structures (Michelson et al. 2011).

This technical paper illustrates the main functionalities of Rack and discusses its performance and future development.

2 Installation

The program code is retrievable at `http://baltrad.fmi.fi/software/rack/download.html`. The page contains links to installation zips of Rack (executable) and Drain (library). A user's manual is also included in the package.

3 Basic usage

Rack is a command-line program and reads its arguments in their order of appearance. Plain arguments (without a leading hyphen) are handled as input volume files. Separate sweeps can be combined simply by issuing file names sequentially. Other arguments are treated as commands to be executed or as variable assignments. A typical invocation consists of reading input file(s), executing commands, and outputting results:

```
rack <input-files> <command> -o <output-file1> \
     <command2> -o <output-file2> ...
```

Notice that after writing a product file one may continue with further commands. This is practical in generating varying end products computed from one input. Rack internally updates and references three radar data structures: input data volume, intermediate polar product and Cartesian product. The data flow inside Rack is shown in Fig. 1.

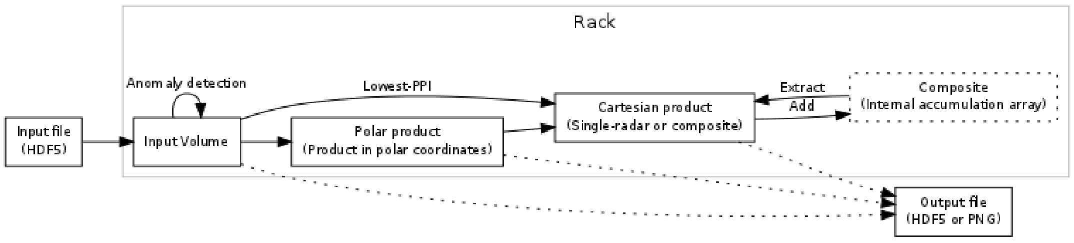
Rack
Anomaly detection
Lowest-PPI
Input file (HDF5)
Input Volume
Polar product (Product in polar coordinates)
Cartesian product (Single-radar or composite)
Extract
Add
Composite (Internal accumulation array)
Output file (HDF5 or PNG)

Figure 1: The overall data flow inside Rack.

4 Anomaly detection and removal

The set of anomaly detection and removal functions of Rack is called AnDRe. The detection is based on conventional ie. non dual-polarimetric data. As radar users know, echoes received by a weather radar originate not only from precipitating clouds but also from insects, birds, aircraft, ships, ground, sea clutter and electromagnetic emitters. Recently, especially developers of numerical weather prediction have reported quality problems caused by non-meteorological echoes.

In AnDRe, the processing is divided into two stages: detecting anomalies and removing them from input data. These stages modify ODIM data independently so a user can combine Rack processing with other software. Detectors for speckle noise, biometeors, ships and emitter lines have been implemented. Computations apply fuzzy functions that map results to continuous truth values between 0.0 (no) and 1.0 (yes). The following code is an example of applying both detection and removal in Rack:

```
rack volume-anomalous.h5 \
 --aStoreDetections \
 --aBiomet 0dBZ,500m,5dBZ,500m \
 --aEmitter 5000m,3.0deg,0.7  --aEmitter 20000m,5.0deg,0.7 \
 --aShip 20dBZ,10dBZ,1500m,3deg \
 --aSpeckle -10dBZ,5 \
 --aEraser 0.25,0.25 --aGapFill 1500,5
 -o volume-corrected.h5
```

Examples of results are illustrated in Fig. 2 and Fig. 3.

The anomaly removal is a separate stage with two complementary options: erasing values and/or replacing values with gap filling which uses neighboring values in the extent they have higher quality.

5 Basic products

5.1 Products in polar coordinates

Rack computes single-radar products internally using a polar coordinate system. The resulting products are likewise in polar coordinates, typically processed further to Cartesian single-radar or composite products. Motivations for computing intermediate products in polar coordinates:

- Polar coordinates are the native system for radar; by default a polar product is computed in the same azimuthal resolution as the measurement data, which means that azimuthal interpolations are avoided and more information is hence preserved.

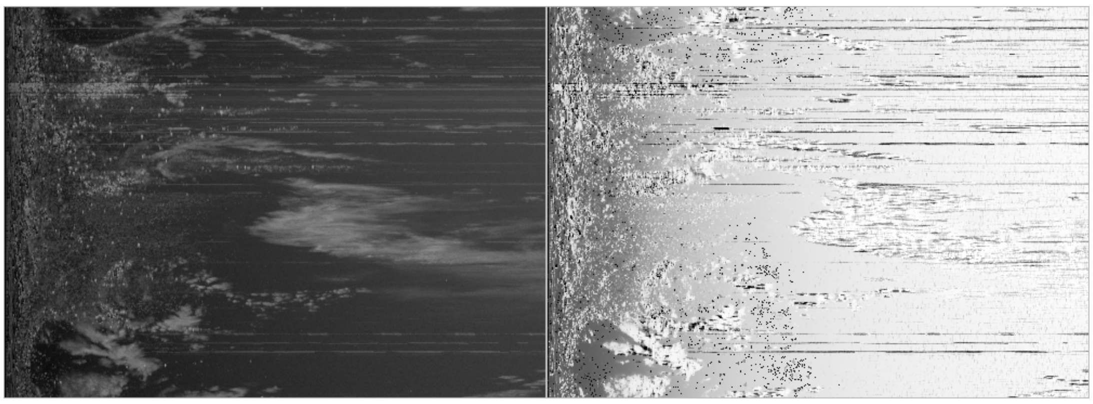

Figure 2: A sample scan containing several types of anomaly (left) and overall quality (right) based on anomaly detections. Dark areas indicate lower quality.

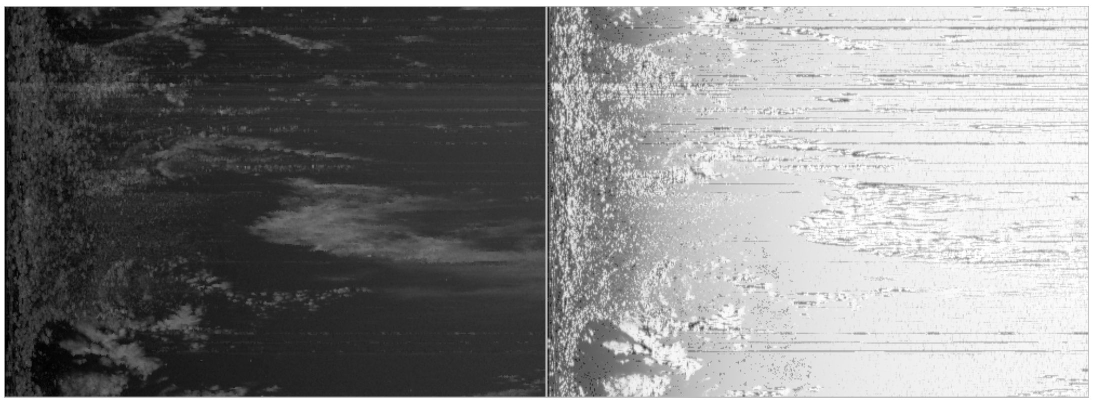

Figure 3: The sample scan after anomaly removal (left). The quality (right) is increased in anomalous locations.

- Speed is gained if several end products are to be computed in varying geographical projections.
- Adding new meteorological algorithms needs minimal code development.

As an example, computing a constant-altitude position indicator (CAPPI) is obtained with `--cappi` command. It takes altitude as its first argument.

```
  rack volume.h5 --cappi 500m  -o cappi.h5
```

Other, optional arguments are number of bins, number of rays, bin resolution, intensity gain and intensity offset. Otherwise default values of these ODIM attributes are applied and stored in the resulting HDF5 structure.

Also maximum intensity and echo top/bottom products have been developed.

5.2 Conversion to Cartesian coordinates

Data in polar coordinates can be mapped to a Cartesian product with `--cCreate` (abbreviated `-c`). For example, a Cartesian cappi can be generated with

```
rack volume.h5 --cappi 500m --cCreate -o cappi.h5
```

By default, the azimuthal equidistant projection is applied. The projection can be changed with `--cProj` using PROJ.4 syntax (Evenden and Warmerdam n.d.), including EPSG codes.

5.3 Image outputs

Writing output images in Rack is easy: replacing output filename extension `.h5` with `.png` causes Rack to store data as an image. The default data for output is the first dataset in the hdf5 hierarchy. One may tune the contrast with `gain` and `offset`. The following command creates a grayscale, an RGB and an RGBA image with one run (See Fig. 4):

```
rack volume.h5 --cappi 1500m -c  -o erad-cappi-bw.png   \
  --palette palette-dbz16.txt    -o erad-cappi-rgb.png  \
  --imageAlpha 0.2,-32.1,1,100   -o erad-cappi-rgba.png
```

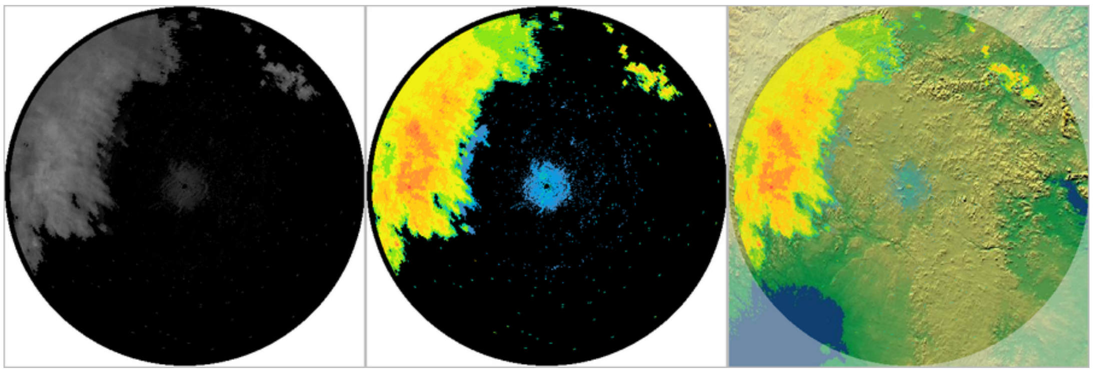

Figure 4: A CAPPI product mapped with `--cCreate` *(left), further coloured with* `--palette` *(center), made transparent with* `--imageAlpha` *and superposed on a map with* `convert` *program (right).*

6 Compositing

In dBZ composite products, there are several compositing principles: maximum, average, maximum-quality and quality weighted, including nearest-radar. Currently, the traditional backward (surjective) mapping has been implemented but a forward (injective) mapping is under development, too. It is coarser but faster and hence suitable for e.g. on-demand services. Internally, compositing uses a cumulative array from which the final intensities and qualities are extracted to the end product. The applied method also provides the radar count and standard deviation for each pixel.

An example of a large composite is shown in Fig. 5. Essentially, a compositing command consists of three parts: initialization, addition of radar data, and extraction of the result. Something like this:

```
rack  --cProj '+init=epsg:4326' --cBBox -12,39,28,61 --cSize 1200,800  --cMethod WAVG,1,1 \
  volume1.h5  [optional andre commands]  --cCreate  --cAddWeighted 1.0  \
  volume2.h5  ...
  ...
--cExtract=dwsC -o composite.h5 [optional other outputs]
```

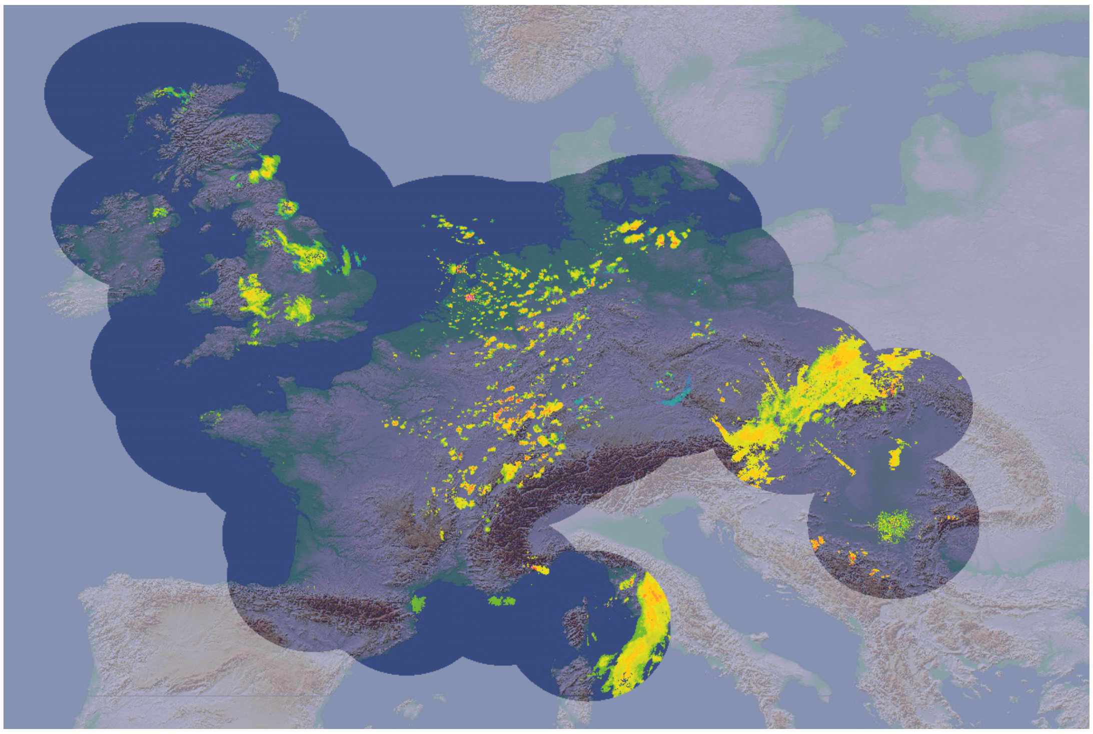

Figure 5: A 1200x800 pixel composite of 52 European radars.

A more covering support for quality algebra demonstrated earlier (Peura and Koistinen 2007) and distributed, continuous, "living" compositing (Peura 2010) will be implemented in future versions of Rack.

7 Other functionalities

With rack, one can easily convert ODIM HDF5 to new ones applying different storage type (unsigned char, float, double,...) as well as `gain`, `offset`, `undetect` and `nodata` values. Future versions may support reading and writing BUFR files, too.

In operational processing chains, one often needs to generate also text output about processed radar data and products. Such output might be log entries, www pages (HTML) for process monitoring, metadata text files (txt), graphical end products (SVG) or geographic definition files (KML). Such text output can be generated with `--format`, `--formatFile`, `--formatOut` commands.

Sometimes one should know whether a geographical area is within the range of a radar. For example in compositing one should handle (uncompress, convert, transfer, read) only radars which have overlap with the area. This can be checked with `--cBboxTest` command.

8 Performance issues

In programming, several dedicated techniques have been applied in pursuit of good accuracy and processing speed. Using polar coordinate system in anomaly detection and basic product generation was mentioned already above. Smoothing and other window based operations apply moving windows which are updated incrementally. Sequential single pixel operations apply direct iterations in memory, not two-dimensional loops. At the edges of image arrays, coordinate handlers guarantee that the coordinates are properly wrapped or limited. In flood-fill oriented operations a semi-recursive technique is applied, combining a non-recursive horizontal traversal with a recursive vertical traversal.

We tested the speed of Rack in the compositing case of Fig. 5 involving 52 European radars. The time required for generating the composite was about 7 seconds inclucing anomaly detection (5 detection and 2 removal operations), hence less than 0.15 seconds per radar. See Table 1. In the test set, the average sweep size was about 110500 bins ($\approx 307 \times 360$ sweep).

File I/O	**Anomaly detection**	**Compositing**	**Total**
0.91 sec	5.23 sec	1.04 sec	7.2 sec
15%	73%	13%	100%

Table 1: Total time requirements in creating a 1200x800 composite of 52 radars (see Fig. 5).

References

Evenden, G. and Warmerdam, F. (n.d.). PROJ.4 cartographic library. http://trac.osgeo.org/proj/.

ImageMagick Software suite (n.d.). http://www.imagemagick.org/.

LGPL (2007). GNU lesser general public license, version 3. http://www.gnu.org/licenses/lgpl.html.

Michelson, D. B., Lewandowski, R., Szewczykowski, M. and Beekhuis, H. (2011). EUMETNET OPERA weather radar information model for implementation with the HDF5 file format, version 2.1, *Opera working document WD_2008_03*, EUMETNET OPERA.

Peura, M. (2002). Computer vision methods for anomaly removal, *Second European Conference on Radar Meteorology (ERAD02)*, Copernicus Gesellschaft, pp. 312–317.

Peura, M. (2010). The living composite, *Proceedings of the Sixth European Conference on Radar in Meteorology and Hydrology (ERAD2010)*, Vol. 1 (Advances in Radar Applications), pp. 350–354.

Peura, M. and Koistinen, J. (2007). Using radar data quality in computing composites and nowcasting products, *33rd Conference on Radar Meteorology*, American Meteorological Society. CD-ROM.

Acknowledgement

Development of Rack has been partly supported by BALTRAD (2009-2012) "An advanced weather radar network for the Baltic Sea Region: BALTRAD" (Baltic Sea Region Programme 2007-2013), part-financed by the European Union (ERDF and ENPI).

The author wishes to thank the colleagues at FMI, especially Seppo Pulkkinen and Harri Hohti for support and suggestions.