Pyrad architecture and principles

Jordi Figueras i Ventura, Daniel Wolfensberger
AMS Radar conference 2023 - Open Radar Short Course

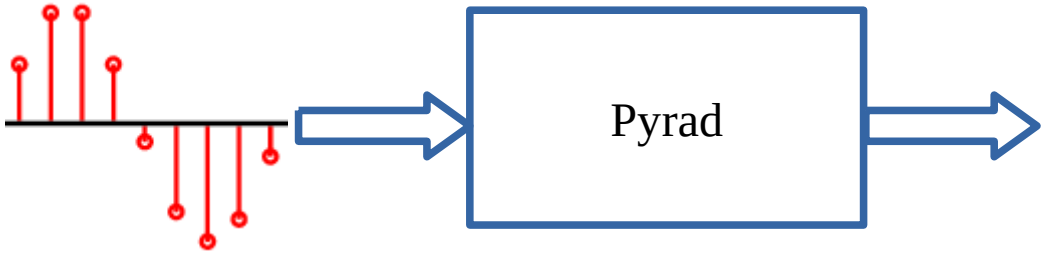
Contents

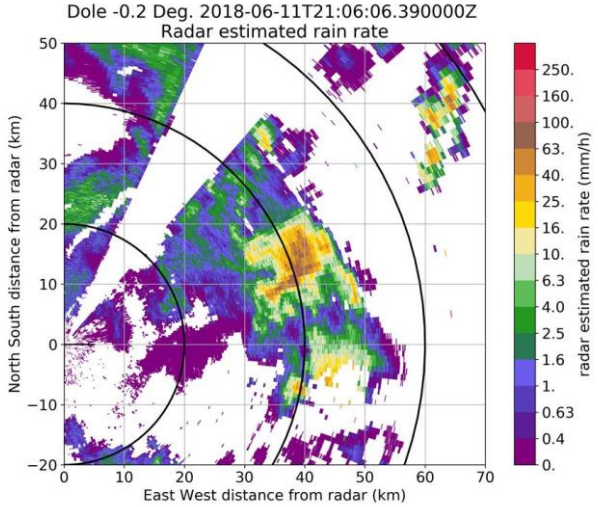
- Introduction
- Pyrad working philosophy
- Launching Pyrad
- Constructing the processing chain
- Config files : an example

1. Introduction

What is Pyrad ?

- Pyrad is a processing framework that allows the creation of flexible and replicable data processing chains with no programming. It is capable of operating in **real time** or **off-line.** It is aimed mainly for weather radar data processing but has some limited functionality allowing to process data from other sensors.





General characteristics

- Python-based (> v. 3.7)
- Linux platform
- Open source, version controlled (**https://github.com/MeteoSwiss/pyrad**)
- Core based on our **own version** of **ARM-DOE Py-ART** (The Pyrad project contributes back regularly)
- Possibility to ingest data from multiple radars
- Ingests multiple data types: IQ data, spectral data, polarimetric and Doppler moments, Cartesian data, etc.
- Capable of reading the main file formats used for volume radar data storage
- **Automatic documentation published online** based on doc-strings
- Easy to install (PyPI, conda)

2. Pyrad working philosophy

General characteristics

KEY CONCEPT : Separation between **dataset** generation and **product** generation

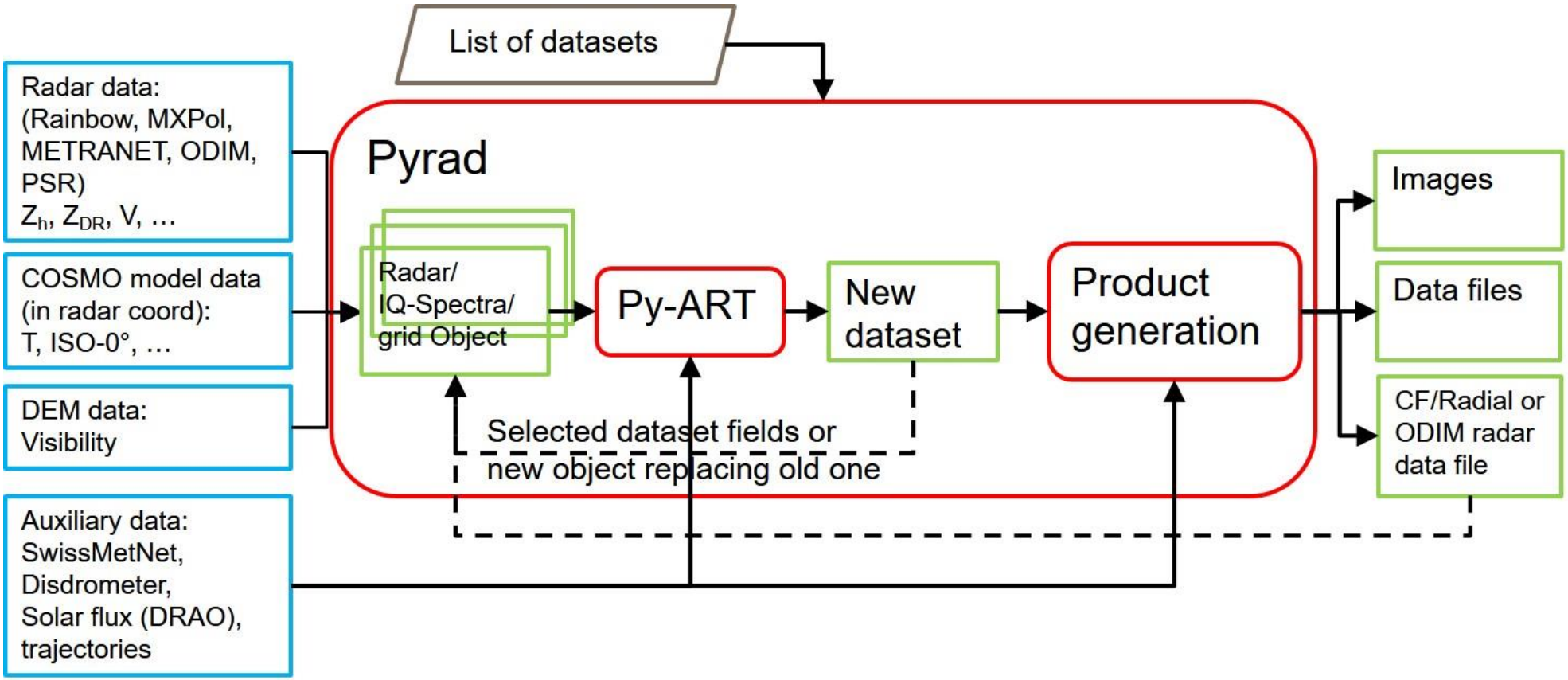
Dataset : New data generated from the processing of a Pyrad data object. It can be in the form of a new field of the same object type or a completely new object. It can be re-ingested in the data processing chain. Examples :

- rainfall rate field generated from a reflectivity field contained in a radar object
- Reflectivity generated from a spectral data object

Product : Output generated out of a dataset for human or machine consumption. Examples :

- PPI of reflectivity
- File containing timeseries of values at a point of interest

Pyrad flow diagram



Configuration files

Main config file	• Base paths of data • Location of config files • Base path for output data and image format(s)
location config file	• Radar(s) name and scan list • General radar and scan characteristics (scan periodicity, radar constant, …) • Images configuration

product config file	• List of datasets to generate
	• For each dataset
	  • List of inputs
	  • Dataset specific configuration
	  • List of products to generate
	  • MAKE_GLOBAL
	  • SUBSTITUTE_OBJECT
	  • FIELDS_TO_REMOVE

Internally all the configuration parameters are stored in a dictionary cfg
From cfg pyrad creates :
- datacfg : necessary parameters to read the input data
- dscfg : parameters to create the datasets (one per dataset)
- prdcfg : parameters to create products (one per product)

Pyrad config files examples

3. Launching Pyrad

Pyrad processing status

Status 0 : Initialization of datasets

Status 1 : Sequential processing of input data. Persistent data and parameters are stored internally

Status 2 : All input data consumed. Final dataset production if necessary

Pyrad Launching scripts

main_process_data.py	Process data sequentially according to starttime and endtime defined in command line or by input file
main_process_data_period.py	Process available data within time intervals specified by user between dates specified by user. Useful to obtain daily statistics
main_process_data_rt.py	Real time data processing

6. Constructing the processing chain

Constructing the processing chain

- Each dataset type is identified by a **keyword** that internally is associated to a **processing function**
- Internally the datasets are grouped in **families**. Those families can generate similar products
- The user can define up to 99 **levels of processing** for each **input data object**
- Datasets that are not inter-dependent can be generated at the same processing level. There is an option to generate those in parallel
- Levels will be processed sequentially. At the end of each processing level the keywords ***MAKE_GLOBAL, SUBSTITUTE_OBJECT*** and ***FIELDS_TO_REMOVE*** will control the behaviour of the new dataset created
- For each new dataset as many **products** as specified by the user will be generated. There is an option to generate those in parallel. If desired no product needs to be generated

Constructing the processing chain

MAKE_GLOBAL	1 - The newly generated dataset fields will be added to the data object and will be available for the next processing level. Assumes that the generated dataset is compatible with the current data object. If a field with the same name exists it will be overwritten 0 – The dataset will not be added to the data object
FIELDS_TO_REMOVE	List of fields to remove. The fields removed will not be available for the next processing level. Useful to remove intermediate fields to reduce the memory footprint
SUBSTITUTE_OBJECT	1 - The data object used at the current level will be erased and substituted by the newly generated dataset. Useful for e.g. transition from a volume radar object to a grid

Dataset families. Most common

VOL	generate_vol_products	Radar volume output
TIMEAVG	generate_time_avg_products	Radar volume time averaging products. Same products as above but with different time stamp
SPECTRA	generate_spectra_products	Spectra or IQ volume output
GRID	generate_grid_products	Cartesian grid data ouput
GRID_TIMEAVG	generate_grid_time_avg_products	Time-averaging or accumulation of grid data. Same products as above but with different time stamp
QVP	generate_qvp_products	Time-column data (e.g. QVP, columns, etc.)
TIMESERIES	generate_timeseries_products	Time series of POIs products

List of Datasets
List of Products

Identification of data fields

- Internally pyrad uses the naming convention of Py-ART which follows closely the **CF/Radial convention** for radar data fields.
- There are some fields that are non-standard and therefore defined by Pyrad
- The Py-ART names used internally are defined in the **Py-ART config file**
- Since the Py-ART naming is very long, Pyrad uses short keywords in the Pyrad config files. The short keywords are mapped internally using the function get_fieldname_pyart in io_aux.py
- The special keyword « all_fields » allows to load all data saved in a Pyrad generated volume

List of data fields

Generation of images

- Standard radar images (PPI, CAPPI, RHI, etc.) are generated using the **plotting functions from Py-ART**
- Standard Cartesian images (Surface, cross-sections, etc.) are generated using the **plotting functions from Py-ART**
- Other images (e.g. B-scope) and graphics (e.g. time series) are generated by Pyrad itself
- Colormap and value limits, field name to show, units, etc. are generally defined at the **Py-ART config file**. Pyrad allows using a discrete colormap
- The image size, resolution and area plotted are defined by ***ImageConfig*** structs in the pyrad loc config file
- Images can be overplot on a map generated by cartopy

Generation of images

ppiImageConfig	PPI-like image configuration
rhiImageConfig	RHI-like image configuration
ppiMapImageConfig	PPI-like image overplotted on a map configuration
gridMapImageConfig	Cartesian grid image overplot on a map configuration
sunhitsImageConfig	Sun hits image (delta_az, delta_ele)


Thank you!
Grazie mille!
Moltes Gràcies!
Merci!