D2.5 New release of OS processing package

Version 1.0

Working Group	WG2
Deliverable	D2.5
Date	08/10/2025
Version	1.0
Author	Christian Chwala
Reviewer	Aart Overeem

Description	This deliverable is a report on the final state, at the end of the Action, of the OS processing software packages that were developed during OpenSense.
Keywords	software, methods, open-source

Table of contents

Funded by
the European Union

www.cost.eu
www.opensenseaction.eu

Glossary

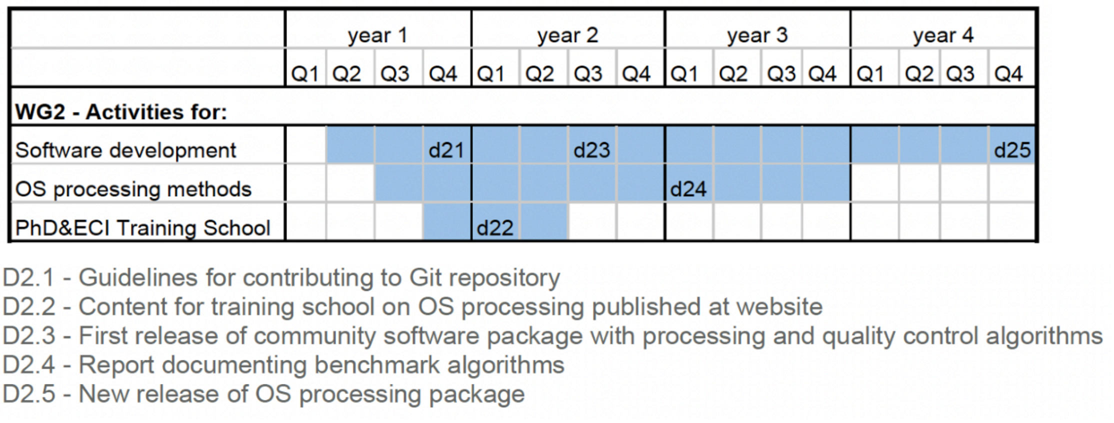
WG	Working Group
MoU	Memorandum of Understanding
OS	Opportunistic Sensors
CML	Commercial Microwave Link
PWS	Personal Weather Station
SML	Satellite Microwave Link

1. Overview of WG2 tasks and synchronisation with D2.5

This document reports on the official OpenSense deliverable D2.5, which is part of the activities carried out by WG2. These activities around collaborative software development started in GP1 and continued throughout the duration of the Action.

D2.5 is the result of WG2's work on the homogenization of processing methods and the corresponding software tools. It builds on the work done for D2.1 and D2.3 which has resulted in the joint development of three new software packages. These WG2 activities and deliverables are shown in Table 1.

Table 1. A timetable of WG2 activities and deliverables.



	year 1				year 2				year 3				year 4			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
WG2 - Activities for:																
Software development		🟦	🟦	🟦 d21	🟦	🟦	🟦 d23	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦 d25
OS processing methods			🟦	🟦	🟦	🟦	🟦	🟦	🟦 d24	🟦	🟦	🟦				
PhD&ECI Training School				🟦	🟦 d22	🟦										

D2.1 - Guidelines for contributing to Git repository
D2.2 - Content for training school on OS processing published at website
D2.3 - First release of community software package with processing and quality control algorithms
D2.4 - Report documenting benchmark algorithms
D2.5 - New release of OS processing package

2. Assessment of the preexisting software packages and decision on new developments

Here we give a brief overview over the process that led to the development of the new OpenSense software packages that are presented in more detail in the next section. During GP1 we assessed the current state of the existing software packages used for OS data processing by reviewing the included methods, the documentation (or lack thereof) and accessibility (ease of installation). In addition, we made a selection of the packages available in the OpenSense software sandbox https://github.com/OpenSenseAction/OPENSENSE_sandbox, where examples can be run online in a pre-configured containerized application in the cloud. While this gave a good overview of the available methods and their usage it also showed that interoperability of the packages was poor. Instead of improving interoperability of several small, partially outdated and suboptimally

Funded by
the European Union

www.cost.eu
www.opensenseaction.eu

COST (European Cooperation in Science and Technology) is a funding agency for research and innovation networks. Our Actions help connect research initiatives across Europe and enable scientists to grow their ideas by sharing them with their peers. This boosts their research, career and innovation.

structured, software packages, we decided to extract common functionality into one central software package. On top of this central software package - which we named *poligrain* because it deals with point, line and gridded rainfall data - we planned to have the software packages that provide the actual methods for OS data processing and merging of sensor data. For CML data processing, it was decided to transfer the existing methods from the R package *RAINLINK* to the existing Python package *pycomlink*, which already contained a large number of processing methods and which had a functioning interface with *poligrain* and the WG1 data format conventions, which *poligrain* relies on. In addition, we decided to start a new software package called *pwpwsqc* to which we planned to transfer the existing quality control methods available in R and Python from the OpenSense community. In collaboration with WG3, which focuses on sensor data merging, we started the software package *mergeplg* as a collection of merging methods with point, line and grid data. Applications can then be, and already are, built with a combination of *poligrain* and the packages that provide data processing methods, as shown in the overview of the envisioned ecosystem of the OpenSense software packages in Figure 1.

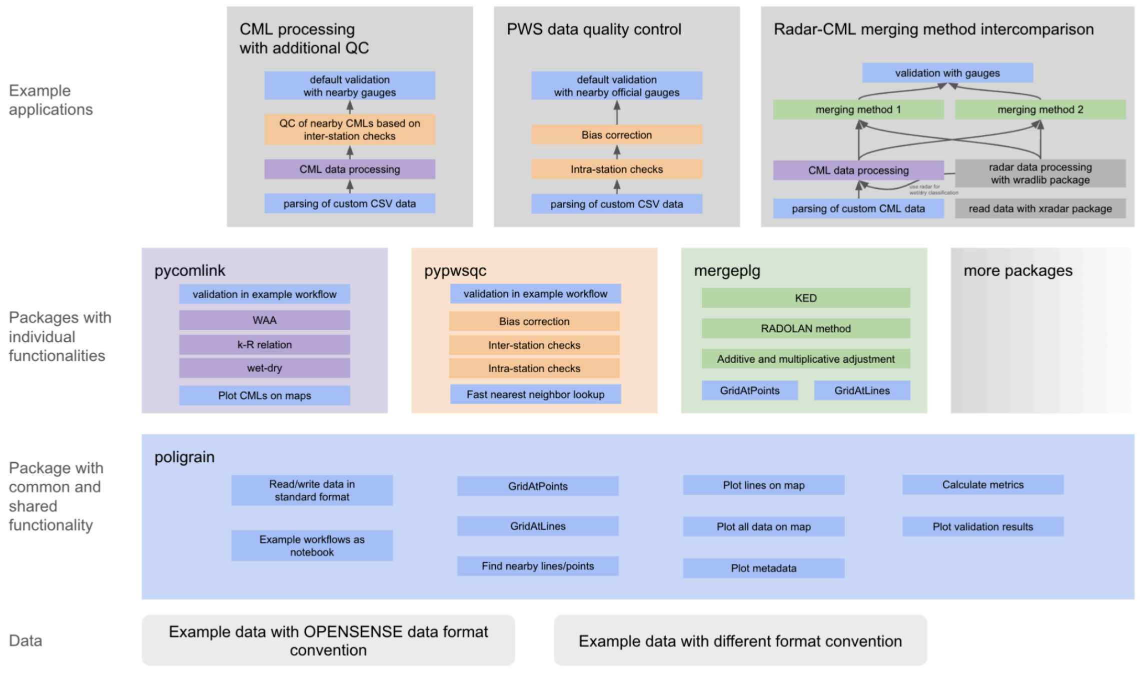

Example applications
CML processing with additional QC
default validation with nearby gauges
QC of nearby CMLs based on inter-station checks
CML data processing
parsing of custom CSV data
PWS data quality control
default validation with nearby official gauges
Bias correction
Intra-station checks
parsing of custom CSV data
Radar-CML merging method intercomparison
validation with gauges
merging method 1
merging method 2
CML data processing
radar data processing with wradlib package
use radar for wet/dry classification
parsing of custom CML data
read data with xradar package
Packages with individual functionalities
pycomlink
validation in example workflow
WAA
k-R relation
wet-dry
Plot CMLs on maps
pypwsqc
validation in example workflow
Bias correction
Inter-station checks
Intra-station checks
Fast nearest neighbor lookup
mergeplg
KED
RADOLAN method
Additive and multiplicative adjustment
GridAtPoints
GridAtLines
more packages
Package with common and shared functionality
poligrain
Read/write data in standard format
Example workflows as notebook
GridAtPoints
GridAtLines
Find nearby lines/points
Plot lines on map
Plot all data on map
Plot metadata
Calculate metrics
Plot validation results
Data
Example data with OPENSENSE data format convention
Example data with different format convention

Figure 1: Overview of the envisioned OpenSense software package ecosystem

3. Summary of current first final state of the OpenSense software packages

Here, we briefly summarise the current list of features of each software package. Development work will continue beyond the duration of OpenSense, hence, there is no final state of these packages, but the current state at the end of OpenSense marks an important milestone where crucial features are available.

It shall be noted here that many larger contributions to the software packages have been done during coding workshops of WG1, WG2 and WG3 and during STSMs.

poligrain

The goal of *poligrain* https://github.com/OpenSenseAction/poligrain is to simplify common tasks for working with point, line and gridded sensor data, focusing on rainfall observations. It is built with best practices of modern software development, including rules and checks of coding style, 100% unit test coverage, automated testing, building of documentation and deployment as Python package via pypi.org and via conda-forge. All contributions are done transparently via pull-request (PR) on Github. Contributions have been provided by six WG2 members. The current list of features includes:

- Seamlessly plot point, line and grid data with a colorscale on a map
- Add map backgrounds from webmap tiles or geographic maps (PR currently in progress)
- Load short example datasets with gauge, CML and radar data
- Calculate distance for point-to-point and line-to-point geometries or find nearest neighbors
- Provide validation plots and metrics
- Fast calculation of intersection of grid and lines to provide e.g. weather radar data averaged along a CML path
- Several example notebooks

pycomlink

Even though pycomlink https://github.com/pycomlink/pycomlink existed before OpenSense, it is considered part of the OpenSense software ecosystem because its data model, using xarray.Datasets for CML and radar data was the blueprint of the WG1 data format standards and thus interoperability with *poligrain* was excellent from the beginning. Due to the convergence towards Python in our community, the “nearby link approach” was ported from the R package RAINLINK to Python during an STSM at TU Delft so that the most widely used CML processing methods were available in *pycomlink.* This made *pycomlink* the central tool for CML processing during other OpenSense activities, like the CML method intercomparison, reported in D2.4 by WG2, or the ongoing comparison of radar-CML merging methods which require standardized CML data processing. The main features of *pycomlink* are:

- Wet-dry classification using several methods (std-dev, SFTF, nearby link, MLP)
- Interface to pytorch and tensorflow to easily run trained models, currently only done for wet-dry classification, but in the future also for direct rain rate estimation (this feature was added during two recent STSMs at KIT)
- Two different k-R relations
- Different WAA estimation methods (time-dependent, different rain-rate dependent ones)
- A detection of data gaps caused by heavy attenuation
- Spatial interpolation using IDW and Kriging
- A reproducible example for a basic CML processing workflow

pypwsqc

With *pypwsqc* https://github.com/OpenSenseAction/pypwsqc we have combined several quality control (QC) and bias correction methods for PWS data which were only available as scripts in R or as unmaintained Python code that did not interface well with the OpenSense dataformat and with *poligrain*. While porting the R implementation, and with a slight variation of the method, a significant speedup of the slowest QC filter was achieved. In addition to porting existing methods, the indicator correlation filter was extended and a new peak removal filter was added. In total five contributors were involved in the development. The same PR-based workflow and the same best-practices as for *poligrain* are also applied for *pypwsqc.* The current list of features include:

- Faulty zero, high influx and station outlier filter from the original R script in *PWSQC*
- Indicator correlation filter from *pws-pyqc*
- Bias correction methods using nearby stations or quantile mapping (PRs for both are currently in progress)
- Reproducible example notebooks

mergeplg

For the work in merging radar with CML data, we needed methods that worked with line data instead of point data, as it is used for radar-gauge adjustment. The relevant code for the fast calculation of grid-line intersection was already in *pycomlink*, but we decided that the merging methods need a separate package. Hence, we created the now Python package *mergeplg* https://github.com/OpenSenseAction/mergeplg to seamlessly use both point and line data for radar adjustment. The grid-intersection code was extracted from *pycomlink* and other existing code, e.g. for the RADOLAN radar-adjustment method as it is done at the German Weather Service (DWD), was added. Based on that, a collection of IDW- and Kriging-based methods have been added. For Kriging a Block-Kriging extension was added to account for the line geometry of CMLs also when interpolating the adjustment field. The method intercomparison for radar-CML merging, carried out in collaboration with WG3, is based on *mergeplg*. The same PR-based workflow and the same best-practices as for *poligrain* are also applied for *mergeplg.* Currently three OpenSense members have contributed to the package. The list of features includes:

- Interpolation function for IDW, Ordinary Kriging and Block Kriging
- Radar adjustment with point and line data based on additive or multiplicative adjustment and using one of the available interpolators
- Kriging with external drift for merging radar, gauge and CML data
- The RADOLAN method for merging radar, gauge and CML data
- Reproducible example notebooks

Example applications

All software packages were used in the OpenSense training school on Merging and Application of OS Rainfall Sensor Data https://github.com/OpenSenseAction/TrainingSchoolMergingApplication, held in Budapest in spring 2025.

The packages *poligrain, pycomlink* and *mergeplg* are also an integral part of the radar-CML merging method intercomparison done in collaboration with WG3 and carried out in a reproducible way on Github https://github.com/OpenSenseAction/radar_adjustment_intercomparison.

Funded by
the European Union

www.cost.eu
www.opensenseaction.eu