OTB integration in operational processing chains

OTB User Days 2021

CS GROUP France – Key elements

480
EMPLOYEES
- Engineers and Experts

40 Years
Experience in Space
- Earth Observation
- Telecom
- Science
- Navigation
- Launchers

49 M€
REVENUES
- France
- Germany
- Netherlands
- Romania
- Canada

CS GROUP France – OTB Integrator and Promotor

- OTB = development framework for image processing over large dataset
  - Flexible: open source, bindings (python)…
  - Completeness: many algorithms available
  - Performance
  - Maintainability: documentation, strong development rules, …
- OTB used for operational development with strong constraints
- OTB training activities for developers or users
- member of the PSC
- Contribution to OTB
- Sponsor of the OTB User Days since 2015

Operational projects with OTB

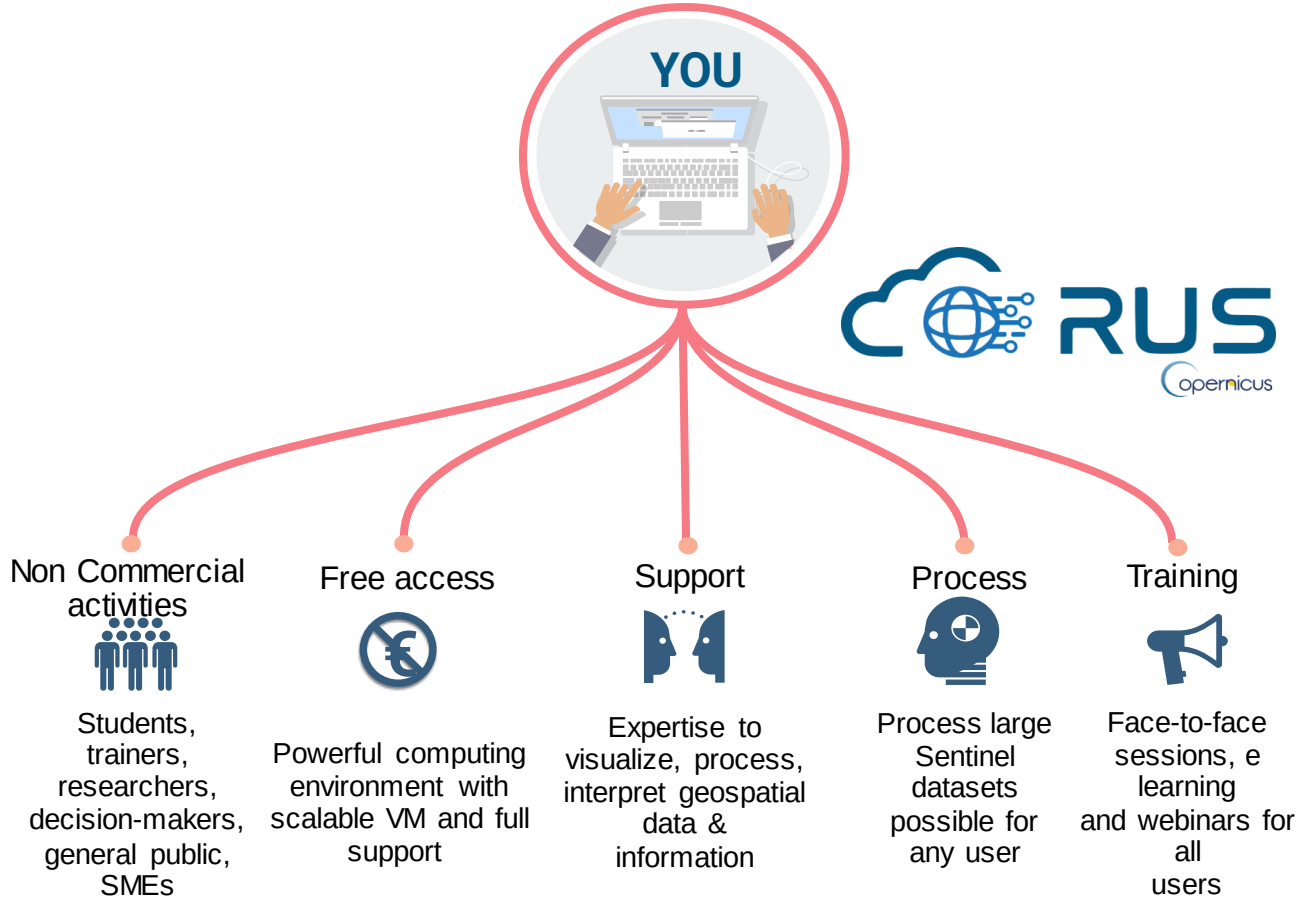
- THEIA/ MUSCATE chains:
  - MAJA L2A Processor
  - WASP
  - LIS
  - BioPhy
  - IOTA²
- Copernicus components:
  - Sentinel-2 PDGS as main component of IPF S2
  - Land – High Resolution Snow and Ice monitoring service
  - RUS
- Sentinel-1 Chains: S1-Tiling, DiapOTB
- Large scale agriculture production Sen2AGRI / SEN4CAP / SEN4STAT / WorldCereal projects
- Kalideos production chain
- SNAP integration since V6, MAJA integrated in SNAP9
- AI4Geo

RUS – bringing expertise to Sentinel data users

More than 2500 users for 3 first years

OTB provide to all users

IOTA² Trainings last year



AI4GEO: AUTOMATIC 3D GEOSPATIAL INFORMATION PRODUCTION

Development of a collaborative Virtual Research Environment optimised for Earth Observation data, AI models and 3D visualization

METIS APPLICATION: Scientific Virtual Research Environment

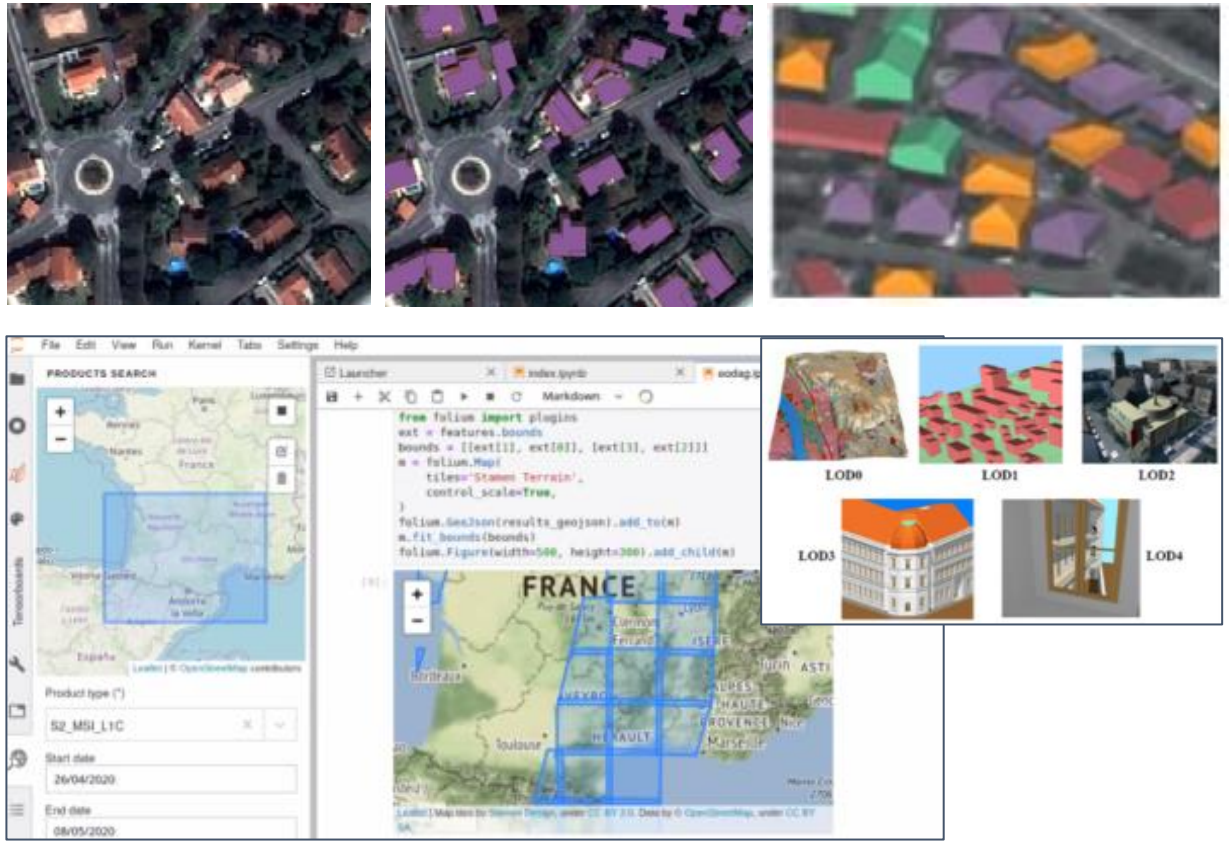
ENVIRONMENT: HPC / CLOUD

METIS BRICKS:

- Big Data processing management (HPC)
- Virtual Research Environment
- Artificial Intelligence tools
- Automatic workflow
- Collaboration tools
- Tools integration: OTB, IOTA²

DATA:

- Satellite and aerial imagery,
- In-situ data
- Lidar 3D points cloud



Open source operational processing chains…

CHAINS:

MAJA

Atmospheric correction and cloud screening software

LIS

Snow cover extent detection algorithm

WASP

Create monthly syntheses of cloud-free reflectances

REMOTE MODULES:

S1Tiling

Ortho-rectification of Sentinel-1 data on Sentinel-2 grid

DiapOTB

SAR Interferometric processing chain

Remote Module

- Easy way to develop new features in the OTB framework
- Template module available*
  - a library (cxx source in src folder)
  - headers and templated classes (include folder)
  - a OTB Application (app folder)
  - tests for C++ sources, applications and python wrappers (test folder)

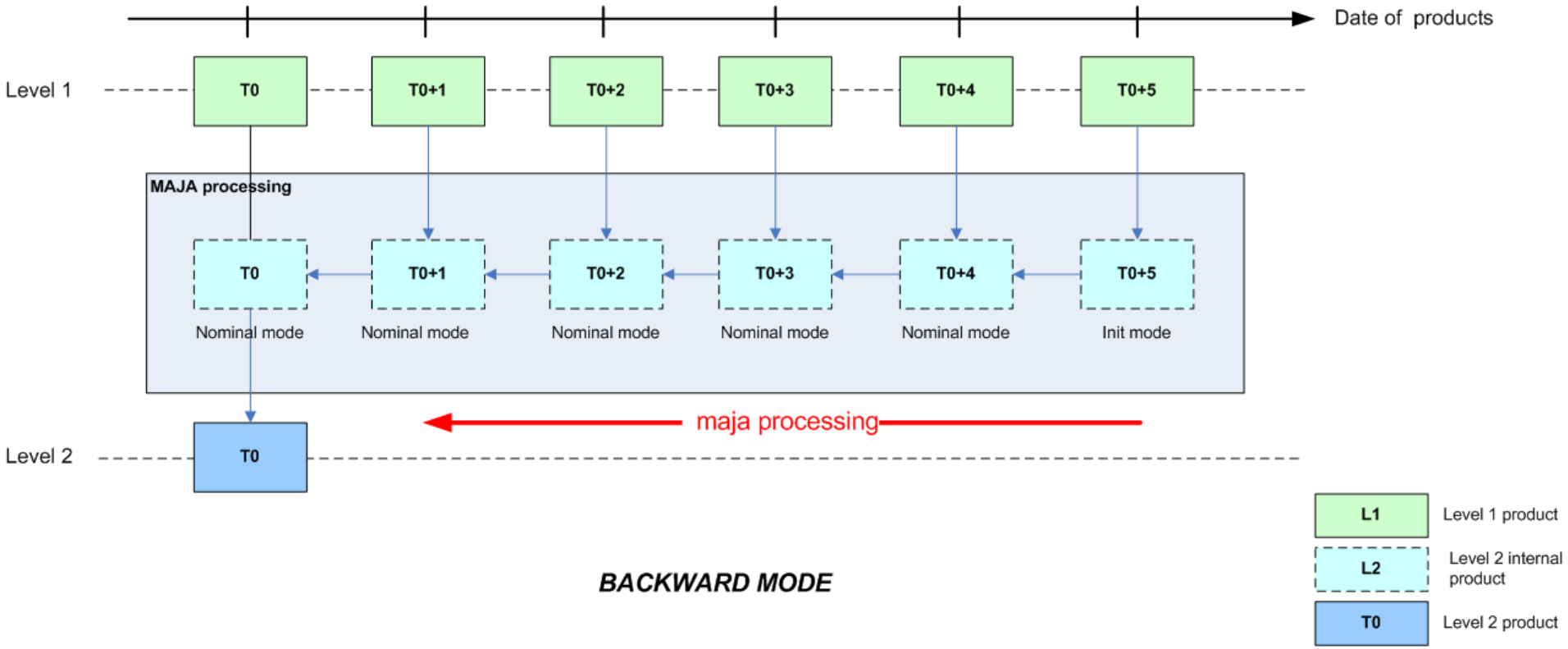
**https://gitlab.orfeo-toolbox.org/remote_modules/remote-module-template*

MAJA, a chain based on OTB

- Atmospheric correction and cloud screening software
- Multi-temporal chain

MAJA 3.3.x	MAJA 4.x
• Main C++ program • Read/Write products in C++ • Algorithms: C++ Filters	• Open source • Orchestration written in python • Read/Write products in python • Framework of OTB Applications • StartMaja included within maja

MAJA, multitemporal chain processing



OTB App Handler in MAJA

- OtbAppHandler interface: OTB application instance

```
from orchestrator.cots.otb.otb_app_handler import OtbAppHandler

# Launch the app
param_reflectance = {"tocr": dict_of_output.get("RayleighIPTOCR"),
                     "edg":  dict_of_input.get("L1Reader").get_value("IPEDGSubOutput"),
                     "sat": dict_of_input.get("L1Reader").get_value("IPSATSubOutput"),
                     "waterthreshold": water_treshold,
                      "bluebandtocr": bluebandtocr_idx,
                      "redbandtocr": redbandtocr_idx,
                      "nirbandtocr": nirbandtocr_idx,
                      "correlbandtocr": correlbandtocr_idx,
                     "bluereflectancethresholdvalue": dict_of_input.get("L2COMM").get_value_f(
"CloudBlueReflectanceThreshold"),
                     "redreflectancethresholdvalue": dict_of_input.get("L2COMM").get_value_f(
"CloudRedReflectanceThreshold"),
                     }

reflectance_app = OtbAppHandler("CloudReflectance", param_reflectance, write_output=true)
```

OTB pipeline manager in MAJA

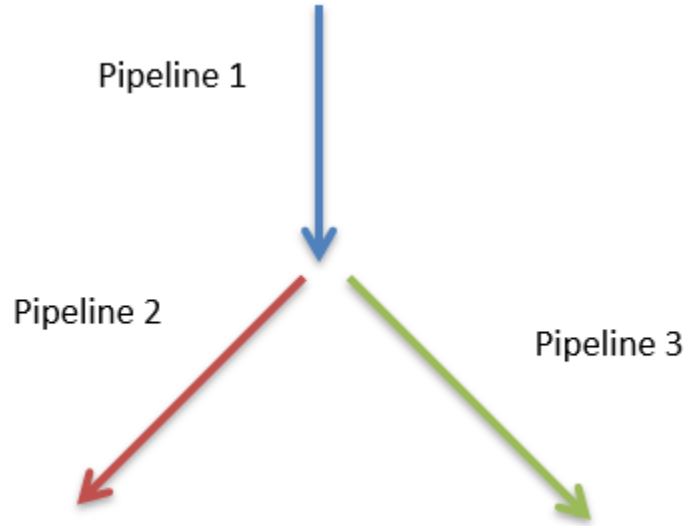
- OtbPipelineManager:
  - Add_otb_app(app) : add pipeline
  - Free_otb_app : free pipeline
  - Get_last_app : get the last application of the pipeline
  - Automatically free up the pipeline after being removed

```
from orchestrator.cots.otb.otb_app_handler import OtbAppHandler
from orchestrator.cots.otb.otb_pipeline_manager import OtbPipelineManager

a_pipeline = OtbPipelineManager()

app1 = OtbAppHandler(« BandMath », {« in » : tmp.tif , « out » : out1.tiff, « exp » : im1b1 },
Write_output =false)
a_pipeline.add_otb_app(app1)

b_pipeline = OtbPipelineManager()
app2 = OtbAppHandler(« BandMath », {« in » : app1.getOutput()[« ou »] , « out » : out2.tiff, « exp » : im1b1
},
Write_output =false)
b_pipeline.add_otb_app(app2) // Pipeline b depends on pipeline a cannot erase pipeline a
```



Conclusion

- OTB is currently a component for operational chains
- How to improve OTB integration in chains ?
- Reuse the different experiences:
  - Dask ?
  - Python integration at library level?
- Share a common processing framework?
- Share best practices with other projects!