felyx version 2 – the cat is back!

new release of the distributed and cloud/HPC-ready multi-matchup dataset production framework

JF Piollé, E Bodéré, C André (Ifremer)
I Tomazic (EUMETSAT)

felyx

work funded by Copernicus through EUMETSAT

Context

felyx is a **generic open-source** tool for **extracting** Earth Observation data over **static** or **moving** locations, in particular for the production of Matchup Databases

generic means here it is agnostic wrt the type of variables, the source of data, the observation domain,...

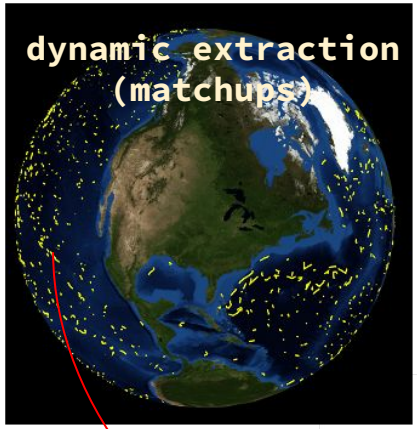
Initially developed under ESA funding

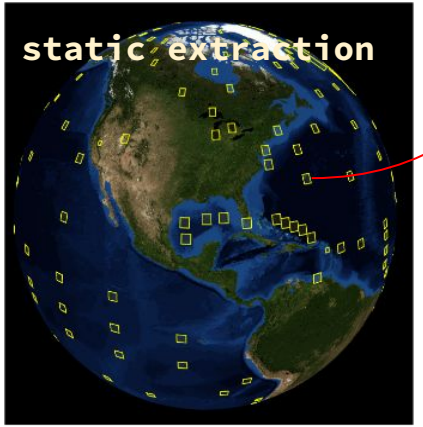
Has been around for some years, suffered some **flaws** and **missing functionalities**

new requirements defined by EUMETSAT based on previous experience, new version **funded by Copernicus through EUMETSAT** (https://www.eumetsat.int/Sci4MaST)

implementation by an **Ifremer** team (cooperation LOPS/CERSAT with Marine Data & Information Systems Department) over 2021-2022

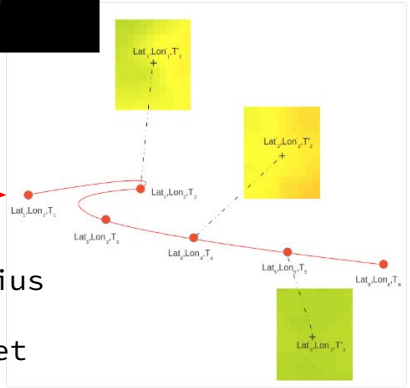
felyx extraction principle





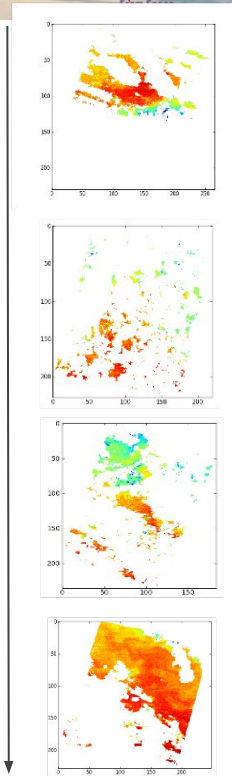
trajectories in parquet format or Elasticsearch

configurable subset size, colocation radius and maximum time difference per dataset



extracted subsets (**child products** from main source files) can be saved to disk or just indexed (and collected/assembled later)

metrics can be computed for child product (configurable)



Multi matchup assembly

The extracted matchups (previous step) are then assembled into NetCDF files containing multiple matchups

jointure with in situ data - configurable in situ history can be provided for each matchup

Flexibility in MMDB output format through YAML configuration file:

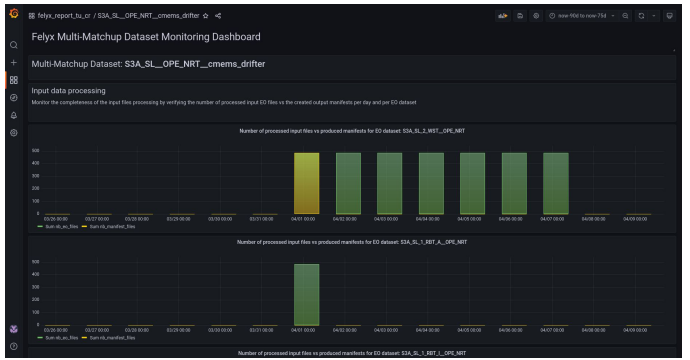
- configurable periodicity (hours, days, etc…)
- combining different datasets (or processing level: SLSTR L1, L2, L2P)
- keeping only relevant variables and attributes from each source product (right side)
- dividing into subproducts => different files (core MMDB, expert MMDB, ancillary fields, ...)

An end to end command allows to process all steps from input SST files to MMDB output in one go

```yaml
# The output products that will be written to disk, where keys are the
# identifier of each product and the values their definition.
# In most case, there would be only one output product with all the
# selected dataset variables and attributes. However, one can define
# multiple products, each one having a particular selection of EO
# datasets, variables and attributes.
products:
  SLSTRA-MAR-L2P-v1.0_test4dyn:
    # file pattern for the output product
    filenaming: '%Y/%Y%m%d%H%M%S_SLSTRA-MAR-L2P-v1.0__test4syn.nc'
    # tailor the content of the assembled files for the output product
    content:
      SLSTRA-MAR-L2P-v1.0:
        # [Optional] list of variables to include in the assembled files
        # (all of them by default). Python regexp can be used to select
        # several variables at once.
        variables: [.*]

        # [Optional] variables NOT TO include in extracted child products
        # (none of them by default)
        #except_variables:

        # [Optional] list of global attributes to include in the assembled
        # files (all of them by default)
        #attributes:

        # [Optional] list of global attributes NOT TO include in the assembled
        # files (none of them by default)
        except_attributes: [.*]

        # [Optional] list of global attributes from the child products
        # to stack as new variables into the assembled files.
        attributes_as_variables:
          - date_created

        # prefix by which to rename all variables and global attributes
        # coming from this dataset (by default the dataset id is used)
        prefix: s3a
```

Main improvements

- **lighter system :**
  - **reduced dependencies** on third party tools
  - **configuration** is entirely file based (YAML), no more web interface and front-end
  - **storage of in situ data** is based on Apache/parquet format (Elasticsearch storage is still possible)
  - for MMDB, no need to store extracted intermediate child products (replaced within **indexing**)
  - **easier installation**: pypi repo, docker images, soon conda
  - can run in local env in sequential mode with **minimal installation**
- complementary **distributed processing framework** (jobard)
- complementary package for **graphical reporting** and **alerting** (felyx-report)



Other new functionalities

Multi-matchup data files:

- sister datasets : extract/combine simultaneously from L1/L2/L2P without searching twice or more for matchups (ex: for SLSTR)
- traceability to source measurement (both EO and in situ data)
  - file name and index within file of matched data
  - transformation of attributes to traceability variables (version, creation date, UUID,...)

Input data

in situ data

- need to be converted to periodic **Apache/parquet** files
- parquet = compact column based format for big data
- **id, time, lat, lon, (z)**, any param

id	time	z	lat	lon	depth	water_temperature	quality_level	...
2903425	2021-09-15 00:03:06	0	28.231291	153.017426	0.715198	29.293001	5	
		1	28.231291	153.017426	0.794664	29.284002	5	
		2	28.231291	153.017426	1.033063	29.275002	5	
		3	28.231291	153.017426	1.191995	29.278002	5	
		4	28.231291	153.017426	1.231728	29.258001	5	
...			...	...	...	...	...	...
6902756	2021-09-15 23:57:00	0	49.208000	-47.723999	0.000000	11.371000	2	
		1	49.208000	-47.723999	0.991496	11.374001	2	
		2	49.208000	-47.723999	1.982988	11.376000	5	
		3	49.208000	-47.723999	2.974475	11.383000	5	
		4	49.208000	-47.723999	3.965957	11.383000	5	

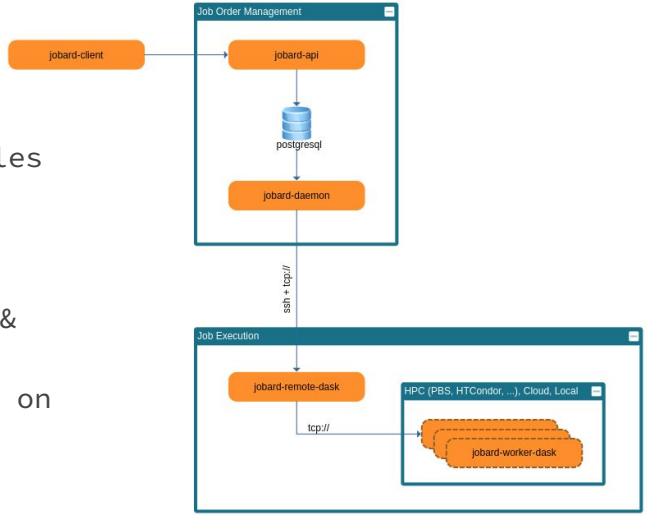
[1980 rows x 5 columns]

satellite data

- data read through **cerbere** generic reading lib (based itself on xarray)
  - https://cerbere.gitlab-pages.ifremer.fr/cerbere/
  - https://gitlab.ifremer.fr/cerbere
- should work straight away with CF compliant datasets
- can be extended through contribs for other formats or conventions (many existing already)
- GHRSST plugin natively available (account for sst_dtime non conformity), plugins for SLSTR L1/L2

Distributed processing with **jobard**

- **jobard** is framework developed for **job-array distributed processing**: running independent processings simultaneously (embarrassingly parallel)
  - e.g. running matchup extraction from multiple GHRSST files in parallel
- **Jobard** come as an independent **python** package based on Dask - usable for many reprocessing tasks
- currently works over **Docker SWARM** (cloud environment) or **PBS** & **HTCondor** (HPC environment), planned **kubernetis** support
- in a cloud environment it will deploy and instantiate workers on multiple VMs
- can process thousands of entries put in a queue
- progress can be monitored, access to processing context (logs,...)
- docs: https://jobard.gitlab-pages.ifremer.fr/documentation
- gitlab repo: https://gitlab.ifremer.fr/jobard
- **public release**: September 2022



Implementation

full **python** implementation

relies on Ifremer **cerbere** lib for generic access to data (itself built upon **xarray**)

emphasis on robustness and operations:

- unitary testing with **PyTest** framework
- continuous integration and deployment (**gitlab**)
- code quality checker: **flake8**, **pylint**
- packaging and dependencies with **poetry**
- trained maintenance team and support

Installation

felyx and complementary packages can be installed:

- in conda from source or with pip
  - repo will be moved to pypi
- through docker
- next:
  - conda package
  - deployment through singularity on HPC

Continuous Integration (CI) / Continuous Deployment (CD) workflows for various environments have been set-up with gitlab & Ansible for deployment to operational environments, automatic updates or deployment to new targets

installation tests to external clouds (WEkEO, AWS) to be be done

conda, from GIT repo

```
conda create -n felyx_processor_from_git -y --file
https://gitlab.ifremer.fr/felyx/felyx_processor/-/raw/master/assets/conda/felyx-dev-li
nux-64.lock
conda activate felyx_processor
pip install --upgrade --force-reinstall
git+https://gitlab.ifremer.fr/felyx/felyx_processor[plugins_metrics_base]
```

conda, with pip repo

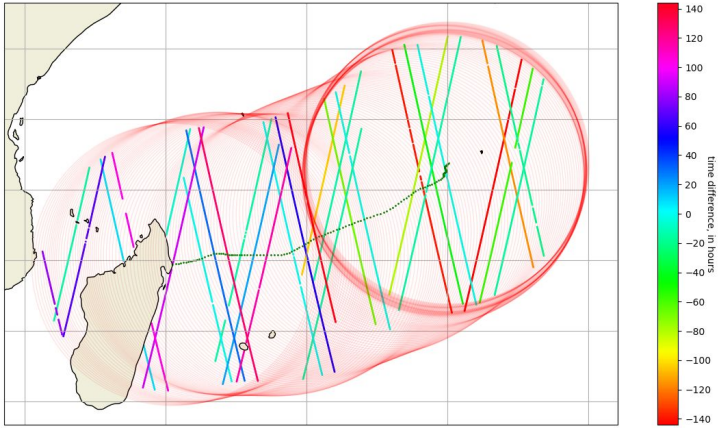
```
conda create -n felyx_processor -y --file
https://gitlab.ifremer.fr/felyx/felyx_processor/-/raw/master/assets/conda/felyx-dev-li
nux-64.lock
conda activate felyx_processor
pip install --upgrade --force-reinstall \
  --extra-index-url
https://gitlab.ifremer.fr/api/v4/projects/1225/packages/pypi/simple \
  felyx_processor[plugins_metrics_base]
```

docker

```
docker run /
  gitlab-registry.ifremer.fr/felyx/felyx_processor:2.1.0 \
  felyx-extraction \
        -c /home/felyx/conf/mmdb/test/s3a_mmdb.yaml \
        --dataset_id S3A_SL_2_WST__OPE_NRT \
        --manifest_dir /home/felyx/data/manifests/
```

Applications

- EUMETSAT Multi-Sensor Matchup Databases for Sentinel-3 A & B/SLSTR, METOP/AVHRR & IASI, NPP/VIIRS
  - from CMEMS Insitu TAC for drifters/moored buoys and Argo
  - TRUSTED buoys
  - Ship4SST radiometer data
  - coming: saildrone data
- EUMETSAT Ice Temperature MDB (coming)
- EUMETSAT Sentinel-3 SRAL validation of wind & wave
- ESA CCI Sea State: colocation of altimeter data with wave buoys
- ESA MAXSS project (https://maxss.org) : Atlas of observations over tropical, extra-tropical and polar lows
- ESA OceanSoda carbonate database (https://doi.org/10.12770/0dc16d62-05f6-4bbe-9dc4-6d47825a5931)
- SWOT mission preparation (Ifremer)
- Future applications
  - validation of very high resolution SST (Landsat, TRISHNA,...)
  - CDAF intercomparison framework
  - MDB intercomparison framework



extraction of altimeter tracks along hurricane path (ABELA)

Conclusion and perspective

- https://felyx.ifremer.fr : documentation, installation, configuration, usage with jobard distributed framework, etc…
- public release of felyx v2 planned in Sept 2022
- next steps:
  - conda packages
  - some more optimizations
  - support for kubernetis
  - test on external cloud platforms w/ object storage hosting GHRSST datasets: WEkEO, PO.DAAC/AWS
  - demonstrate the ability to produce in a consistent manner multiple GHRSST MMDBs close to data location for fair intercomparison
- Any question: jfpiolle@ifremer.fr