Status of reconstruction code

E. Di Marco

Analysis meeting, Coimbra, 7 June 2023

Reconstruction workflow

- Offline reconstruction workflow software is located, together with some other analysis utilities, in this GitHub repository
- What do I mean as reconstruction?
  1. Get an event in a given format with the RAW data (images + PMT waveform) from either the output of the DAQ (data runs) or SIM
  2. Unpack the RAW
  3. Do basic low level stuff (noise suppression, single pixel amplitude corrections)
  4. Run a clustering algorithm to get tracks
  5. Compute cluster variables which need the full pixel information
  6. Store the interesting clusters (with as loose as achievable selection) and variables as simple, plain, ROOT trees
     - Not storing any object, just plain floats and arrays of floats, so that any student can simply use them to make a data/SIM analysis
  - Same code has to run on both data and SIM, with minimal input changes (different input format)
- A RAW EVENT is image + PMTs => ideally in the same job the reco of the image and the PMTs can be run
  - And in practice, this was working in the past with LEMON (1 PMT) with basic stuff
  - Will not talk of PMT reco here, see D. Marques talk tomorrow !

The postProcessing workflow

- This is a second job that runs on the output of the full RECO step. Code framework here.
- Meant to do higher level computations, that can:
  - use this reduced information as an input
  - Need to be redone multiple times once the reconstructed clusters are done
- Working example: energy and z-regressions
  - The training can be done multiple times (new optimisation, new dedicated data taken, one training for sim, etc.)
- Other possible usages:
  - A stable energy calibration is derived, which is not just a constant $N_{\mathrm{pho}} \rightarrow \mathrm{keV}$ constant , and want to store this for all to use
  - High level identification variables (eg. The output of dedicated ER / NR Machine Learning BDTs)
- I.e. compute variables that are fast to calculate and make them persistent in ROOT files that can be joined to the main trees for the other to use w/o need to recompute on the fly and risk to make mistakes
  - i.e. many "friend trees" cycles / 1 cycle of "main trees"
- What does "fast" mean:
  - RECO runs at $\mathcal{O}(1\,\mathrm{Hz})$
  - Post-processing runs at $\mathcal{O}(1\,\mathrm{kHz})$

The CYGNO reco workflow

Source, Cosmic Rays, Dark Matter wind…

RAW
TH2I - Picture
+ (n x TGGraph) -
PMTs waveforms

Event

Zero suppression
+ noise filters
+ optical corrections

RAW

Hits (numPy arrays)

Event

Reconstruction: clustering + cluster shapes + PMT peak finder

RAW

Clusters Peaks

Hits (numPy arrays)

Event

Dataset: Runs (100 - 1000 events each)

Event Collection

Event **Event** **Event** …

Post-Processing (calibrations, high-level analysis)

RAW

RECO

(1 TTree with clusters + properties + reduced set of "interesting hits"

1 TTree with PMT peaks +

FRIENDS

(1 "friend" TTree with calibrated cluster energies)

What changed since ORANGE @ BTF?

- The framework is about the same since 2016. Developed for ORANGE at BTF, but especially for LEMON
  - It's all python code, but using where possible numpy to make it faster
  - The main part which has been changed multiple times is the clustering
  - The code is modular in the sense that changing the CORE clustering algorithm does not need to change the rest of the framework (input, noise suppression, variable calculations)
  - This has PROs and CON's
    - PRO's are obvious
    - CON's: some parts of the code never touched (improved, fixed, checked) for > 5 yrs
- In the meantime, the community of the users increased a lot
  - And also increased the requests of customisations:
    - Simulation of the current mainstream detector (now LIME)
    - Different detectors for parallel studies (eg. MANGO)
    - Save different output format for more demanding analyses (eg. Directionality)

What did we use (overground)

ORANGE:

10 x 10cm²

1 cm sensitive gap

NNC, DBSCAN

LEMON:

500 cm²

20 cm sensitive gap

Geodesic Active Contours (GAC)

to reconstruct both long and short tracks

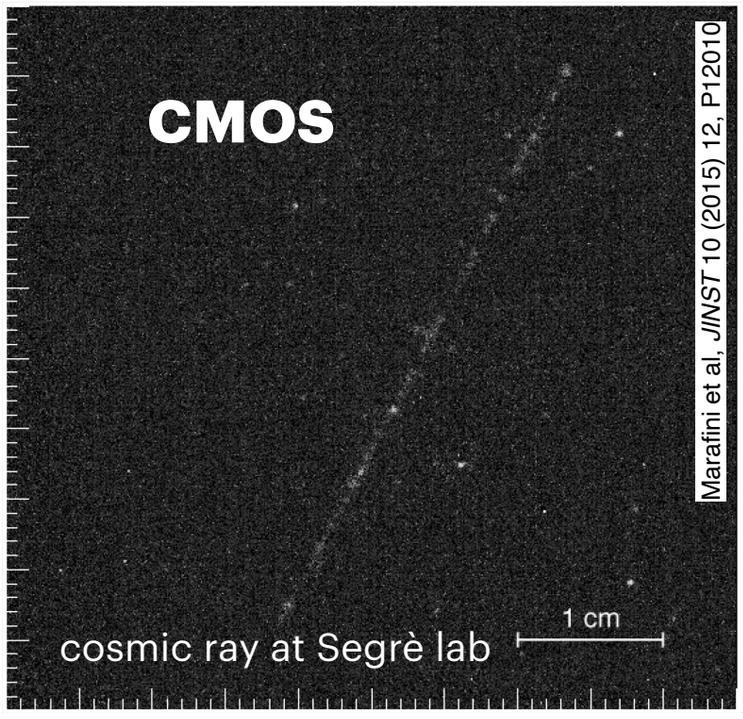
LIME:

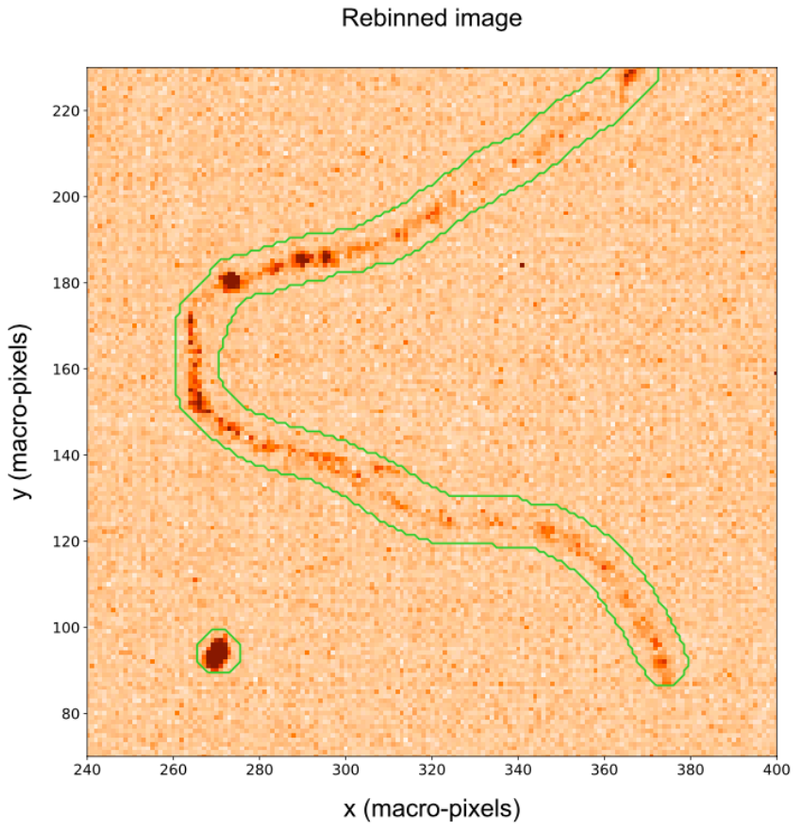
1000 cm²

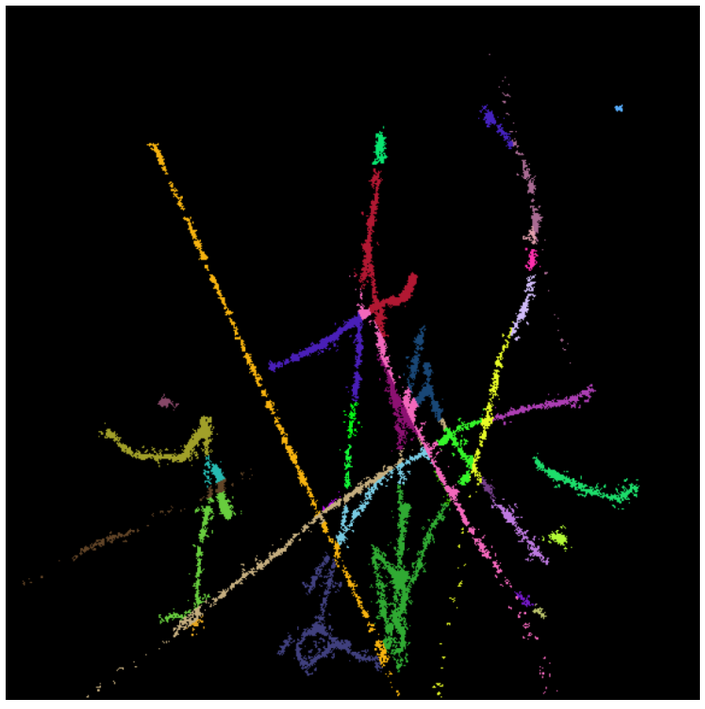
50 cm sensitive gap

directional DBSCAN

for the long and overlapping tracks and **DBSCAN** for the remaining



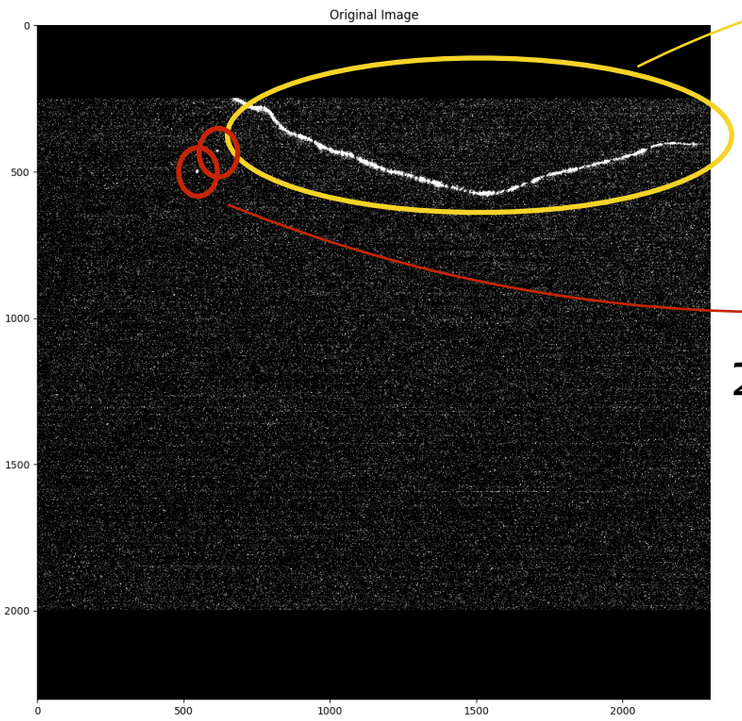




...and underground ?

- We are using still directional DBSCAN + DBSCAN for the remaining because we still have some long tracks
  - Really needed? No, because the rate of overlaps is very small
  - So why? It doesn't harm, and there is no real sign of inefficiency

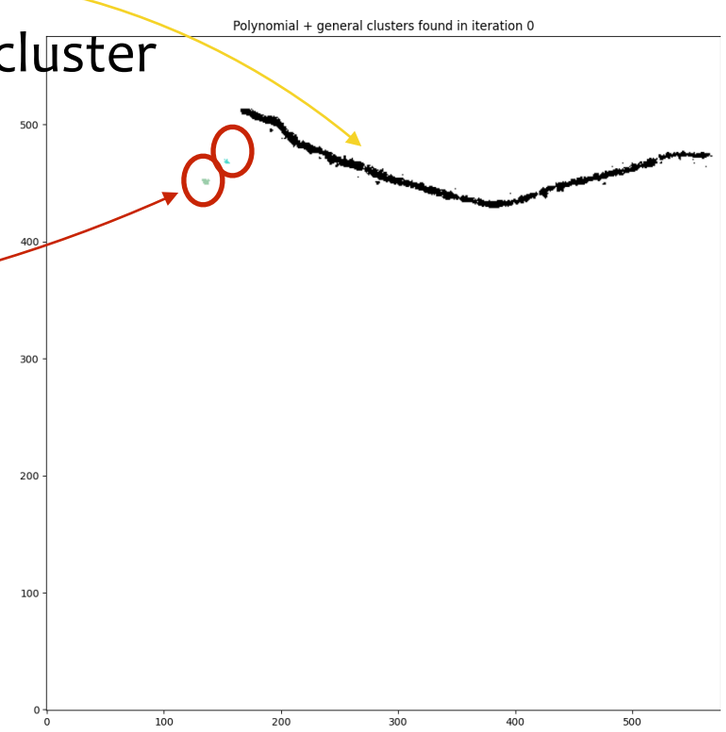
LIME Run-2:



Only 1 directional cluster

2 DB scan clusters

LIME Run-2 output of RECO



- There no real need of this, especially in view of Run-3 (more Cu shielding) and even less for Run>3. Would be good to study it
  - e.g. LEMON-like reconstruction (GAC) could be more efficient, and speed is not a problem
  - Others always cite ML methods (CNNs, etc) as the holy grail. Run3 can be the quiet setup to try

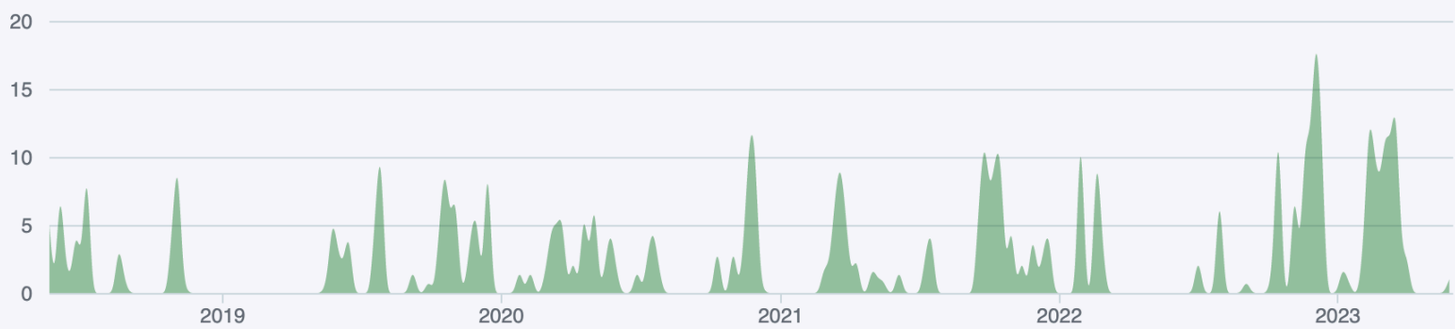
The developers in the last 5 yrs

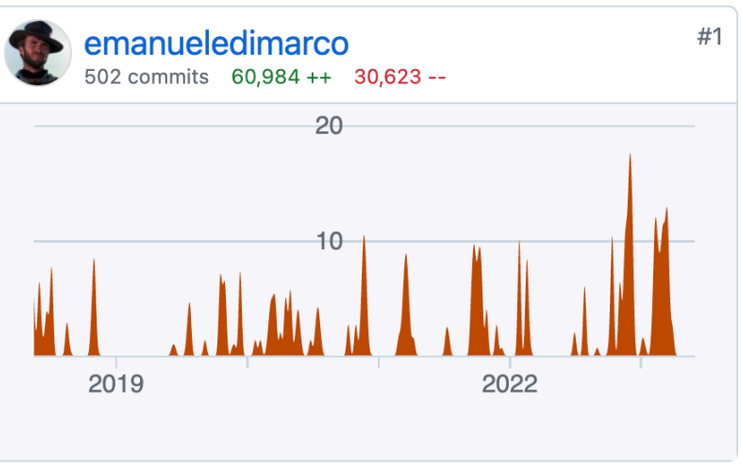
May 20, 2018 – Jun 2, 2023

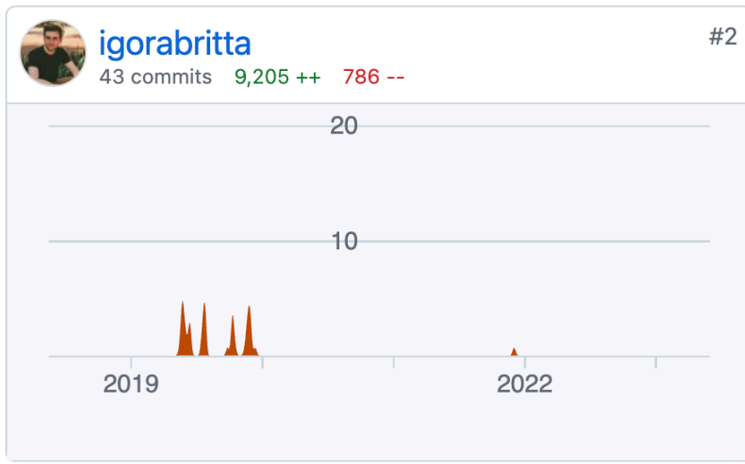
Contributions: Commits

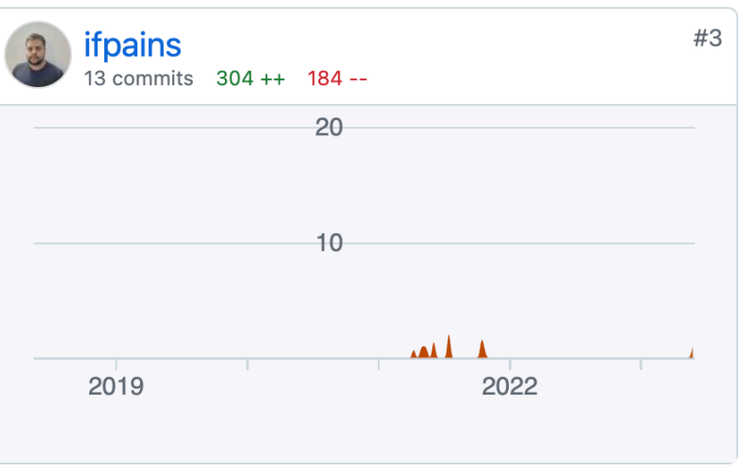
Contributions to winter23, excluding merge commits and bot accounts

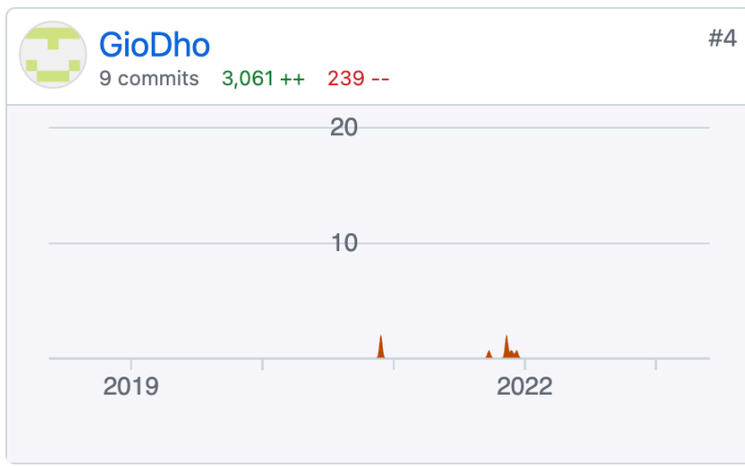
From our GitHub repository











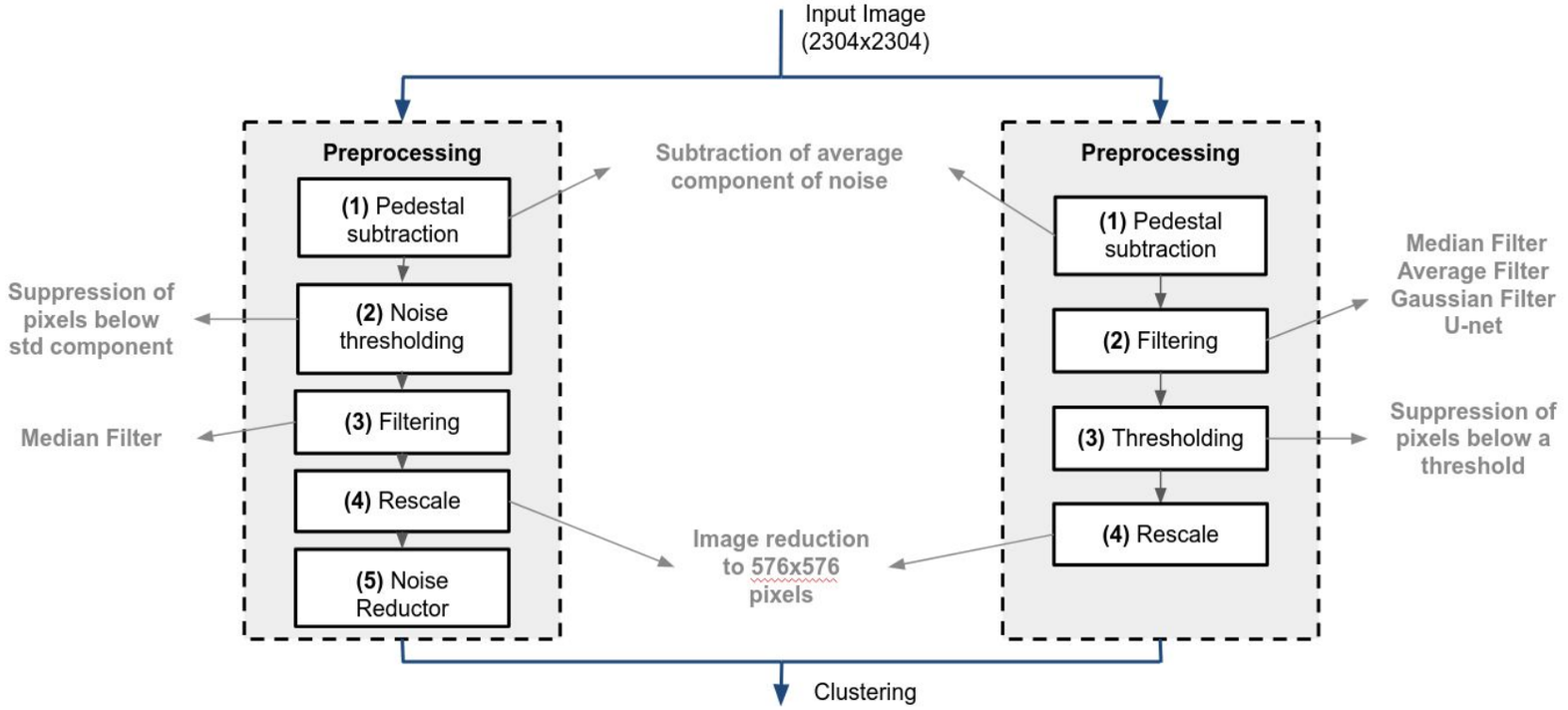
- We need more people that implement stuff here
  - More eyes find bugs earlier
  - It's more efficient and more intellectually stimulating to implement new ideas directly in the code and test it wrt ask for new features
- There have been studies, all done standalone, but a little effort to integrate them
  - No commits, no party !
  - i.e. no analysis - publications

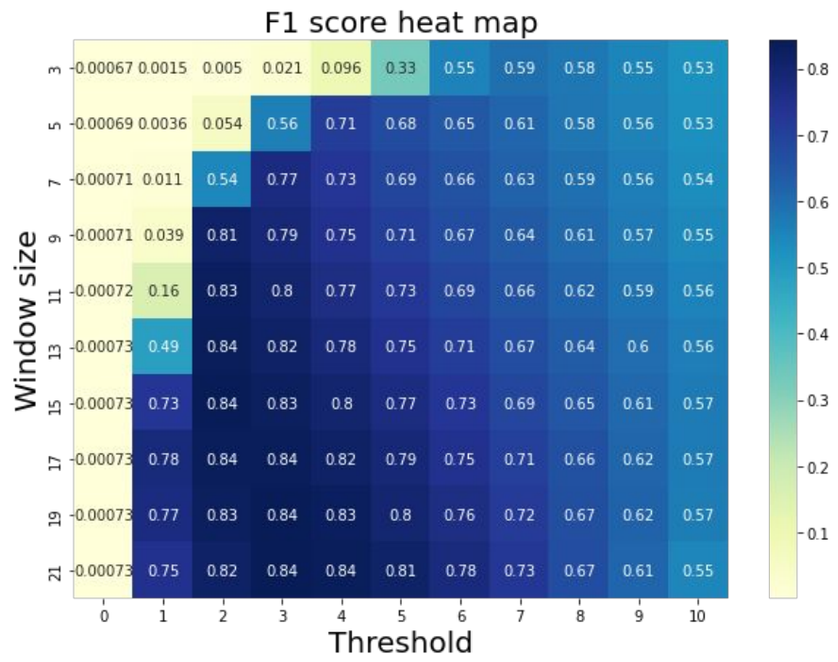
Current issues: simulation

- Simulation issue (link):
  - The real data format moved from ROOT files (with TH2Is + TGraphs for PMTs) to Midas files
  - Why? To integrate more the output of the slow-control
  - This was NOT w/o cost:
    - It introduced a dependency on cygno-lib (to unpack Midas), which broke the compatibility with root_numpy (library for fast conversion of ROOT => numpy arrays, input of RECO)
    - root_numpy not developed any more, because there is a fancier library: uproot
    - I made the migration (also needed to support the reconstruction of Run1 files, LEMON, and all data not taken with CYGNO DAQ) to uproot
    - It works, but has a huge memory leak, isolated only when getting 1 TH2I and streaming the serialised numpy array (probably it harms us because we spit a 5.3×10^6 - long array of floats)
    - SO cannot run SIM efficiently (it breaks in multicore, can be slowly run in single-core on few chunks of events)
  - Solution-1 (preferred): report the issue to uproot developers. Doing it…
  - Solution-2: save the output of SIM in (zipped) py arrays => no need of conversion
    - This seems not viable since output is x 10 larger that ROOT (which uses efficient LZMA compression)

Current developments: noise

- The noise filtering is a combination of zero-suppression (with hand-tuned threshold) + a couple of filters
  - A matrix filter to remove pixels with an amplitude too different wrt the regional average
  - A median filter
- But this block can be changed with more sophisticated stuff
  - e.g. the convolution method that I. Pains developed for another scope (trigger images with "some activity") can be used as noise filter
  - Less radical: use the optimisation of the median filter? Move to Average filter? To U-Net?





Time has come to integrate them and test it for real on data?

Wish developments

- Save sub clusters to make an efficient image - PMT matching in space
  - Can be used to steer the PMT reco, using the x-y precise position from image (see A. Messina's talk in this meeting !)
  - Can be used to apply offline the energy and z-regressions to spot-like clusters compatible with the training phase space of ^{55}Fe in the post-processing step, also for long tracks, in the post-processing step, and no need for a full re-reco
- Optimisation of cluster efficiency at very low energy
  - Go back to GAC (LEMON-like) or different, fancier clusterings
- Calculate more sophisticated variables for ER / NR discrimination
  - For sure these profit from having the full pixels granularity info lost after reconstruction
- Rewrite everything from scratch in C++ if you want to reach the best speed

- For post-processing:
  - Write the code that reads the slow-control detector conditions (Pressure, Temperature, etc.) and makes corrections, for example first principles gain corrections to the LY as a function of P/T

Automation & DQM

- There has been quite some work on “middleware”, i.e. a fast reconstruction for Data Quality Monitoring
  - Ideally, one wants to use the offline reconstruction as it is to make also DQM
  - If it is fast enough. How coarsely in time do we want the response?
    - My answer is no more than 1 / run (no need of event-by-event streaming), because we need some stat to make meaningful plots (eg. Cluster occupancy, cluster integral distributions)
  - I think we are there, i.e. the offline can be used now (see I. Abritta’s study here on code timing and improvements of guilty slow function)
- Plus, we have the automation framework, with Grafana monitoring, that for Run3 runs it w/o the need of my submission in the LNGS queues

See I. A. Costa’s talk in this meeting!

Resources (storage)

- In this table in the RECO wiki there's the documentation of the RECO trees
- In this table there is the full breakdown of the event size in the RECO trees

Event data

collection	kind	vars	items/evt	kb/evt	b/item	plot	%	ascending cumulative %
redpix	collection	4	92524.00	225.165	2.5		86.0%	86.0%
sc	collection	38	335.00	36.278	110.9		13.9%	99.8%
cmos	singleton	3	1.00	0.243	249.0		0.1%	99.9%
pedestal	variable	1	1.00	0.083	85.0		0.0%	99.9%
event	variable	1	1.00	0.076	78.0		0.0%	100.0%
run	variable	1	1.00	0.074	76.0		0.0%	100.0%
All Events data				261.92			99.0%[a]	
Non per-event data or overhead				2.76			1.0%[a]	
File size				264.68				

Reduced pixels collection:
x-y-z of all the pixels passing ZS and belonging to selected (configurable) clusters.

Takes 86% of the size. Not saved by default (Winter23)

SC: all super-cluster variables

36 kb/event => RECO files are very small, and are stored in CLOUD

Resources (CPUs)

- So far (up to Run-2) all the reconstruction has been run smoothly on LNGS batch system
  - Standard "cygno" queue was sufficient (no need more than 1 GB/ event)
  - Cygno-custom was asked to run special workflows

	Cygno	Cygno-custom
Max-jobs-queuable	800	100
Max-jobs-runnable	500	10
Max-CPU/job	8	64
Max-CPUs-runnable	4000	640
Max-RAM/job	9 GB	110 GB
default-RAM/job	1 GB	1 GB

- Now the automation runs on condor, using the cloud resources. Let's see, we can use LNGS to run simulation, or offline analysis

Re-reco

- We will need a new full cycle of reconstruction of the same data ("re-reco") when we update the code because of:
  - fixes (e.g. Winter23 subtracted rotate pedestals, fixed by I. Pains),
  - new developments (eg. New noise filtering, new fancy clustering, NN etc)
  - Save the current energy- and z-regression for "sub-clusters" which are comparable in size with the training ^{55}Fe spot-like
  - different output format
    - save reduced pixels for directionality (implemented, just turn-on in config)
    - save subclusters (need developments)
  - Improved low-level calibrations (change pedestals, change optical corrections)
- I guess this might happen 2/3 times per data-taking era

Conclusions

- We have a baseline reconstruction code and a workflow for post-reconstructions calibrations and corrections
- It implements one of the possible schemes, and now we have an automatic way of running it when we take data
- Still, what it does is very basic, while the needs and the scope of the experiment have grown: time to revisit what is there, and push a new R & D?
- The best is that the people who need new stuff for analysis put their hands on the code and develop stuff
- Be focused on LIME next runs, but also start thinking at how this will scale to CYGNO04 and beyond

The End