# SiWECALsoft: First updates post TB

Jesús P. Márquez, Olmo Arquero

# Current state of our software

- TB analysis: https://github.com/Olmichu22/siwecal-tb2026/tree/main
  - Data conversion
  - Mip & pedestal calibration
  - Event building
  - Validation variables (shower study)
  - Event_display (key4hep native)
  - Event_viewer (Python, real-time tools) - Olmo
- Simulation: https://github.com/marherje/siwecal_k4sim
  - (WIP) Realistic digitization - Yukun
  - Mip Calibration
  - Validation variables (shower study)
  - Event_display (key4hep native)
  - Event_viewer (Python, multiple tools!)

Changes and commits until yesterday, update your local copy ⚠

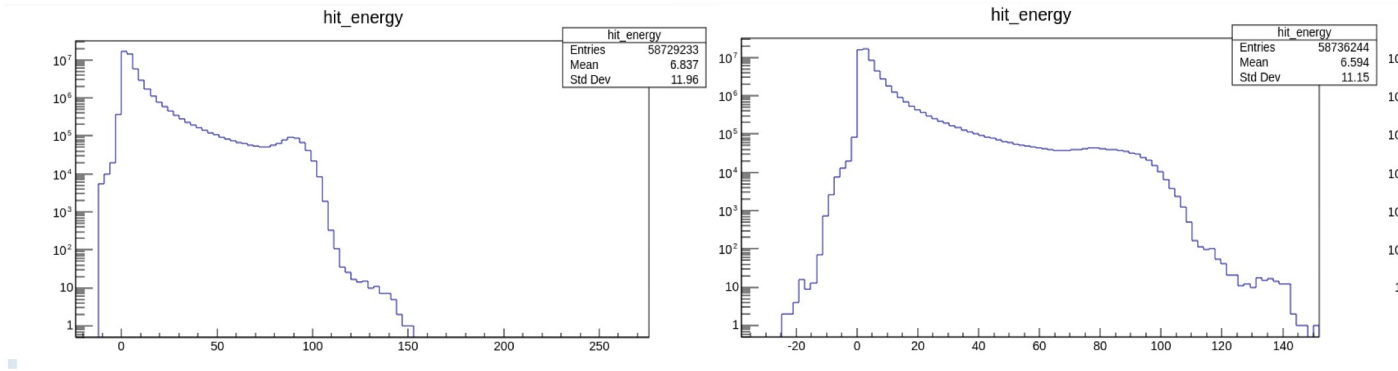
# Conversion and calibration

- The conversion of files is ported from old repositories
  - Inherits all the methods from previous software
    - From binary to root tree, then from root tree to hits
    - The fix of that bug that Taikan spotted was added last night ⚠
  - I also added acq. windows, delays between windows, and global threshold in the tree
    - To use it directly instead of checking by eye or with external files (Run_settings)

- New Gaudi pipeline to do mips and pedestal calibrations
  - Auxiliary tools to scan run thresholds and provided a summary of runs available
  - Fallbacks added in both calibrations to avoid artifacts (-nan, extreme values, etc.)
  - Some calibrations fail just from low statistics, an average value is set instead of masking
    - Only if the entries are exactly 0, then the channel is considered masked
  - I finished th230, th220 is running but it got stuck a few times (Much more statistic)

# Pedestal and MIP calibration

- Dummy calibration (left) vs first calibration th220 (center) vs new calibration th230
  - This example is from run13, in had th230



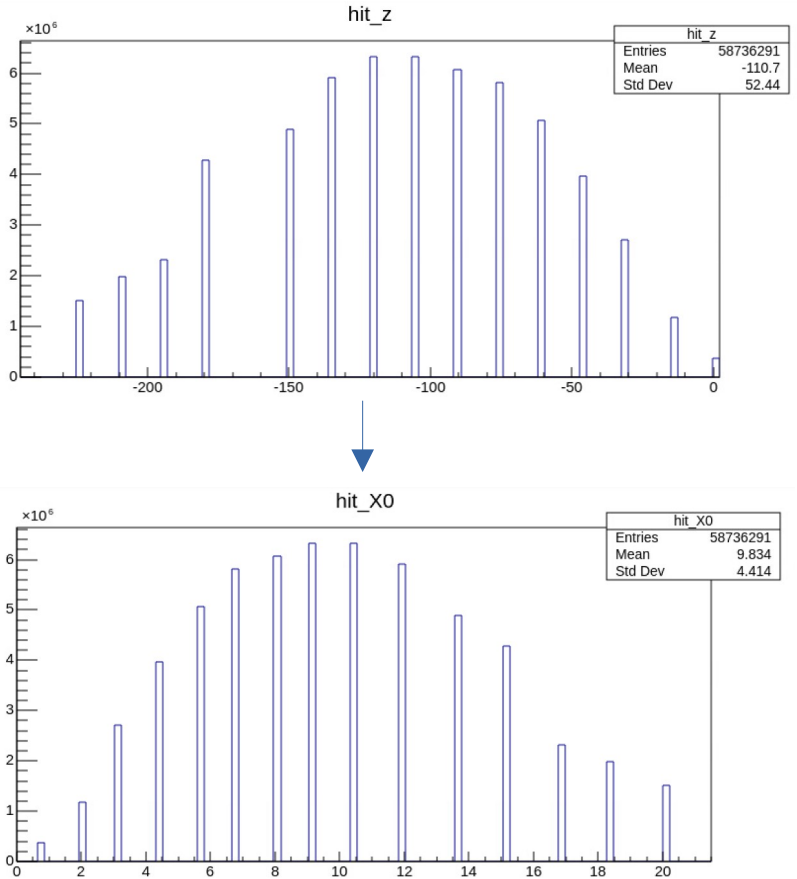

Honest question to the seniors: Does this look like a good calibration?

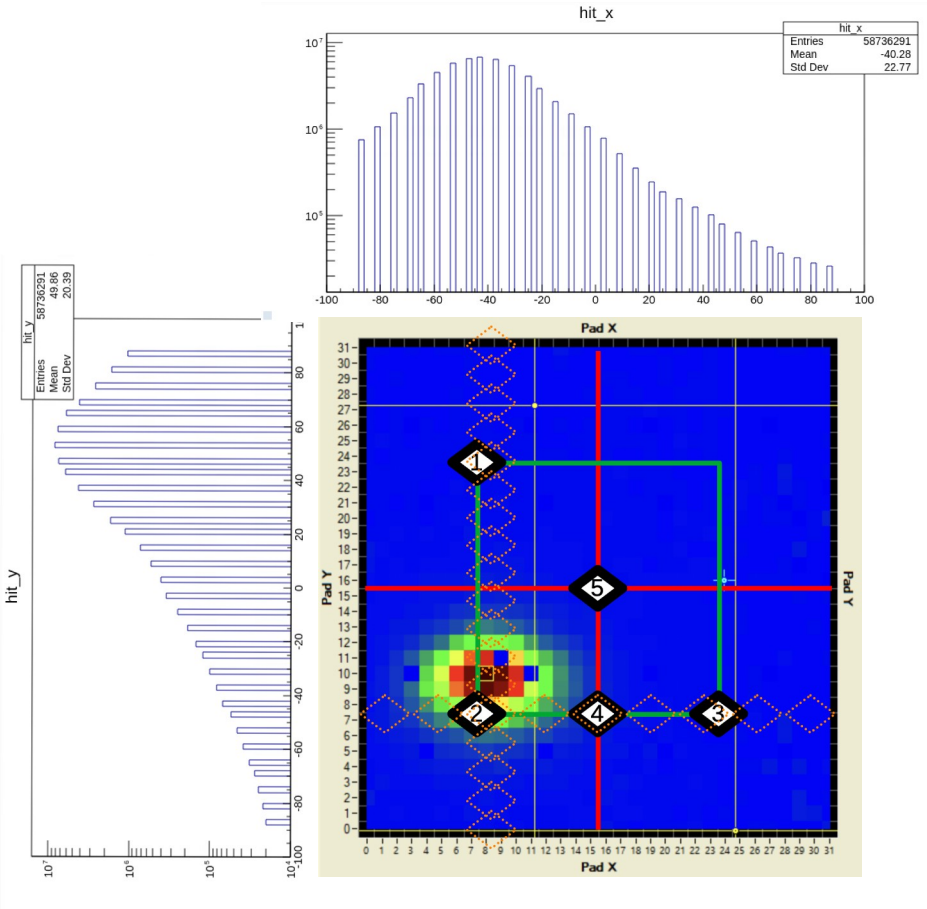
# Current state of our software

- Gaudi Alg. to convert raw hits to events “event_builder”
  - Builds the usual digi calo hits: bcid, event, hit_x, hit_energy, hit_ismasked, etc.
    - Inputs needed: mapping for FEV y COB, vector of W thickness, vector of z positions
      - Now it also saves hit_z, hit_X0 using the provided vectors. TBD: hit_energy_weighted
- First analysis/validation algorithm available: “event_validation”
  - Creates high-level variables (molière radius, bar_z, mip_likeness, etc.)
    - (optional) Creates different mip cuts (0, 0.5, 1)
  - Allows cuts in variables and re-saving the digi calo hits in a new .root file
  - Allows “analysis” mode: single mip cut and variables’ selection going into a new file
- **Everything is in key4hep + edm4hep (using auxiliary collections)**

# Some examples

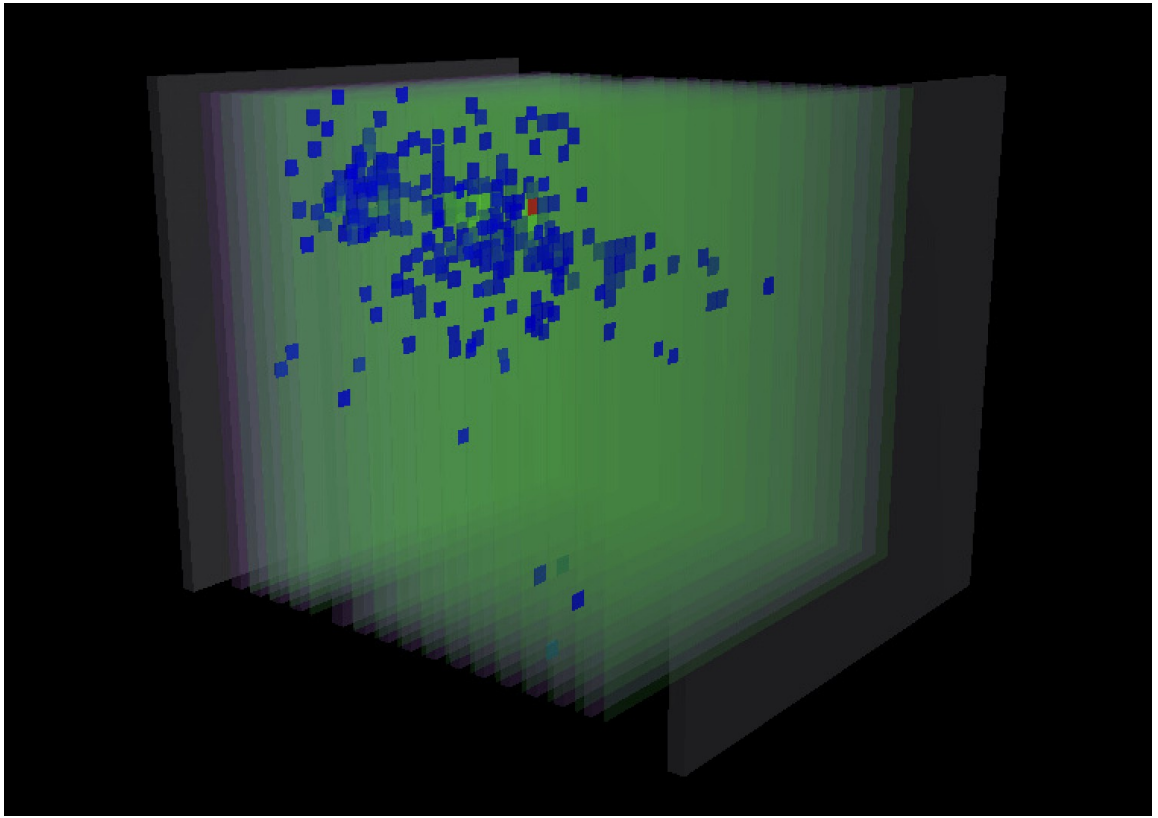
- Run13, P1



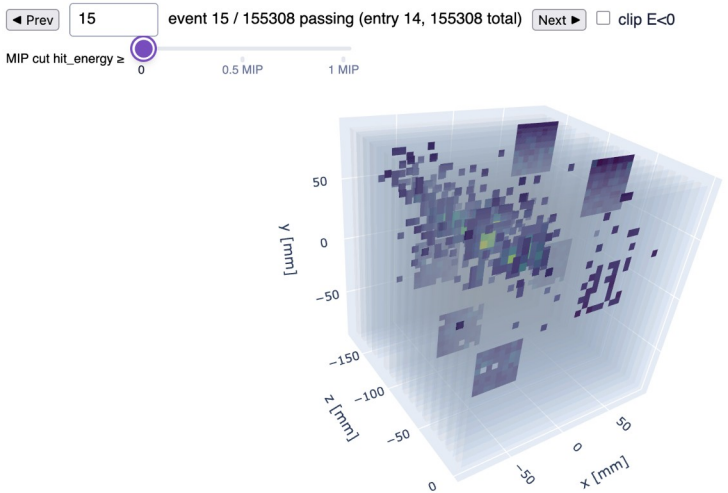
# Event_display (key4hep native)

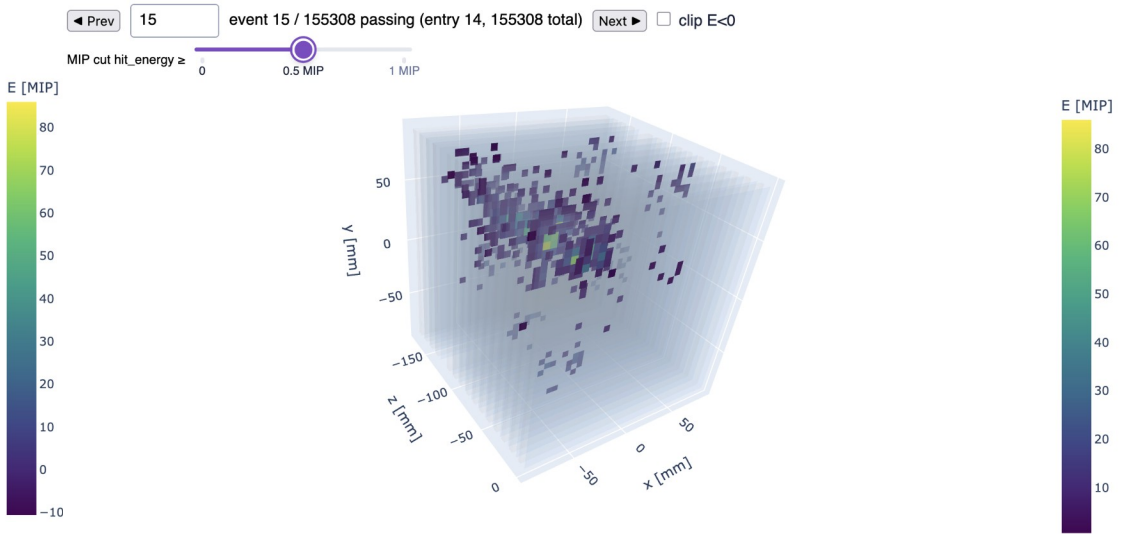
- Example of a 52 GeV e+ shower (run13, entry3)



# Event_viewer

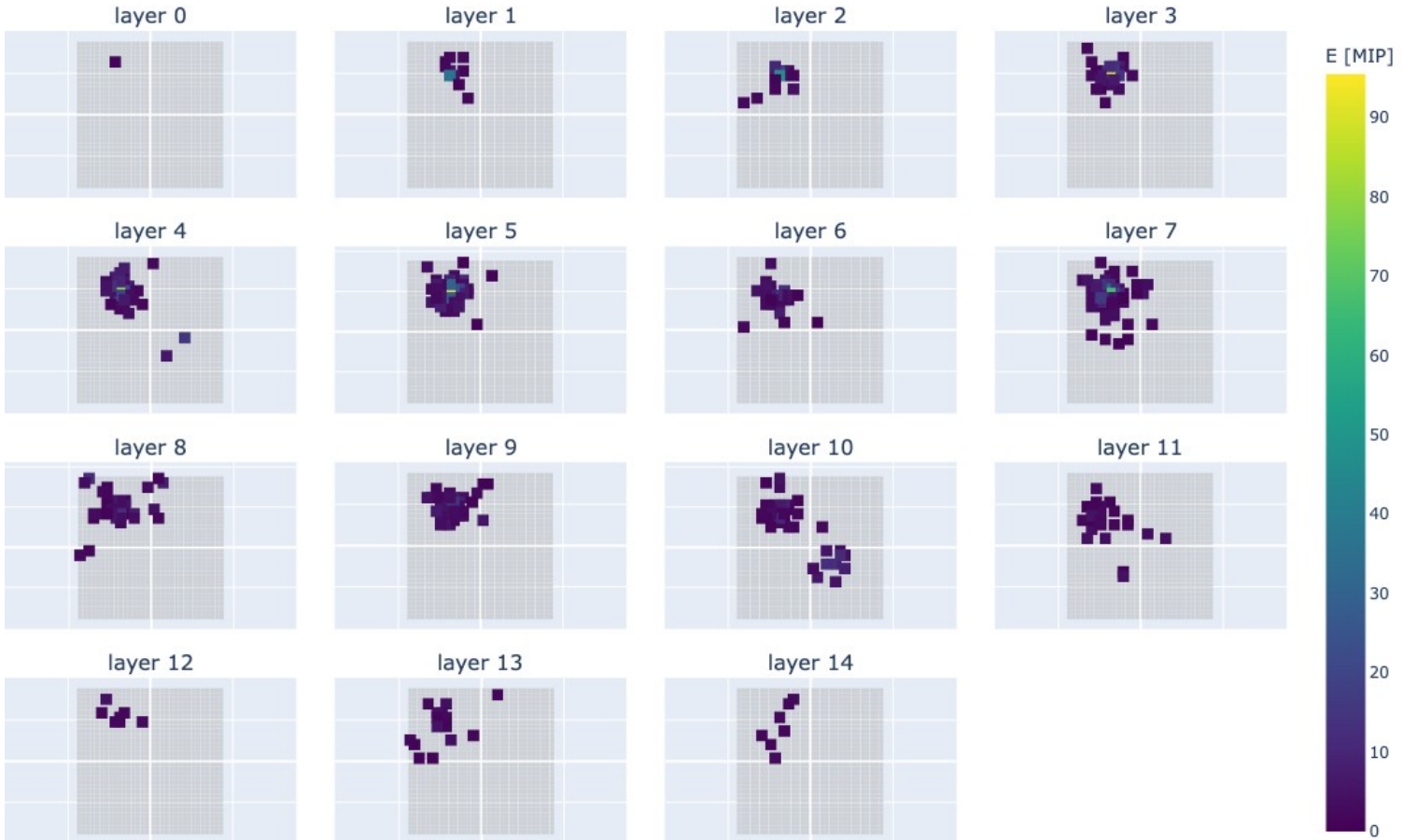
- A very noisy event (run13, evt4003)
  - Interactive mip cut: 0 & 0.5



# Event viewer: 2d-maps

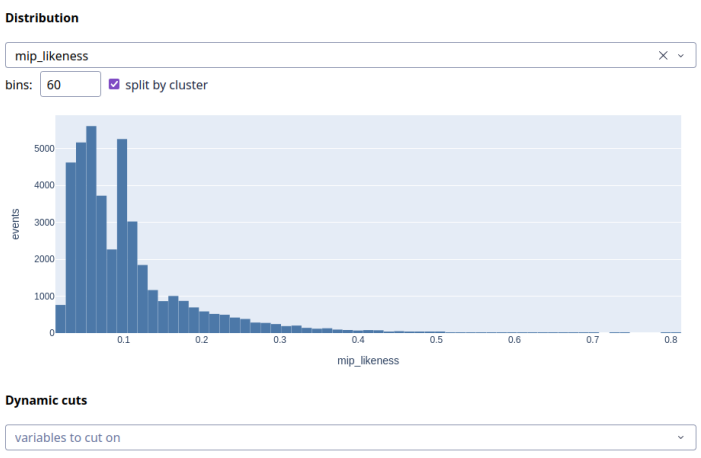
- Example (run13, evt2002)

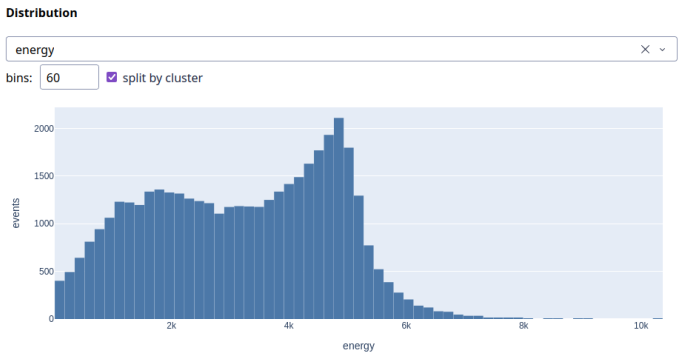


# Event viewer: Dynamic cuts

- Example of using a high-level variable (mip_likeness) to clean an energy distribution in real time:

**Distribution of the variable to discriminate**

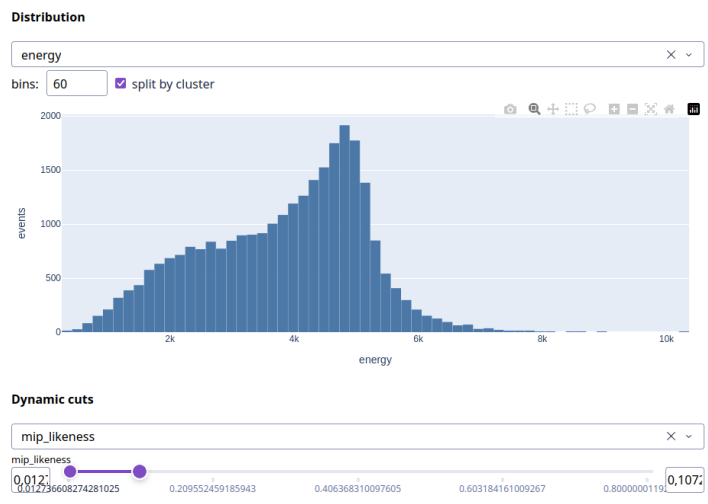



**Energy distribution before cut**

**Result**

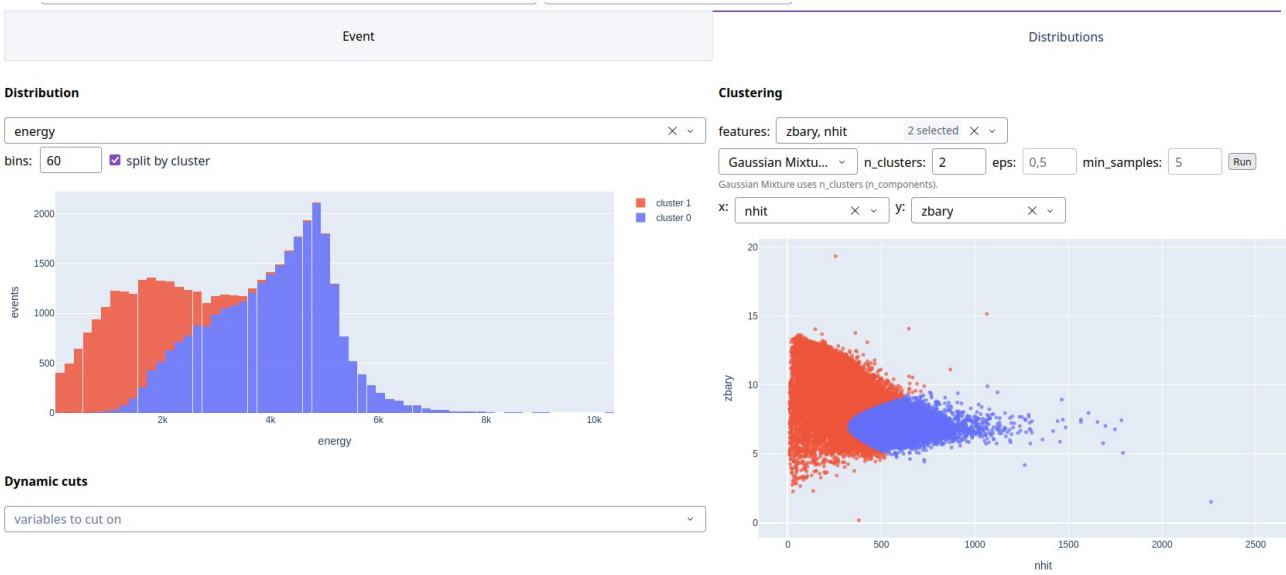
**Multiple cuts can be applied at the same time.**

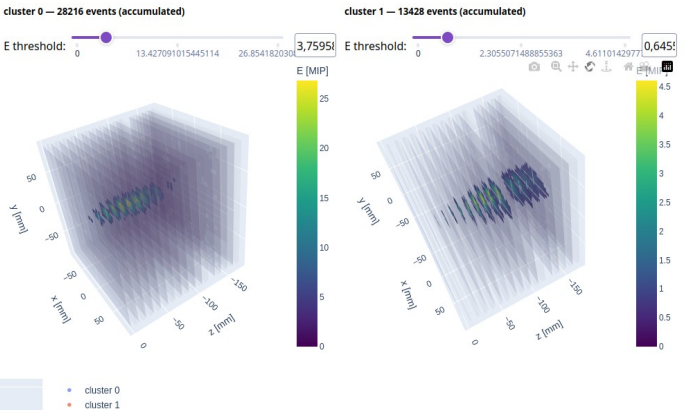


# Event viewer: Clustering

- Dynamic clustering exploration (different algorithms, different variables)

**Example: Gaussian mixture with z-barycenter and nhits clustering (2 clusters)**



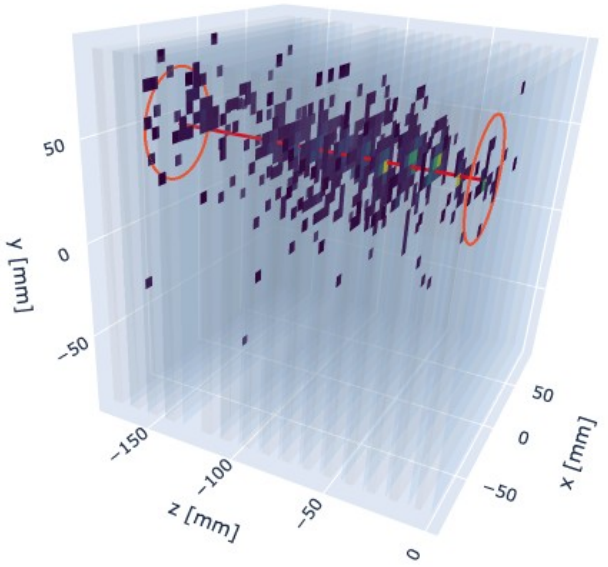



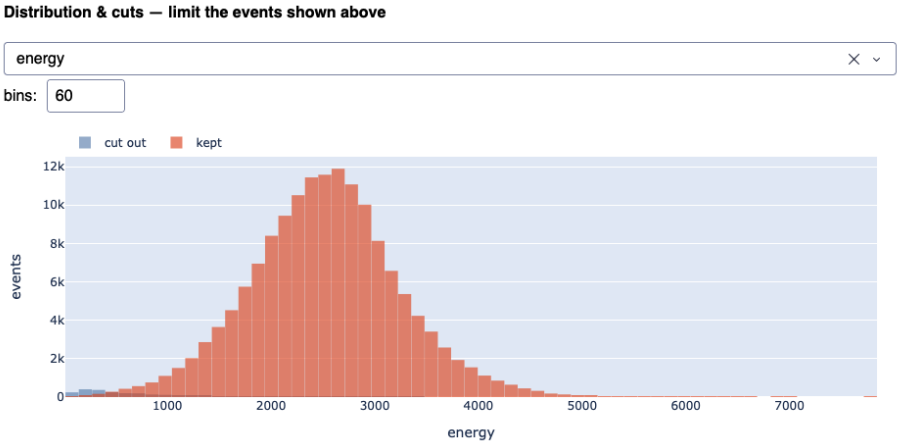

**Result: it separates showers with and w/o leakage.**

**Other variables and/or other algorithms could mitigate BG.**

# Updates in event_viewer (last week)

- Shower axis and Molière radius
  - Plotted event-by-event
- Better UI for interactive cuts
- Both propagated to the simulation event_viewer




# Newer implementations (First week post-TB) - II

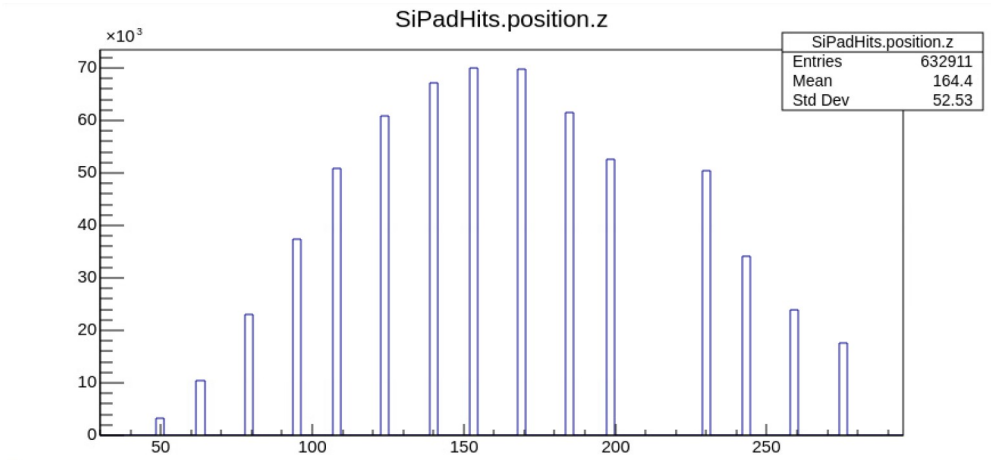
- Basic pipelines built, different possibilities and customization:
  - Some examples are already available:
    - Run000013
    - Calibration pipeline
    - Direct python k4run handlers like “run_full_pipeline_batch.py” for single use
  - Many others still TBD:
    - Some basic example on how to read the collections, etc. (Proposed by Vincent & Valeria)
    - More complex pipelines (macros with different runs and energies, etc. )

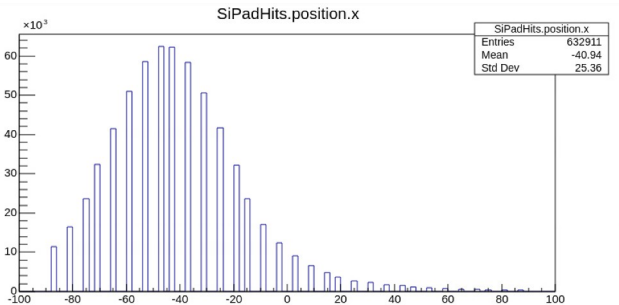
# Simulation

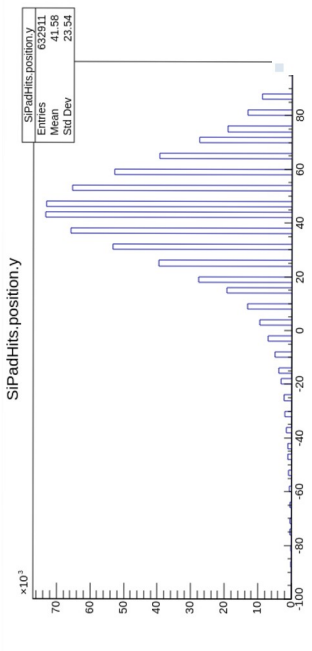
- Updated to the exact geometry used in the test-beam
- Added the validation variables and event_viewer
- Fully operational MIP calibration pipeline (at simhits level)
- Masking can be applied directly using the same MIP files from the TB, no extra files
- Corrected some small bugs in spacing and segmentation
- Yukun is implementing realistic digitization (Or has done already)
  - Not yet in the repository
- (TBD) Adding a Gaudi algorithm with noise simulation:
  - Two layers of noise simulation: random hits + coherent chip fluctuation

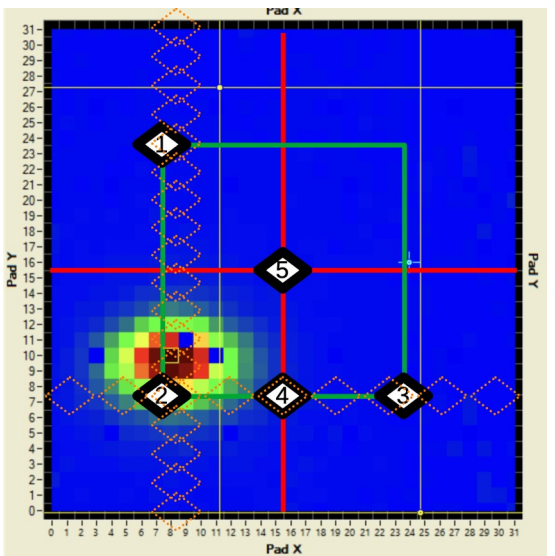
# Position matching

- The detector was flipped in z direction
  - The simulation is adapted so the XY mappings work
    - The z position is flipped later



# Next steps

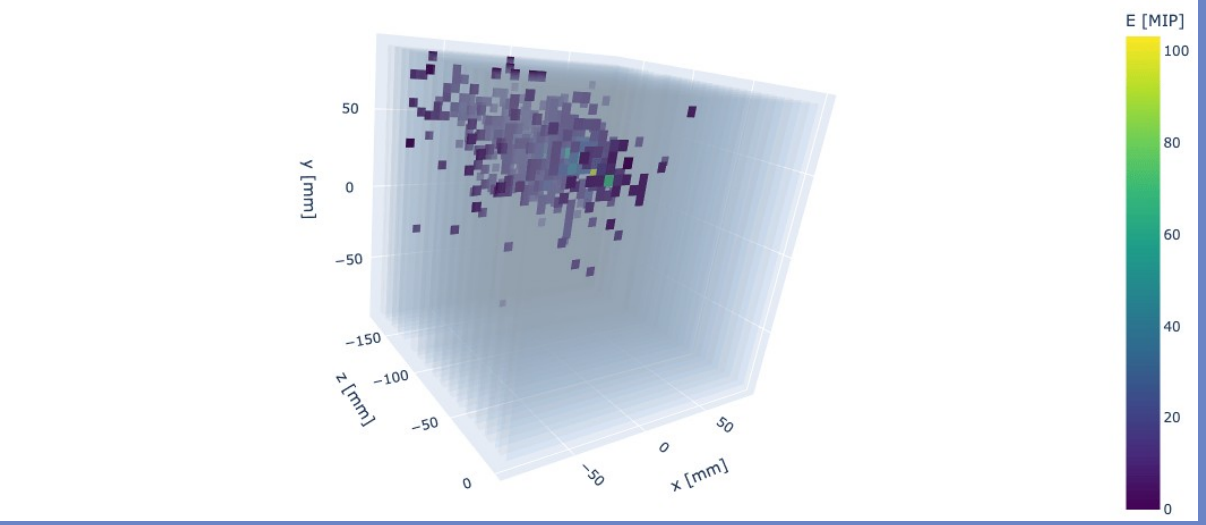
- **External review**: We need others to test this stack, test for bugs and user-level improvements
  - Get your hands to it, try to run it, convert some events, Xcheck with previous conversion, build events, have a look, use the event_viewer, etc.
    - Important: Change the production folders for your own test, the repository right now writes in the common eos folder

- About converted files
  - I think we should reconvert everything progressively, we need to split the task between the people with rights to write/delete files in /eos/experiment/drdcalo/siw-ecal/
  - Roman also proposed to add the new conversion into eudaq in the medium-long term (Ready for Spring TB)

- Going on:
  - Cleaning muon and pion backgrounds from electron events
    - Using high-level variables first (and clustering of variables afterwards)
    - Using ML approaches (Olmo is thinking of an autoencoder)
    - Cleaning noise from events (Besides MIP cut, cleaning hits with r>2 $r_M$ or similar?)
  - ACTS tracks (I have it in my SHiP prototype software, it can be adapted to study muons)

# More next steps

- Prepare pipelines for analyses (Open for self-nomination)
  - Energy linearity and resolution
  - Energy reconstruction
  - Sim vs data comparisons
  - Etc.
- You can start building your own analysis in a new git branch and merge it once is operational
  - But, important, try to follow the same logic:
    - /Gaudi_source/ for the C++ source of your Gaudi algorithms (hit, cluster, pid, tracks)
    - /Gaudi_jobs/ for running pipelines
  - Getting your analysis imported here is fairly easy:
    - Adapting from Marlin is almost-trivial
    - Adapting from python is easy using agents, very tedious doing all by hand
  - In the future an /Analysis/ folder could contain high-level ML/AI things in python




## Which one is TB data and which one is simulation?