Delphes 3

Framework for fast simulation of a generic collider experiment

Michele Selvaggi, for the Delphes Team

Université Catholique de Louvain (UCL)
Center for Particle Physics and Phenomenology (CP3)

ACAT, Beijing
May 18, 2013

Introduction

Detector simulation

- Full simulation (GEANT):

  - **simulates** particle-matter interaction (including e.m. showering, nuclear int., brehmstrahlung, photon conversions, etc ...) → 10 s /ev

- Experiment Fast simulation (ATLAS, CMS ...):

  - **simplifies** and makes faster simulation and reconstruction → 1 s /ev

- Parametric simulation:

  Delphes, PGS**:**

  - **parameterize** detector response, reconstruct complex objects → 10 ms /ev

  TurboSim

  - **no detector**,parameterize object response, parton ↔ reco

Detector simulation

- Full simulation (GEANT):

  - **simulates** particle-matter interaction (including e.m. showering, nuclear int., brehmstrahlung, photon conversions, etc ...) $\rightarrow$ 10 s /ev

- Experiment Fast simulation (ATLAS, CMS ...):

  - **simplifies** and makes faster simulation and reconstruction $\rightarrow$ 1 s /ev

- Parametric simulation:

  Delphes:

  - **parameterize** detector response, reconstruct complex objects $\rightarrow$ 10 ms /ev

  TurboSim

  - **no detector**,parameterize object response, parton $\leftrightarrow$ reco

Parametric simulation

- What do we **expect** from Delphes **parametric detector simulation** ?

  - fast
  - realistic enough
  - flexible detector geometry
  - user-friendly
  - flexible I/O (modular)

- When do you need Delphes?

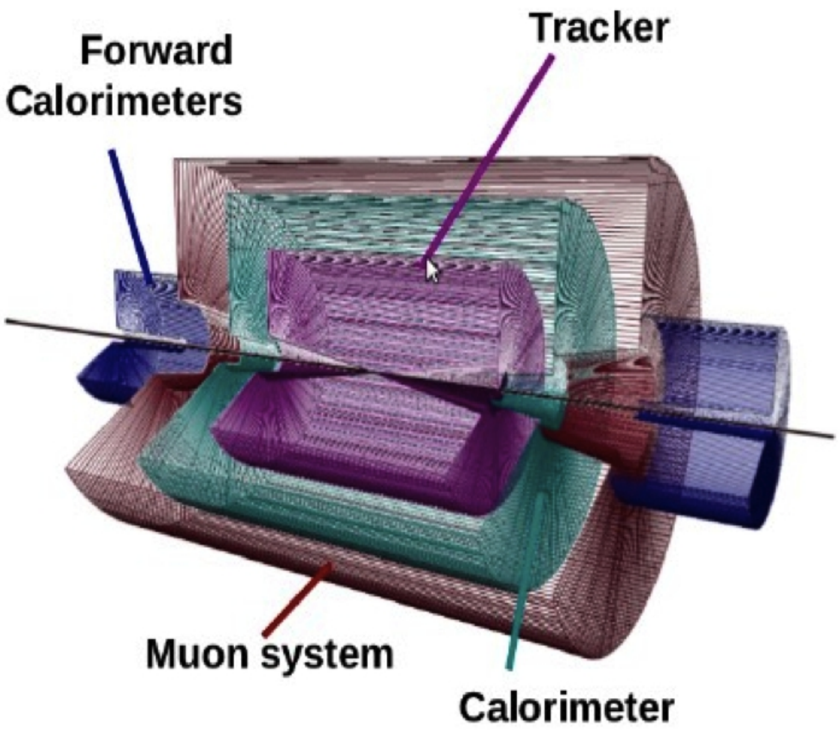
  → more advanced than parton-level studies
  → scan big parameter space (SUSY-like ...)
  → preliminary tests of new geometries/resolutions (upgrades, Snowmass)
  → testing analysis methods (multivariate/Matrix Element)
  → educational purpose (master thesis)

Workflow

What is Delphes?

- **Delphes** is a **modular framework** that simulates of the response of a multipurpose detector

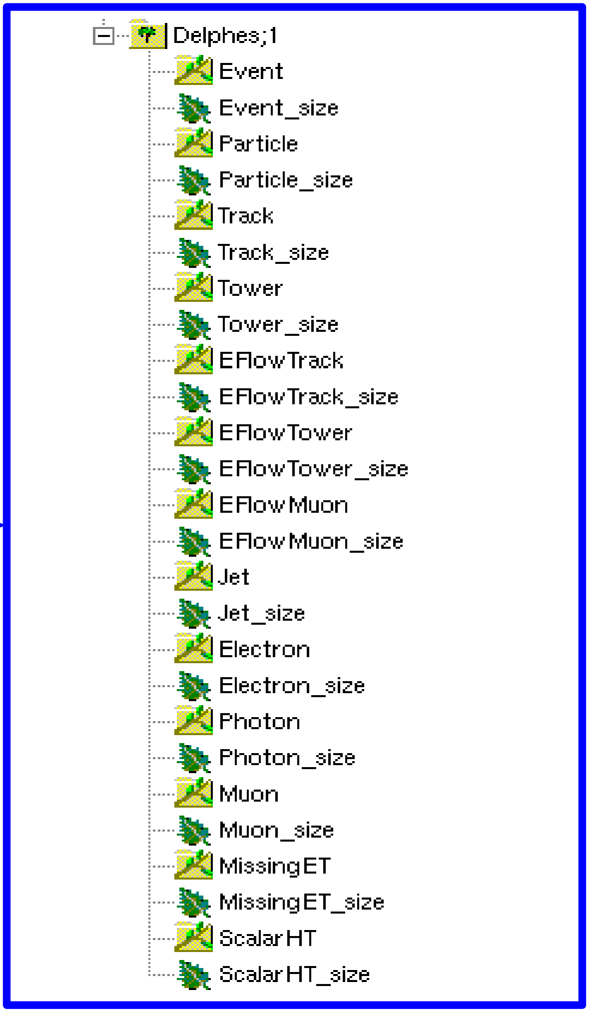
- **simulates**:
  - charged particle propagation in magnetic field: **tracking**
  - electromagnetic and hadronic **calorimeters**
  - **muon** system



- **reconstructs**:
  - leptons (electrons and muons)
  - photons
  - jets and missing transverse energy
  - taus and b's

Technical features

- **modular** C++ code, uses ROOT classes
- Input
  - Pythia/Herwig output (HepMC,STDHEP)
  - LHE (MadGraph/MadEvent)
- Output
  - ROOT trees



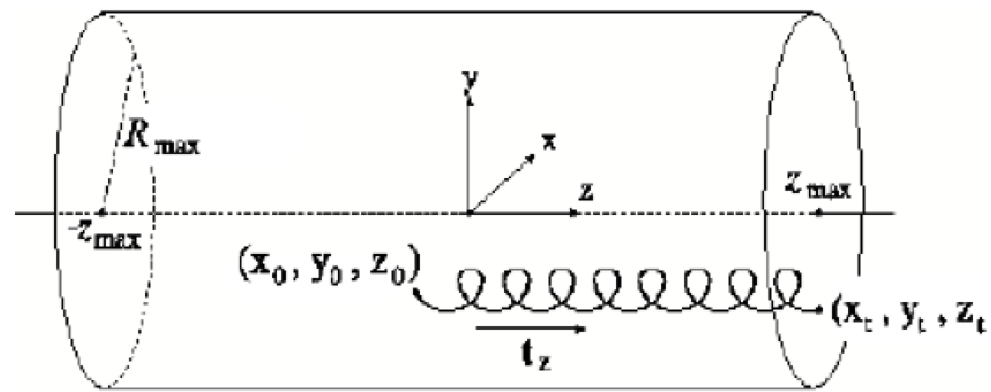
- Configuration file
  - define geometry
  - resolution/reconstruction/selection criteria
  - output object collections

Charged Particle Propagation

- Charged particles are propagated in the magnetic field until they reach the calorimeters

- Propagation parameters:

  - magnetic field B
  - radius and half-length (R_{max}, z_{max})



- Efficiency/resolution depends on:

  - particle ID
  - transverse momentum
  - pseudorapidity

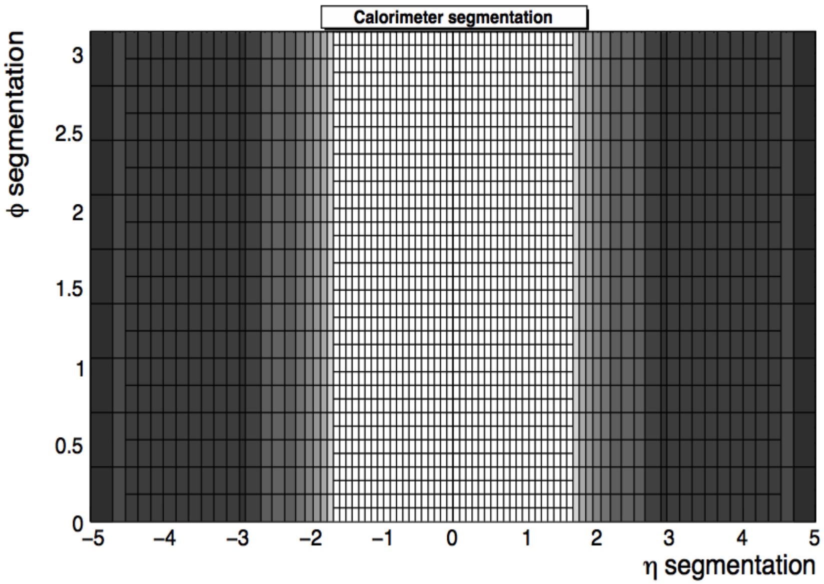
```
# efficiency formula for muons
add EfficiencyFormula {13} {                                             (pt <= 0.1)    * (0.000) + \
                                            (abs(eta) <= 1.5) * (pt > 0.1   && pt <= 1.0)    * (0.750) + \
                                            (abs(eta) <= 1.5) * (pt > 1.0)                   * (1.000) + \
                         (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1   && pt <= 1.0)    * (0.700) + \
                         (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0)                   * (0.975) + \
                         (abs(eta) > 2.5)                                                    * (0.000)}
```

Not real tracking/vertexing !!
→ no fake tracks/ conversions (but can be easily implemented)
→ no dE/dx measurements

Calorimetry

- em/had calorimeters have same **segmentation** in eta/phi

- Each particle that reaches the calorimeters **deposits a fraction of its energy** in one ECAL cell (f_{EM}) and HCAL cell (f_{HAD}), depending on its type:



particles	f_{EM}	f_{HAD}
$e\ \gamma\ \pi^0$	1	0
Long-lived neutral hadrons (K^0_S, Λ^0)	0.3	0.7
$\nu\ \mu$	0	0
others	0	1

- Particle energy is **smeared** according to the calorimeter cell it reaches

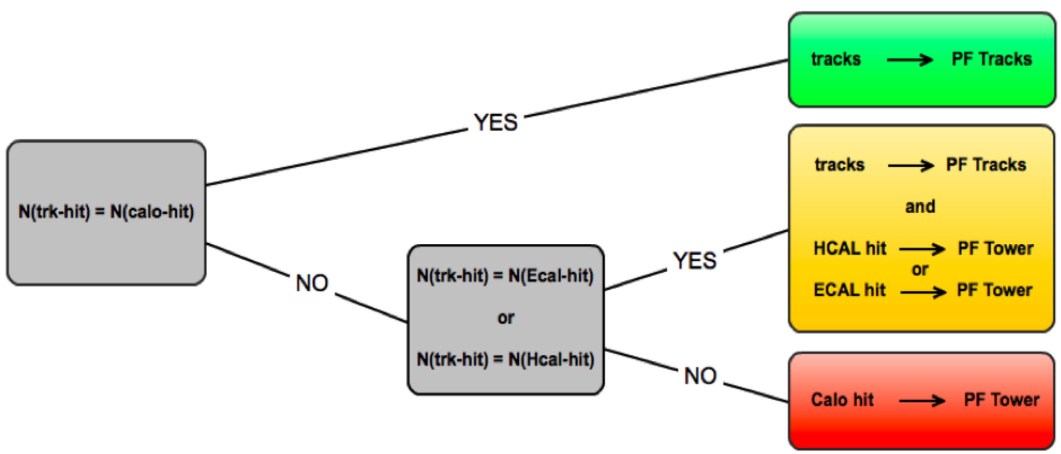
$$\boxed{E_{smeared} = gauss(f_{EM}E, \sigma_{EM}(\eta)) + gauss(f_{HAD}E, \sigma_{HAD}(\eta))}$$

$$\sigma^2(\eta) = N^2(\eta) + S^2(\eta)E + C^2(\eta)E^2$$

Particle-Flow algorithm

“Particle-flow” approach aims at **maximizing** object reconstruction **resolution** by using all sub-detector information (in Delphes tracking, ECAL and HCAL) → **particle-flow candidates** can be used as input for **jets and E_T^{miss}**

- For each calorimeter cell:



Leptons, photons

- Muons/photons/electrons
  - **identified** via their PDG id
  - inside the **tracker coverage** for electrons and muons
  - muons do not deposit energy in calo (independent smearing parameterized in p_T and η)
  - electrons and photons smeared according to electromagnetic calorimeter resolution
- Isolation: $$\mathrm{rel.Iso} = \frac{\sum_{\Delta R<0.5} p_T^{track}}{p_T}$$ → modular structure allows to easily define different isolation

  If rel.Iso ~ 0, the lepton is isolated

- Not taken into account:
  - fakes, punch-through, brehmstrahlung, conversions

Jets / E_T^{miss} / H_T

- **FastJet** library used for jet clustering
  - all clustering algos supported: **anti-kT**, SisCone, ...
- Jets, E_T^{miss} and H_T quantities can be formed from:
  - **calorimeter towers**
  - “**particle-flow**” candidates:

$$\overrightarrow{E_T}^{miss} = -\sum_i \overrightarrow{p_T}(i), \qquad H_T = \sum_i |\overrightarrow{p_T}(i)|,$$

b and tau jets

- <u>b-jets</u>

  - if **b** parton is found in a cone ΔR w.r.t jet direction
    → apply **efficiency**
  - if **c** parton is found in a cone ΔR w.r.t jet direction
    → apply **c-mistag rate**
  - if **u,d,s,g** parton is found in a cone ΔR w.r.t jet direction
    → apply **light-mistag rate**

b-tag **flag** is then stored in the jet collection

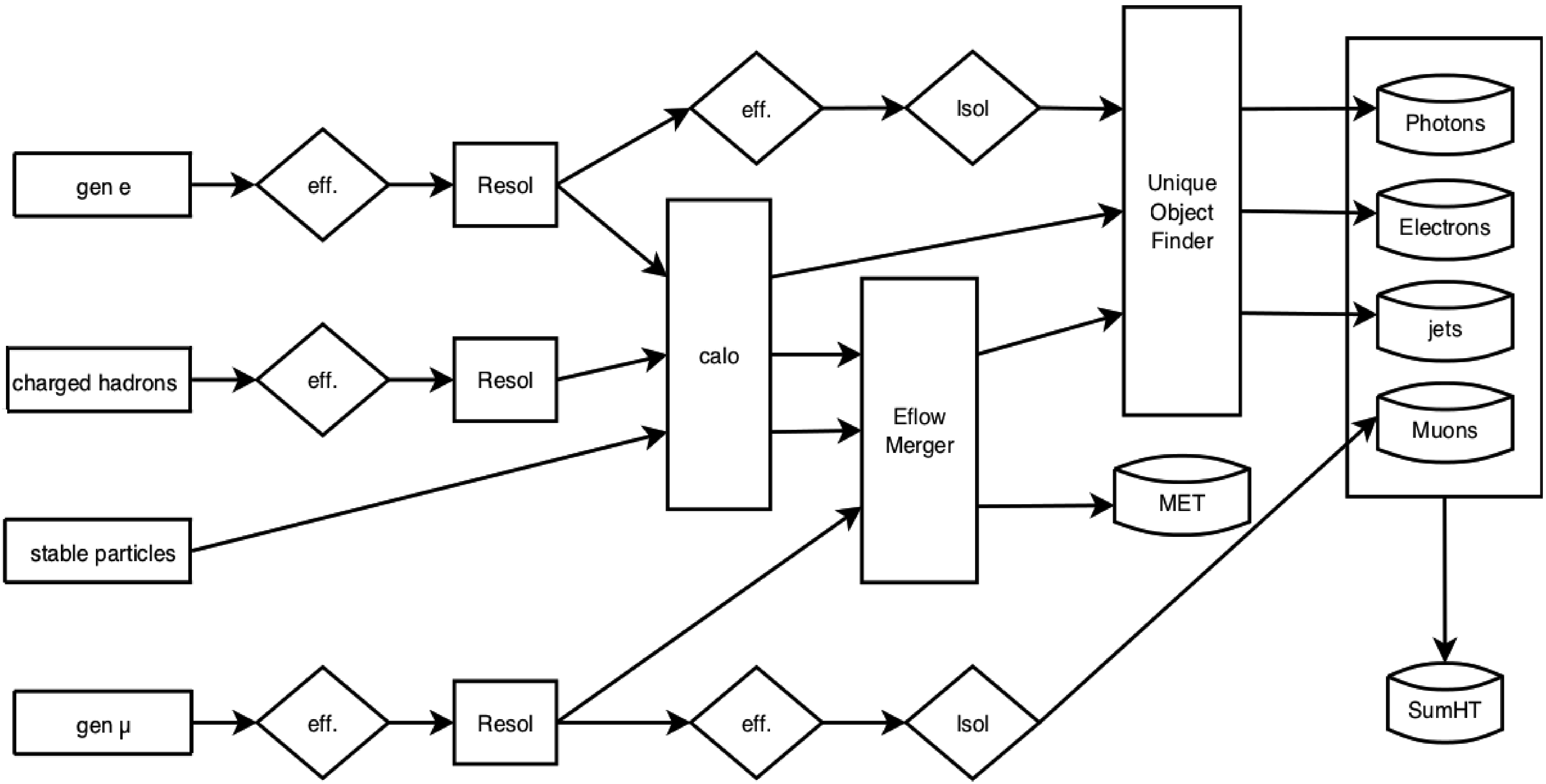
- <u>tau-jets</u>

  - if tau lepton is found in a cone ΔR w.r.t jet direction
    → apply **efficiency**
  - else
    → apply **tau-mistag rate**

tau jets have their own collection (no leptonic tau decays)

can define **p_T and η dependent** efficiency and mistag rate

Modularity in action

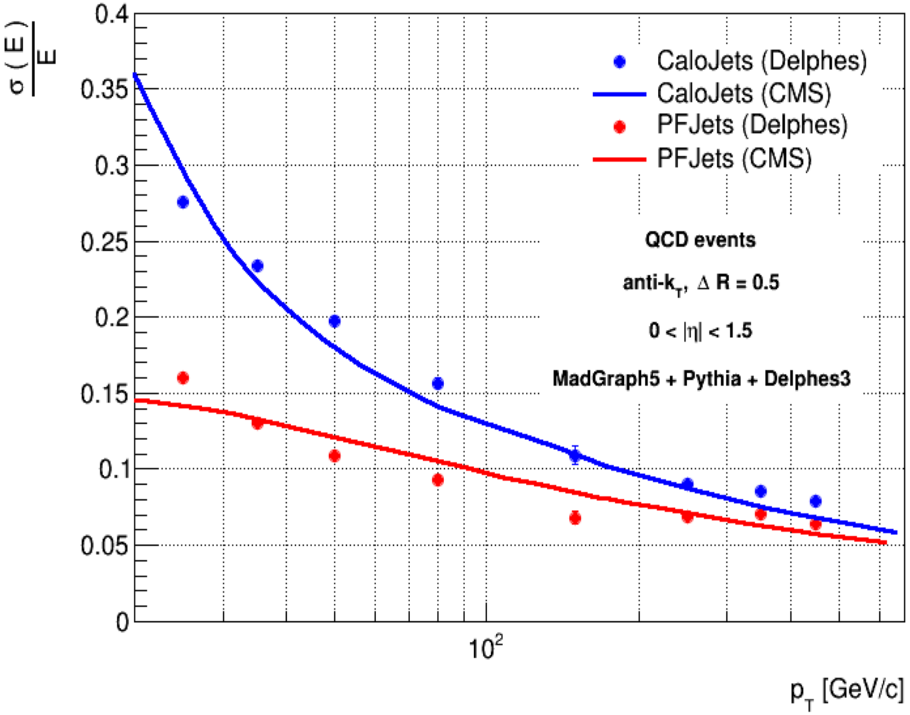


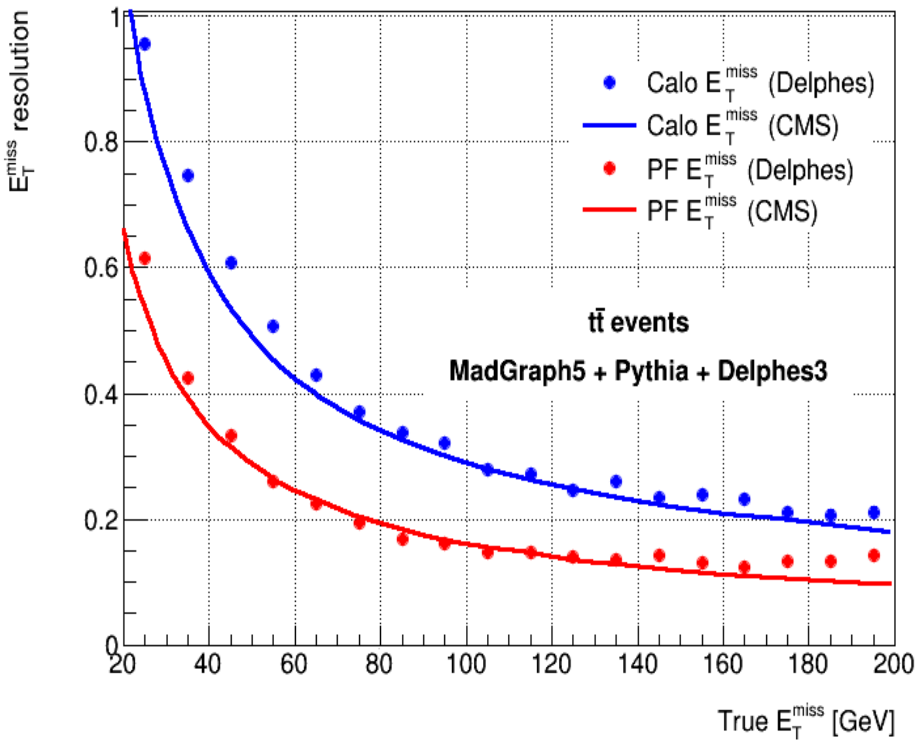
Does this approach work?

Validation: jets and E_T^{miss}

- Electrons, muons and photons are auto-validated by construction
- **Jets** and **missing energy** need to be tested:

CMS resolution from: The CMS Collaboration, CMS-PAS-PFT-09-001

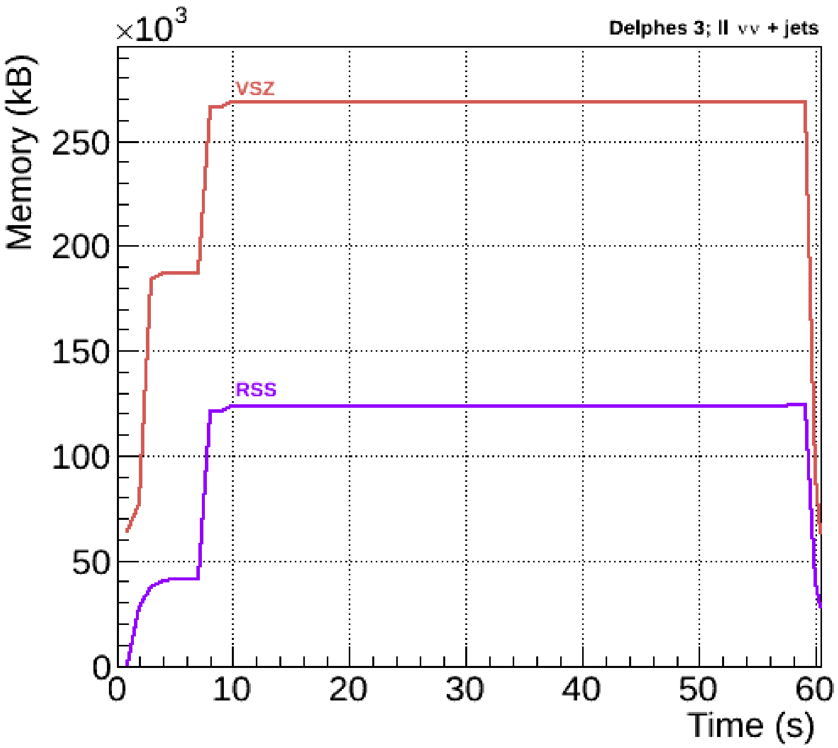


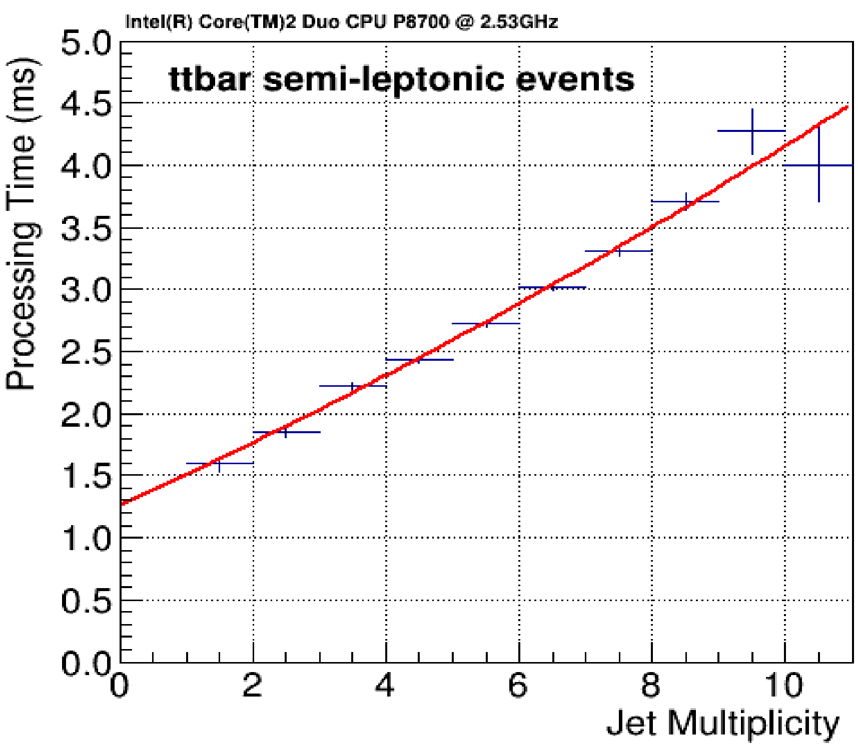


→ **excellent agreement**

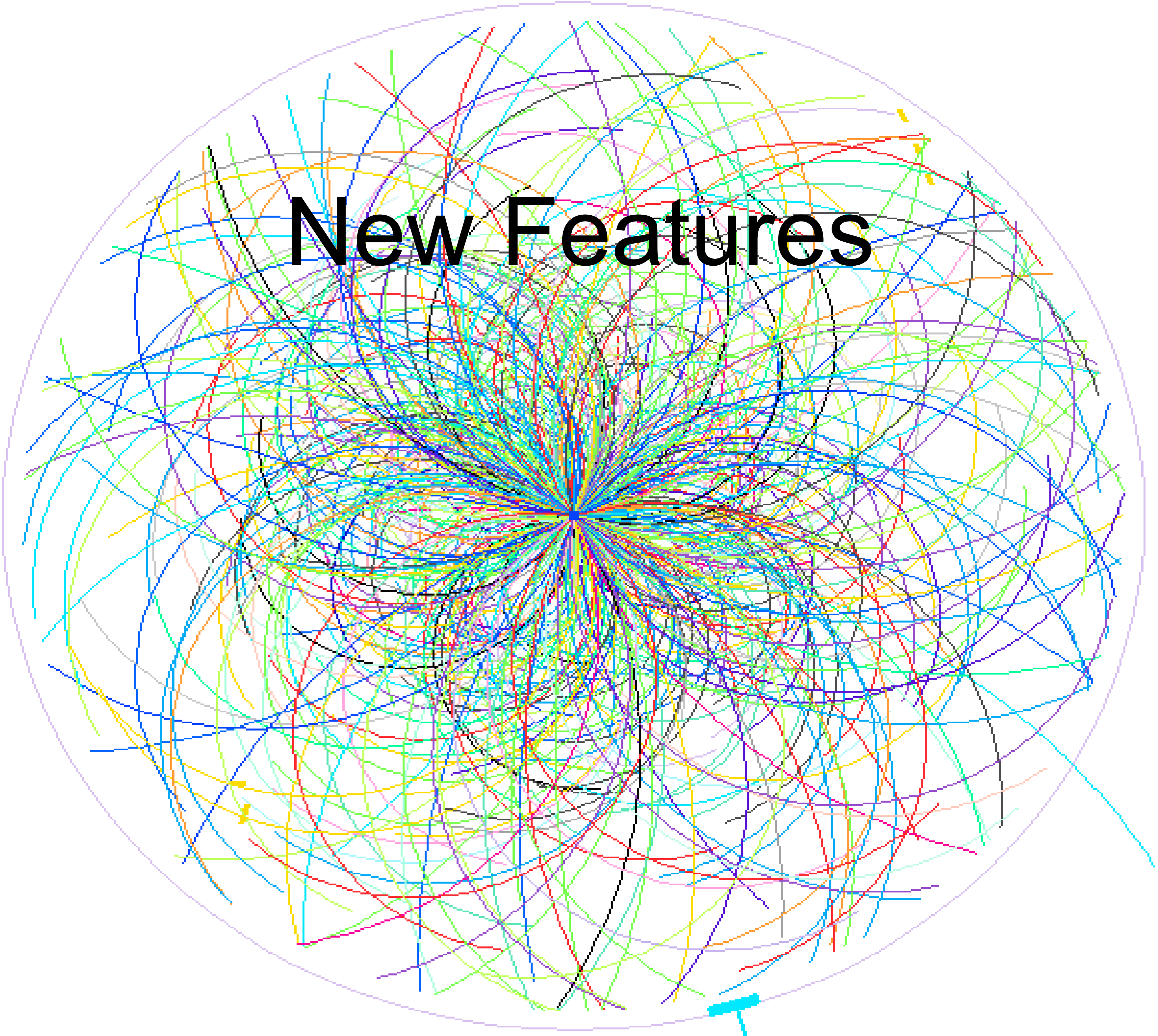
Performances

- **small memory footprint**
- **short processing time** with a standard laptop
- output 50% **smaller** than HepMC!





New Features



Pile-up motivations

- **Pile-up** becomes an issue at **high luminosity** LHC
  - reduced **efficiency**
  - worsened **resolution** (jets, E_T^{miss})
  - degraded **isolation**
  - **fake** tracks, jets
- **Efficiencies** and **resolutions** can be **tuned by hand** to mimic pile-up
- Fake objects need to be simulated. Also, we want to have some predictive power:

  → We therefore introduced: tunable **simulation** of pile-up
  pile-up **substraction** procedure.

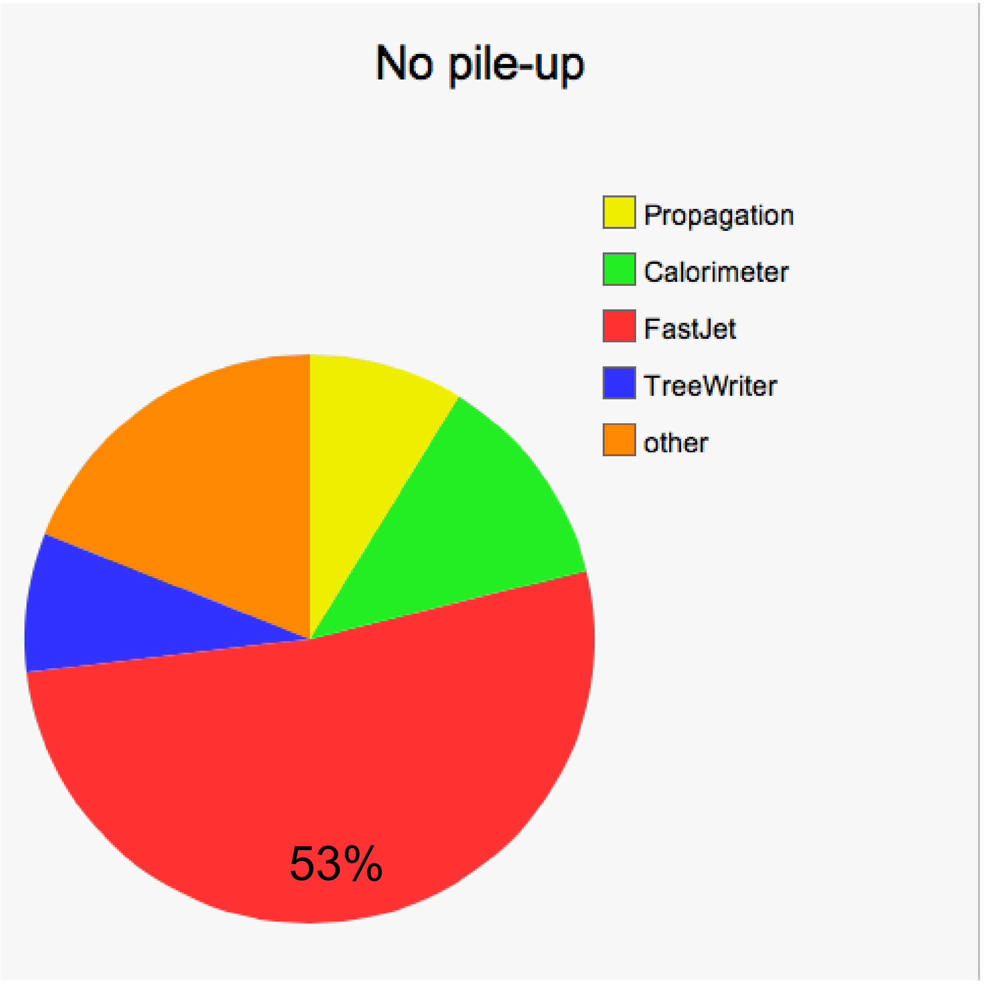
- This new feature is being actively validated in collaboration with the groups preparing results for Snowmass 2013 (CMS and ATLAS).

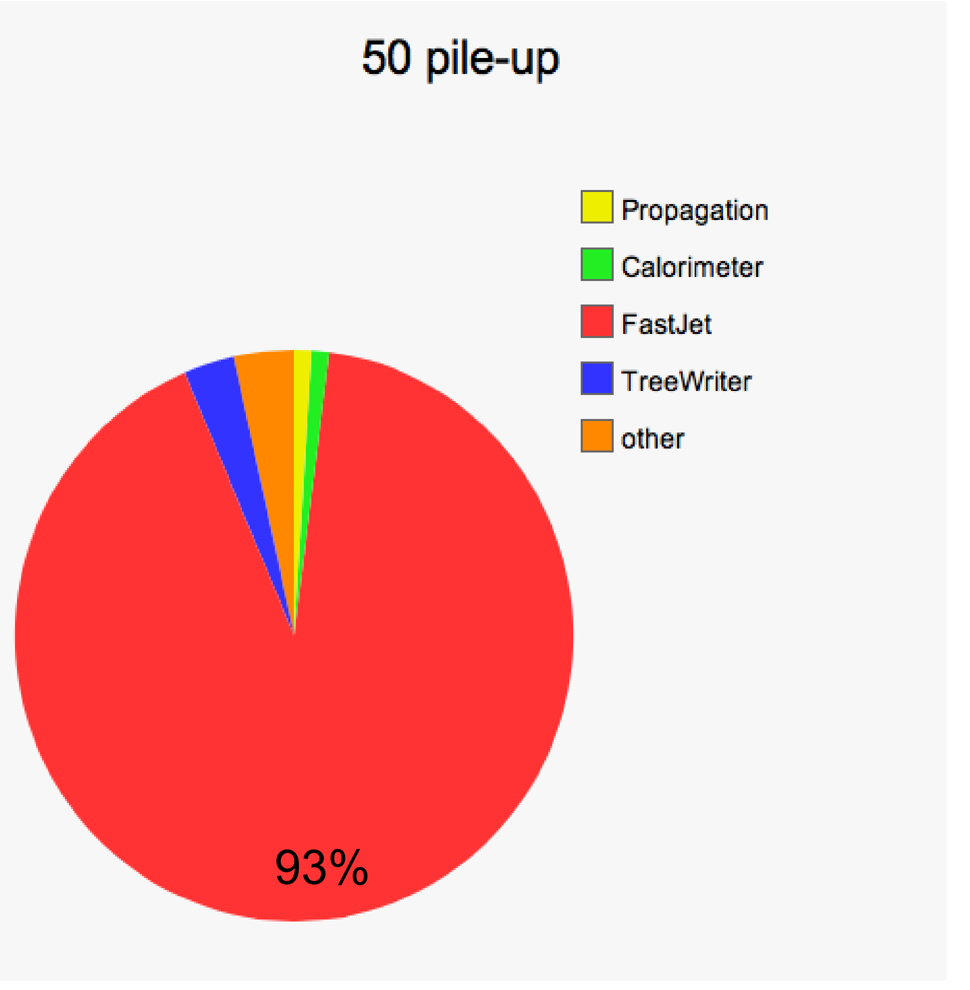
Pile-up implementation

- Pile-up is implemented in Delphes **since version 3.0.4**
  - **mixes** N minimum bias events with hard event sample
  - spreads **poisson(N)** events along z-axis with configurable spread
    - if **z < |Zres|** keep all **charged and neutrals** (→ ch. particles too close to hard scattering to be rejected)
    - if **z > |Zres|** keep only **neutrals** (perfect charged subtraction)
- With this approach :
  - charged subtraction is already done at the mixing level (faster)
  - allows user to tune amount of charged particle subtraction by **adjusting Z spread/resolution**
- Residual pileup substraction is needed for jets and isolation.
  - Use the FastJet Area approach (add ref.)
    - compute ρ = event pile-up density
    - jet correction : $p_T \to p_T - \rho A$ (JetPileUpSubtractor)
    - isolation : $\sum p_T \to \sum p_T - \rho \pi R^2$ (Isolation module itself)

Performances (II)

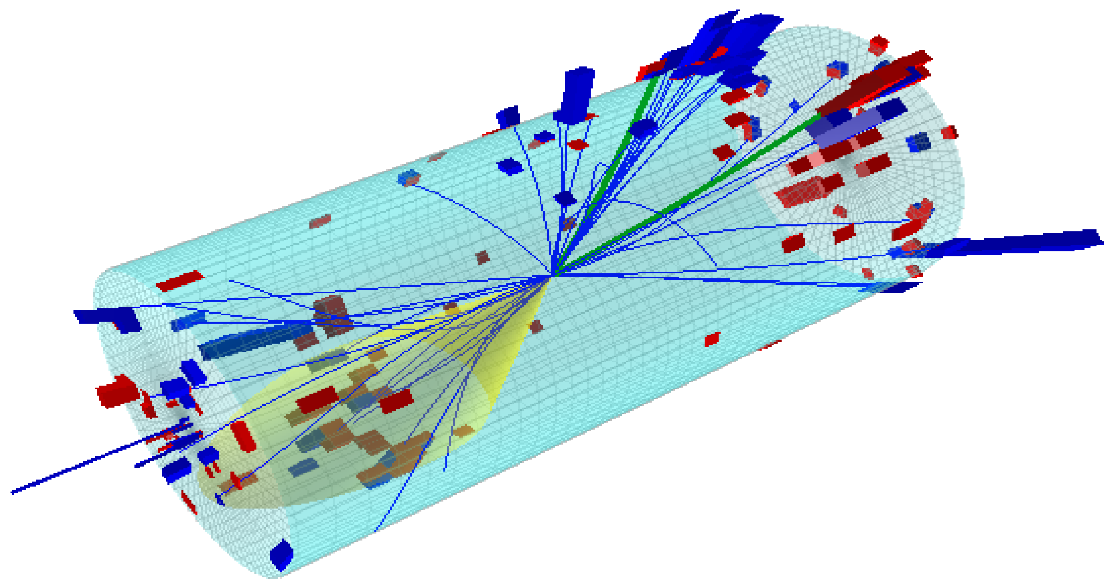
Time consumption per module

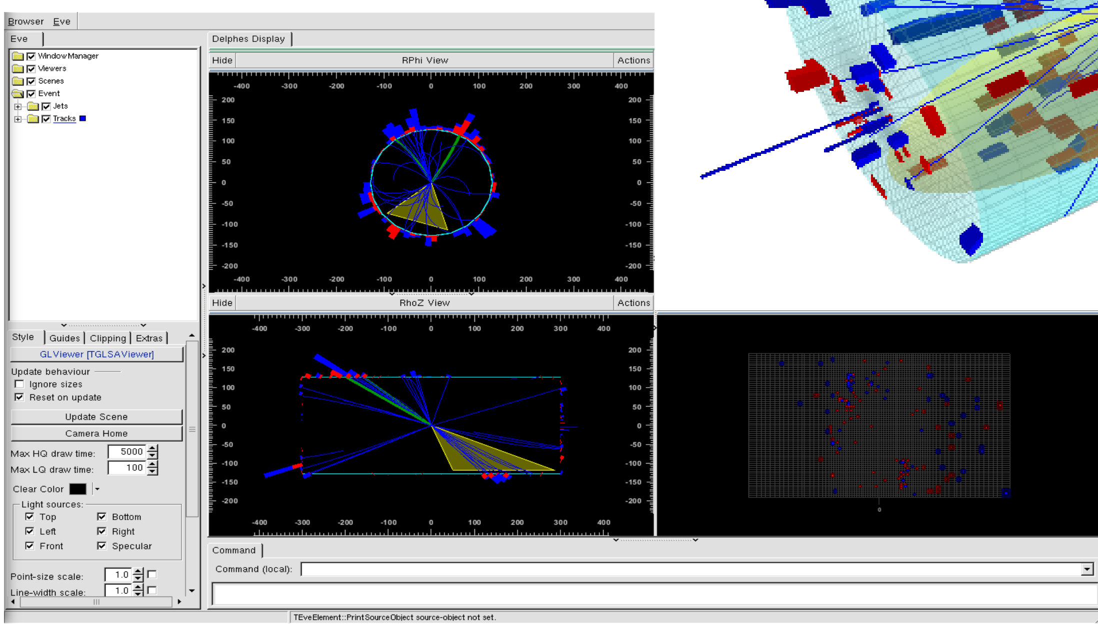




Event-display

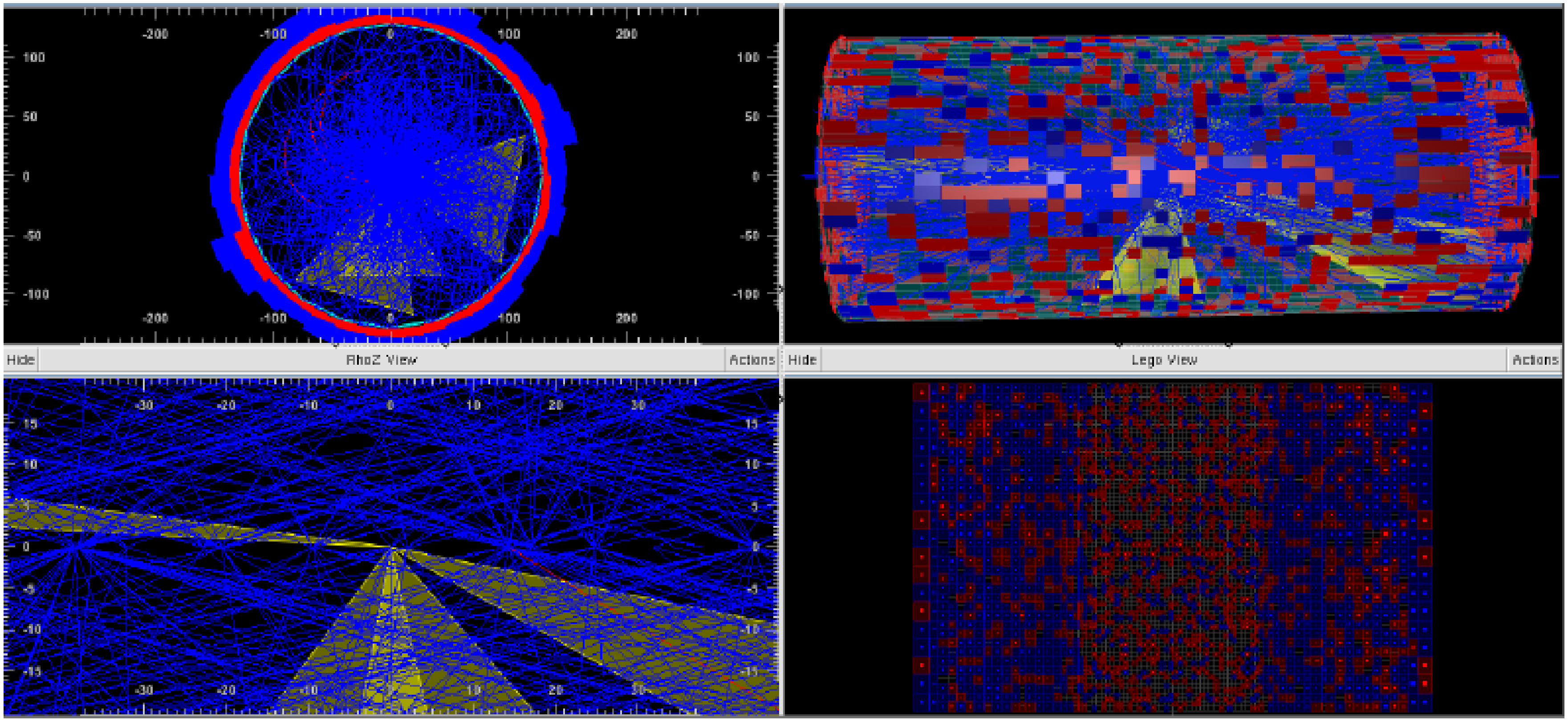
- A basic **event-display** is provided, based on ROOT EVE
  - Displays **tracks, calo-towers, jets**.
  - Useful for debugging
  - More detailed version planed.





Event-display with pile-up

$Z_{spread} \approx Z_{res}$, 50 av. pile-up



Conclusions

Development

- Delphes project started back in 2007

- Since 2009, its development is **community-based**
  - **ticketing system** for improvement and bug-fixes
    → user proposed patches
  - **Quality control** and **core development** is done at the UCL

- Team
  - Two research scientists (P. Demin, J. de Favereau)
    Website, repository, releases
    Core developments and code optimization
    Support

  - One post-doc (M. Selvaggi) and one PhD student (A. Mertens)
    Re-optimization of the performances, validation
    Implementation of new features
    Support

- **Widely** tested and used by the community > 100 citations !!

Conclusions

- **Delphes 3** is out, with **major improvements**:

  - modularity
  - pile-up implementation
  - revamped particle flow algorithm
  - new visualization tool based on ROOT EVE
  - default cards giving results on par with published performance from LHC experiments
  - now fully integrated within MadGraph5

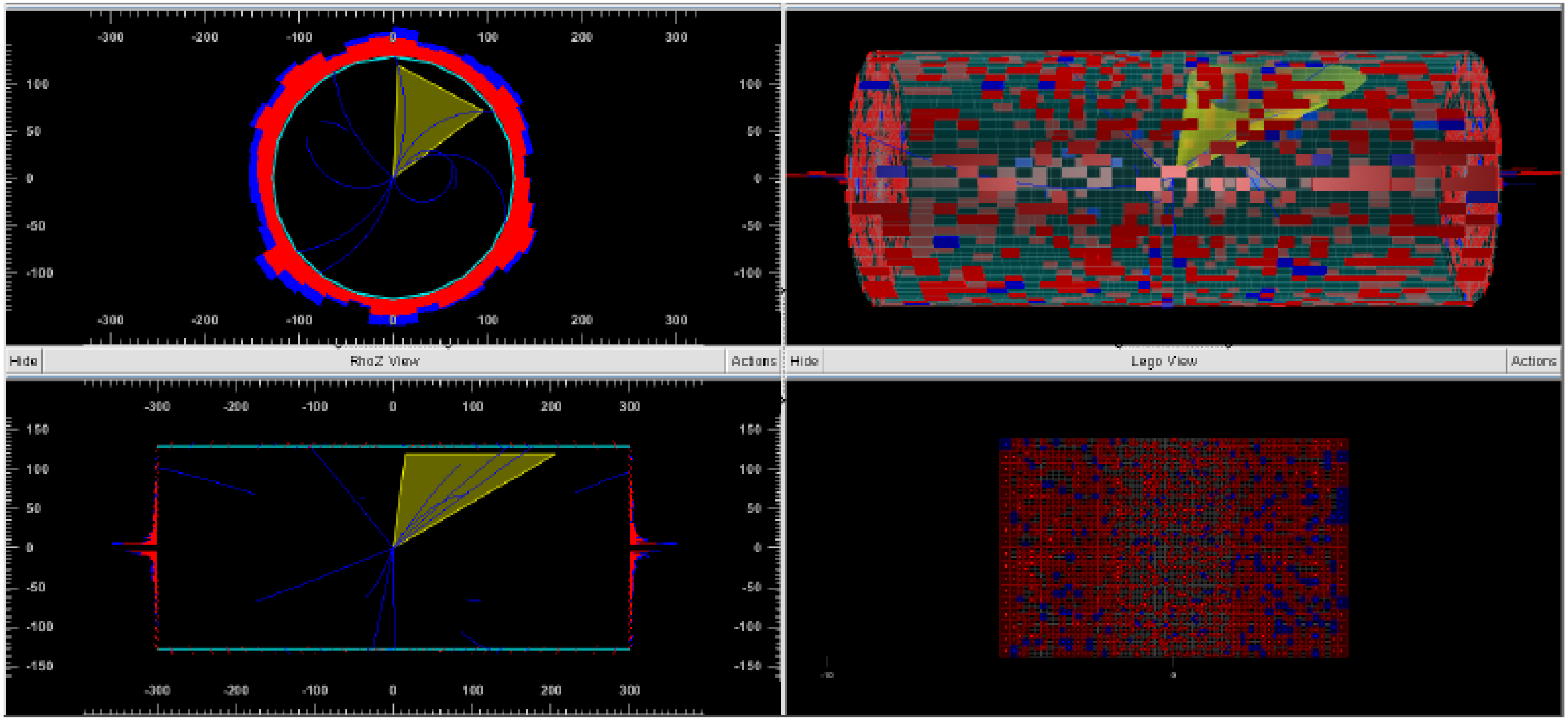
- A paper is in preparation, old paper: arXiv:0903.2225 [hep-ph]

- Test it, and give us feedback!

https://cp3.irmp.ucl.ac.be/projects/delphes

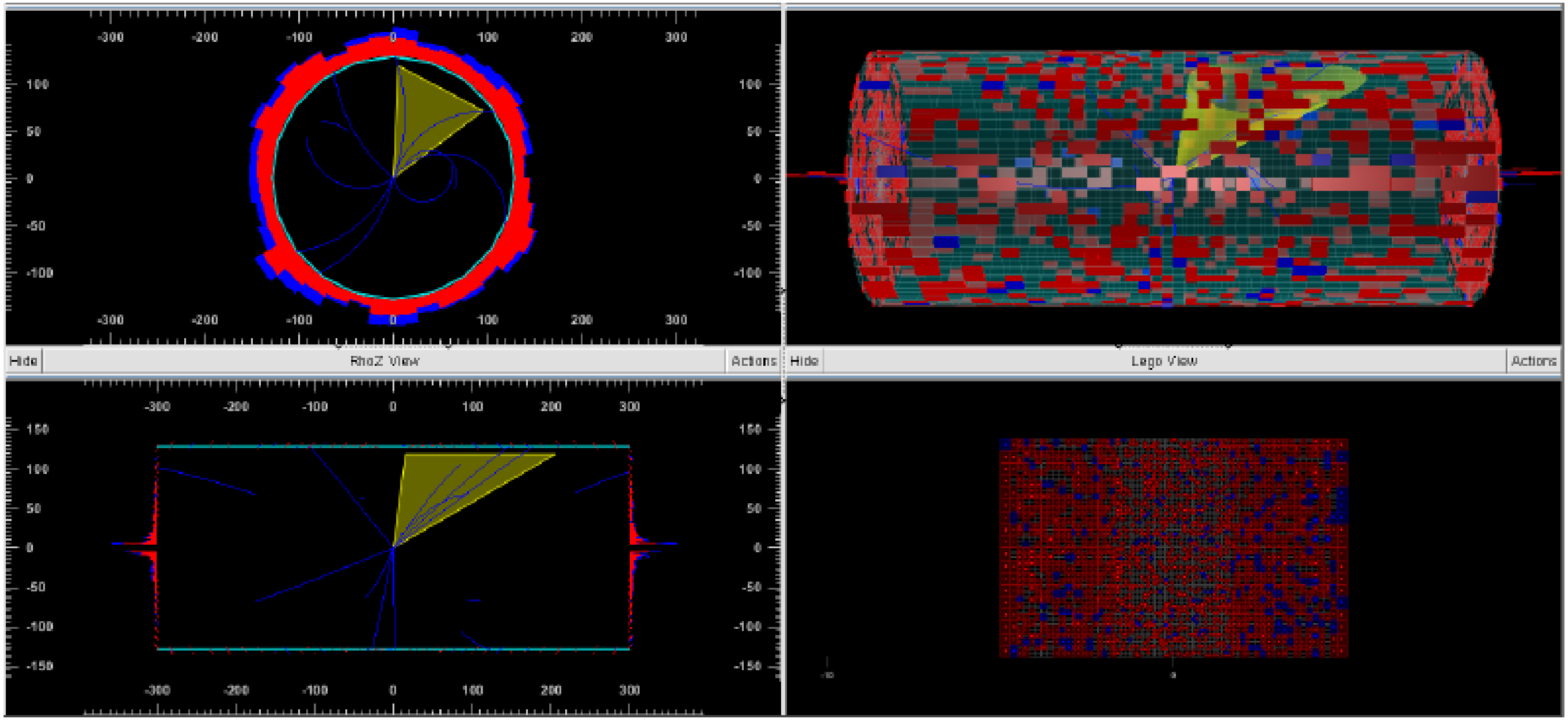
Event-display (Charged PU substr.)

$Z_{spread} \gg Z_{res}$, 50 av. pile-up



Event-display (Charged PU substr.)

$Z_{spread} \gg Z_{res}$, 50 av. pile-up

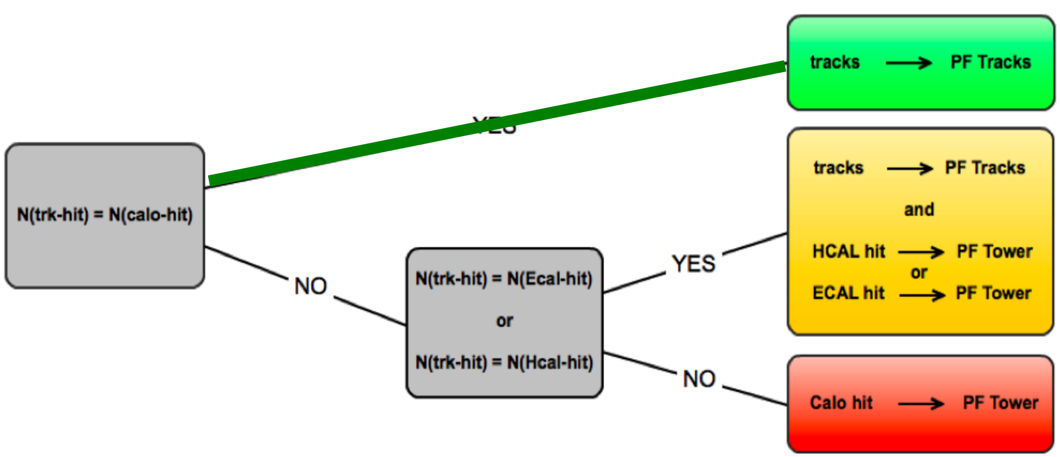


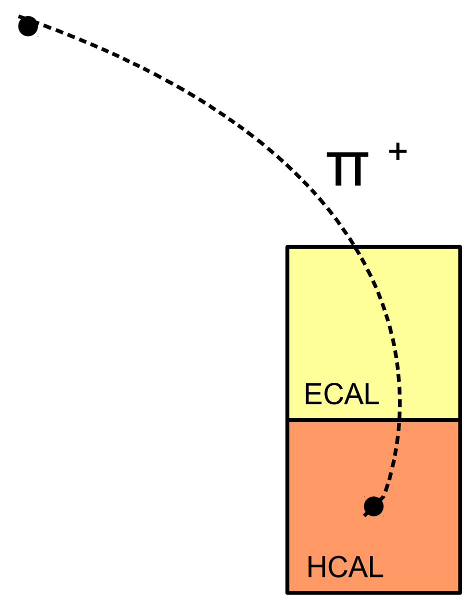
Backup slides

Particle-Flow algorithm

"Particle-flow" approach aims at **maximizing** object reconstruction **resolution** by using all sub-detector information (in Delphes tracking, ECAL and HCAL) → **particle-flow candidates** can be used as input for **jets and E_T^{miss}**

- For each calorimeter cell:

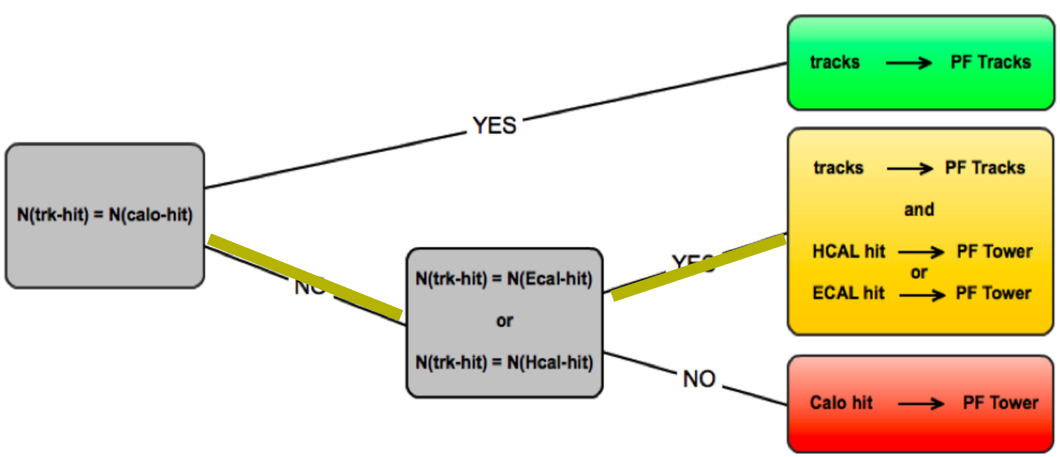


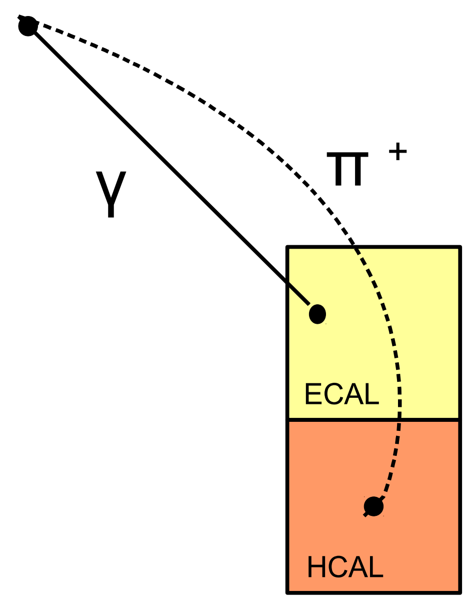


Particle-Flow algorithm

"Particle-flow" approach aims at **maximizing** object reconstruction **resolution** by using all sub-detector information (in Delphes tracking, ECAL and HCAL) → **particle-flow candidates** can be used as input for **jets and E_T^{miss}**

- For each calorimeter cell:



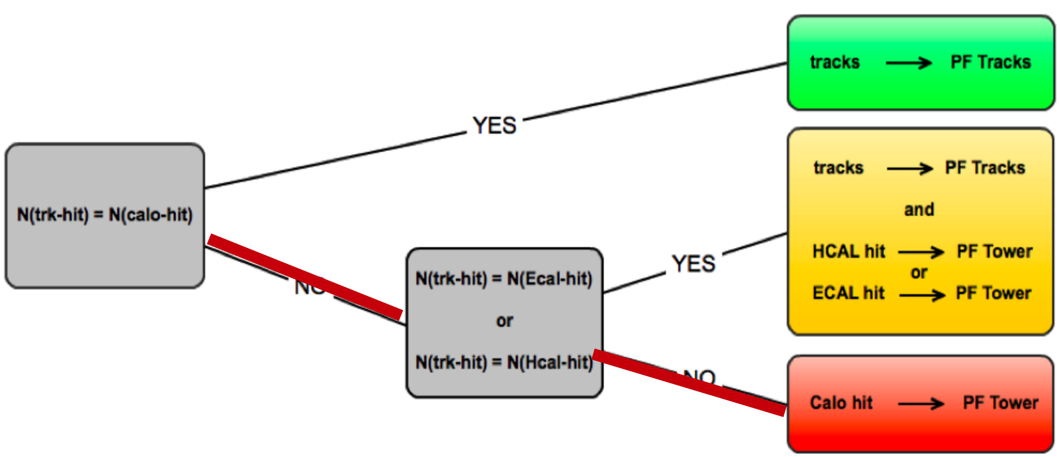


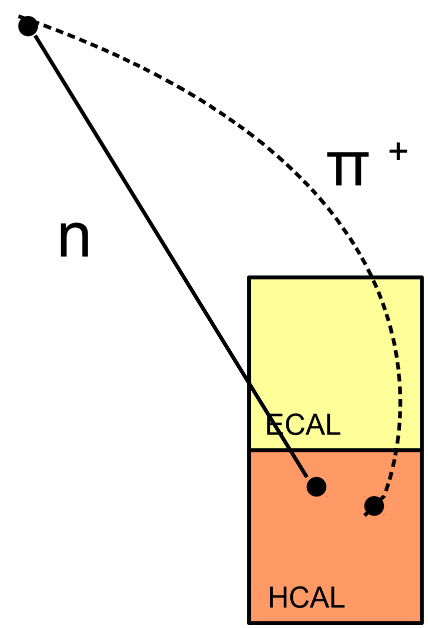
Particle-Flow algorithm

- **“Particle-flow”** approach aims at **maximizing** object reconstruction **resolution** by using all sub-detector information (in Delphes tracking, ECAL and HCAL)

  → **particle-flow candidates** can be used as input for **jets and E_T^{miss}**

- For each calorimeter cell:

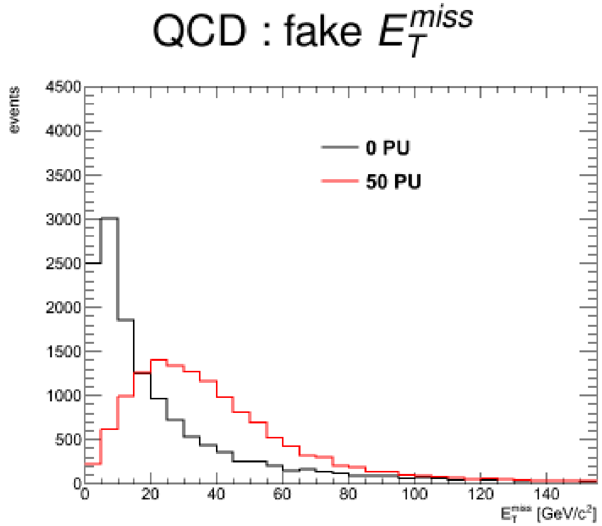


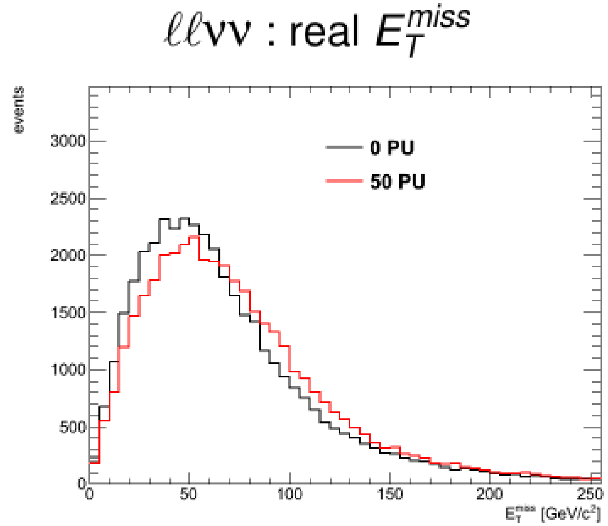


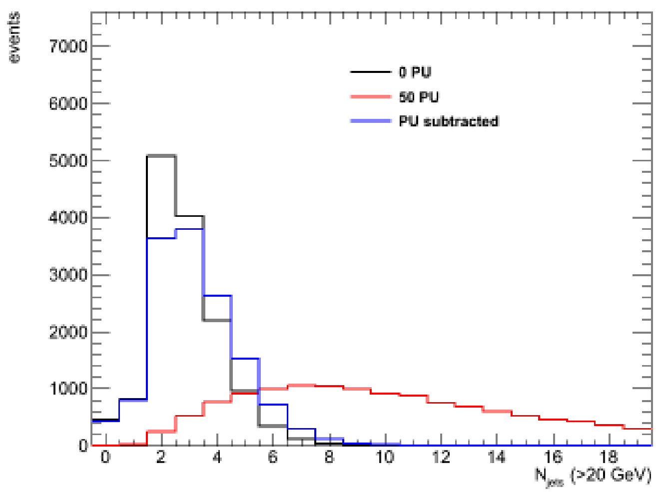
Pile-up sanity checks

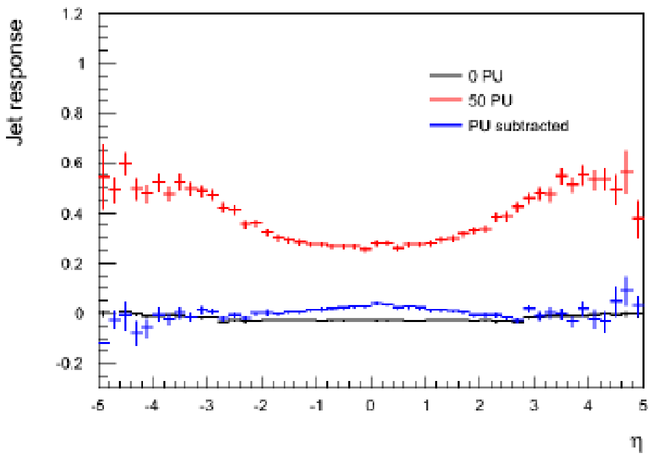
MET

Jets



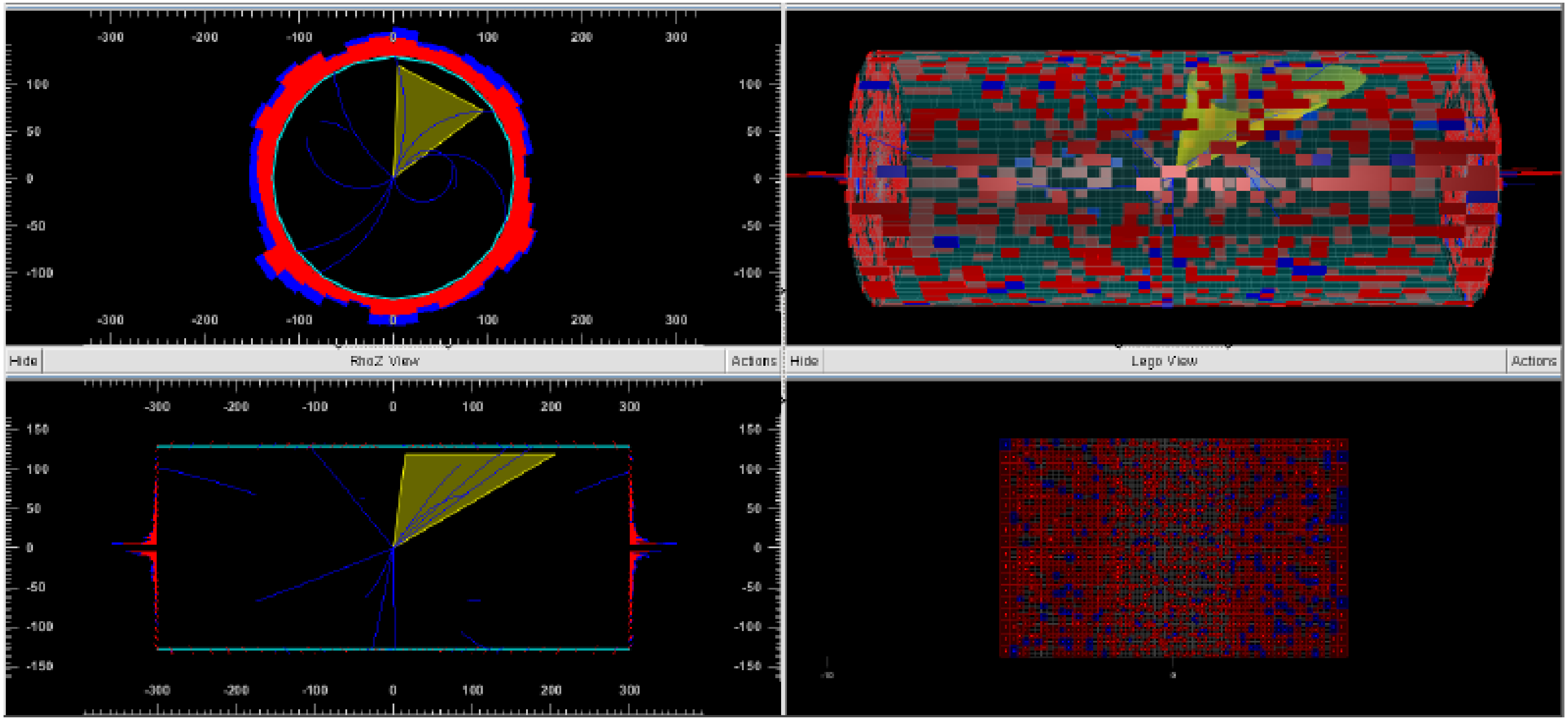






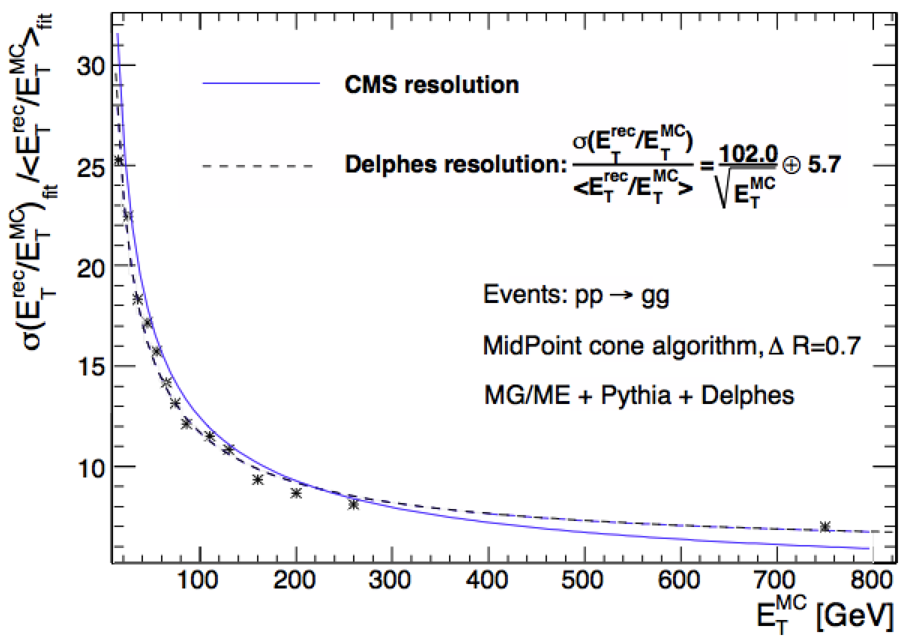
Event-display (Charged PU substr.)

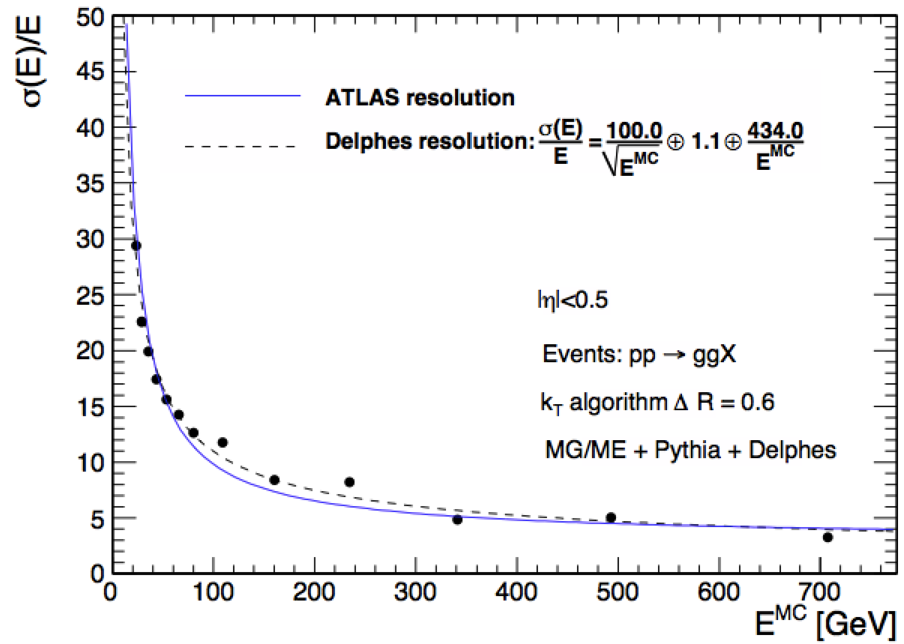
$Z_{spread} \gg Z_{res}$, 50 av. pile-up



Delphes 2 - Validation: jets

- Electrons, muons and photons are auto-validated by construction
- **Jets** and missing energy need to be tested:

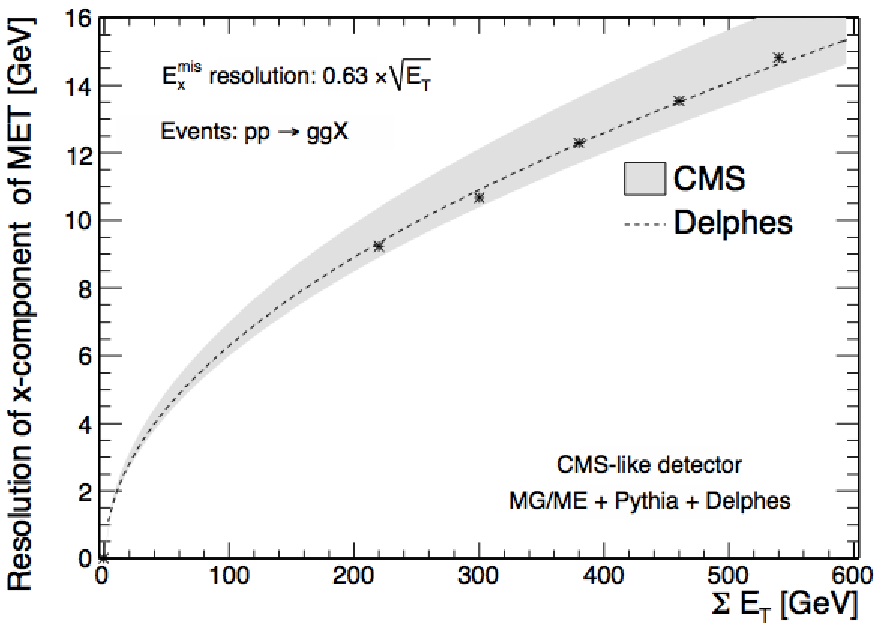


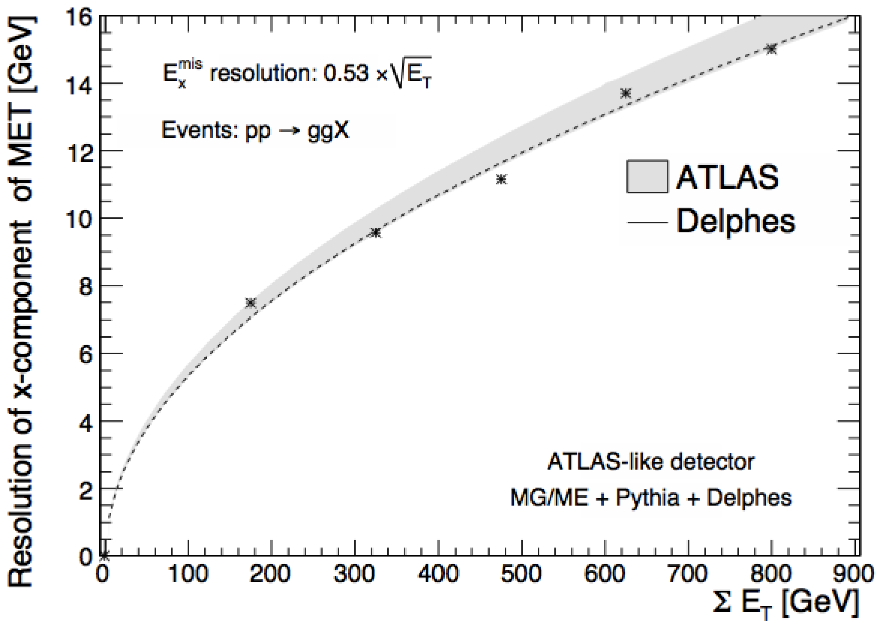


→ **good agreement**

Delphes 2 -Validation: E_T^{miss}

- Electrons, muons and photons are auto-validated by construction
- Jets and **missing energy** need to be tested:





→ **good agreement**