Fast Detector Simulation

Michele Selvaggi

Université catholique de Louvain (UCL)

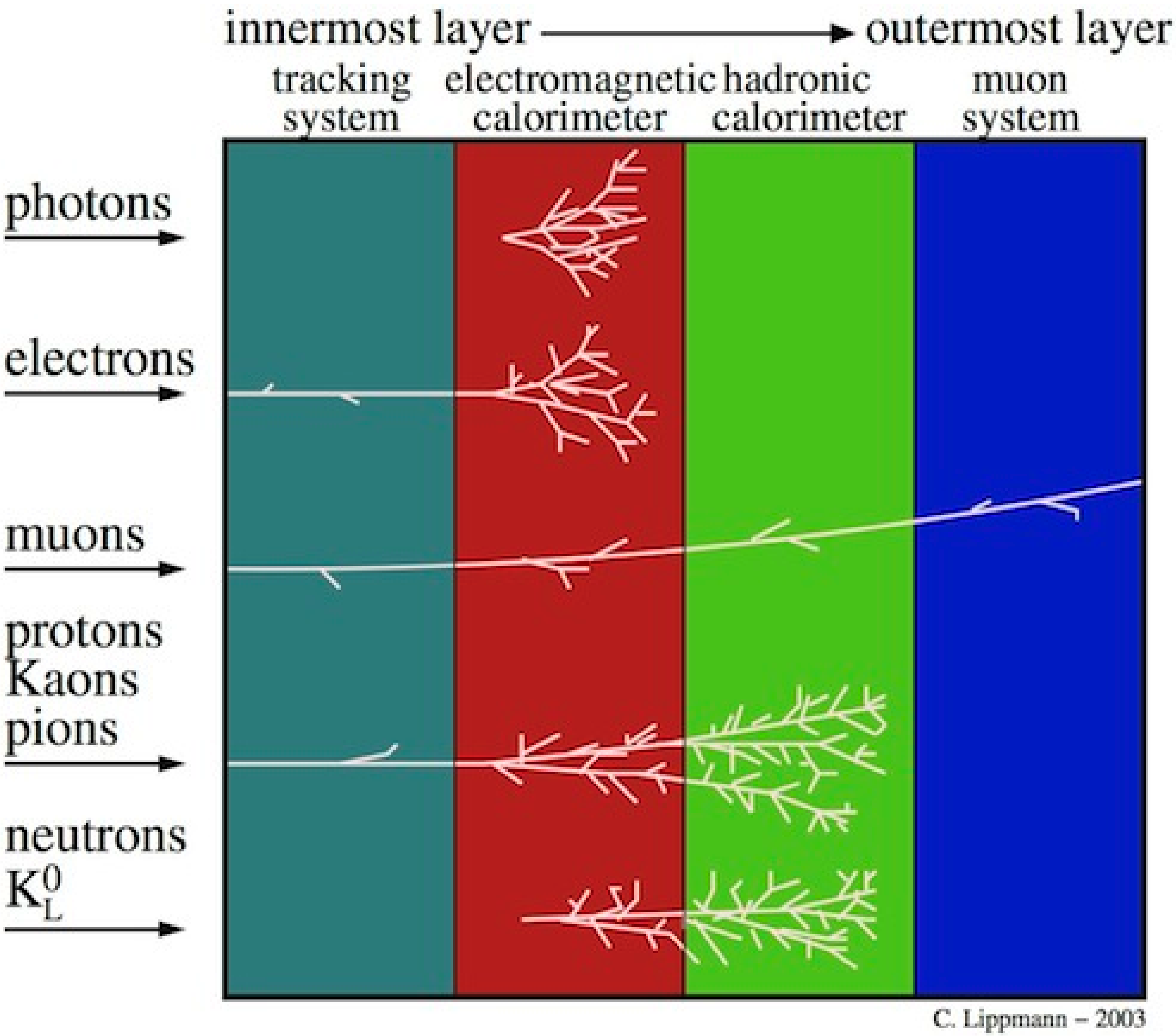
Center for Particle Physics and Phenomenology (CP3)

Shanghai Jiao Tong University
26 November 2015

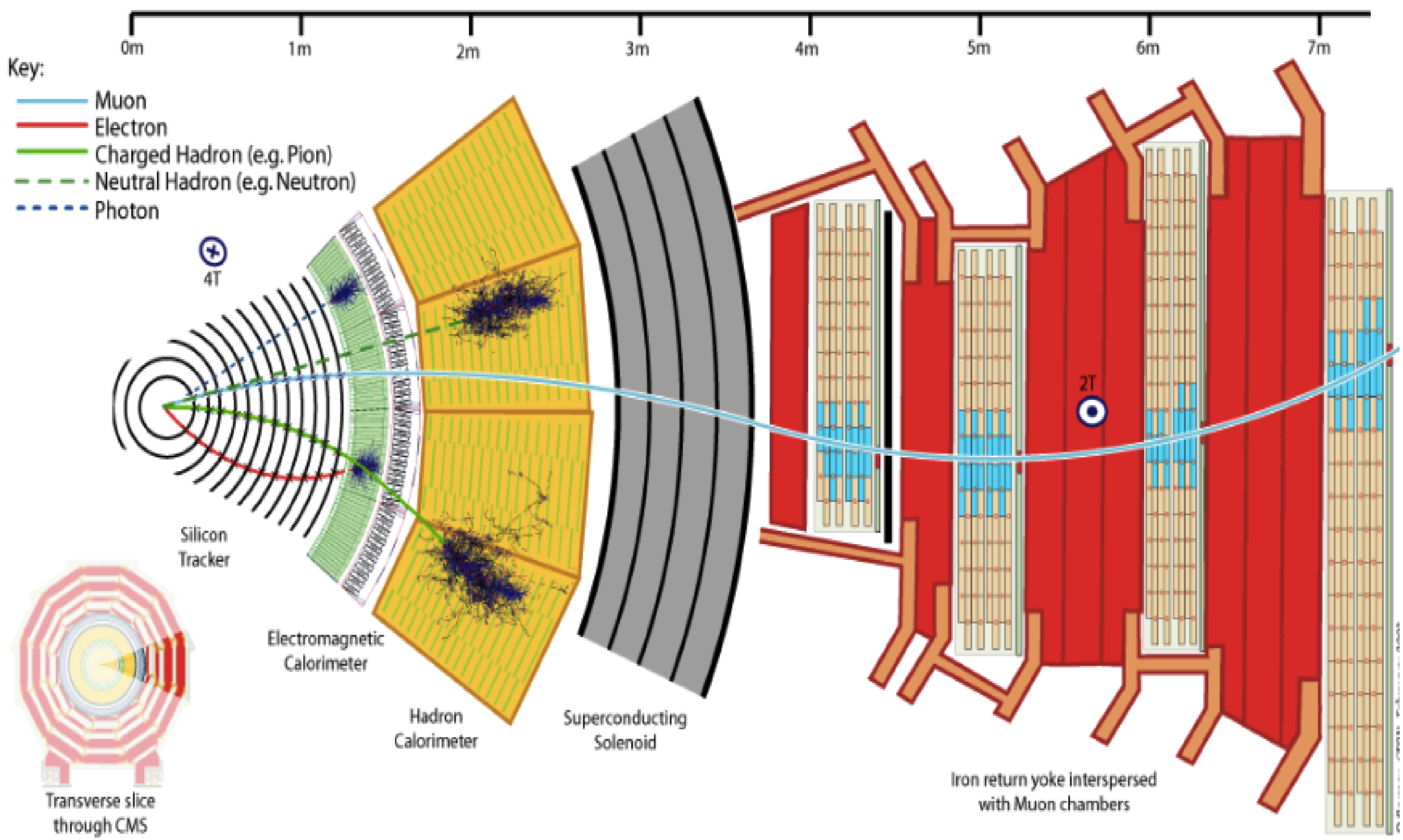
Outline

- Fast Detector simulation
- The Delphes project
- New Features
- Delphes and future colliders
- Conclusion

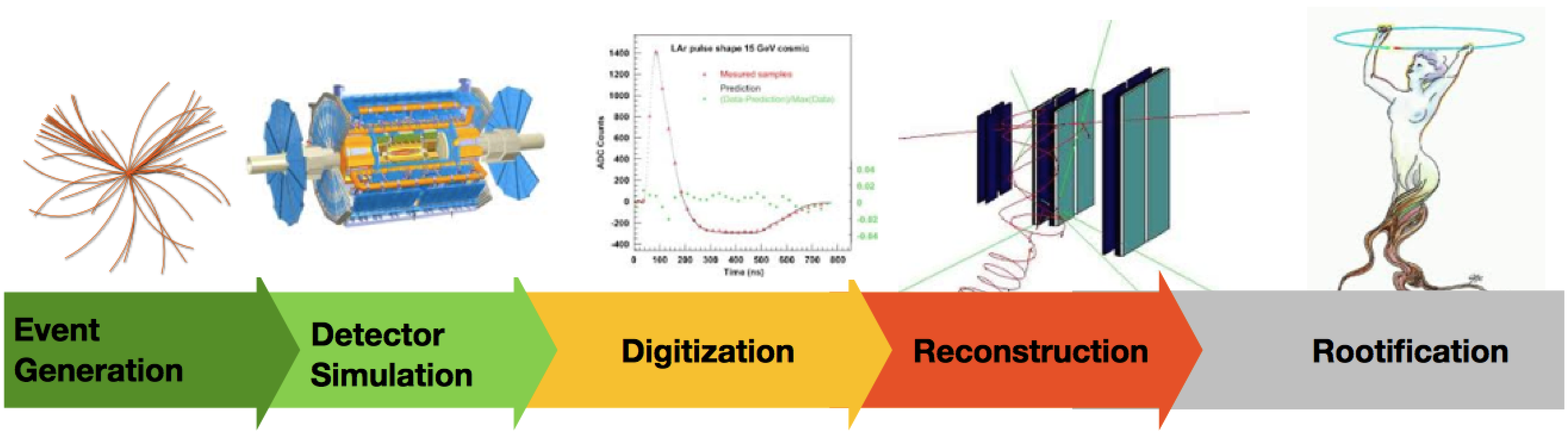
Particle Physics Detector



Particle Physics Detector



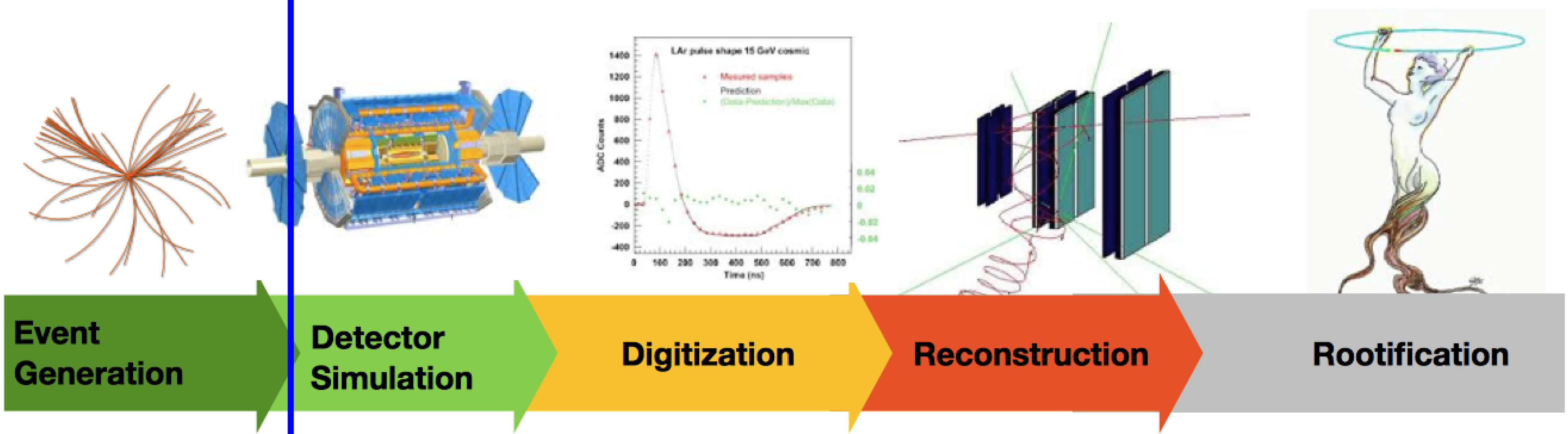
MC chain



courtesy of A. Salzburger

MC chain

FAST SIMULATION



courtesy of A. Salzburger

Detector simulation

- Full simulation (GEANT):

  - **simulates** particle-matter interaction (including e.m. showering, nuclear int., brehmstrahlung, photon conversions, etc ...) $\rightarrow$ 100 s /ev

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

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

- Parametric simulation:

  Delphes, PGS**:**

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

When and when not FastSim?

- When to use FastSim?

→ test your model with detector simulation
→ more advanced than parton-level studies
→ **sensitive to acceptance and complex observable (Jets,MET)**
→ scan big parameter space (SUSY-like)
→ preliminary tests of new geometries/resolutions (jet substructure)
→ educational purpose (bachelor/master thesis)

- When not to use FastSim?

→ high precision studies
→ very exotic topologies (heavy stable charged particles)

The Delphes Project

The Delphes project: A bit of history

DELPHES fast simulation

- Delphes project started back in 2007 at UCL as a side project to allow quick feasibility studies

- 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

- In 2013, **DELPHES 3** was released:
  - modular software
  - new features
  - also included in MadGraph suite (and interfaced with Pythia8)

- **Widely** tested and used by the community (pheno, Snowmass, CMS ECFA efforts, recasting ...)

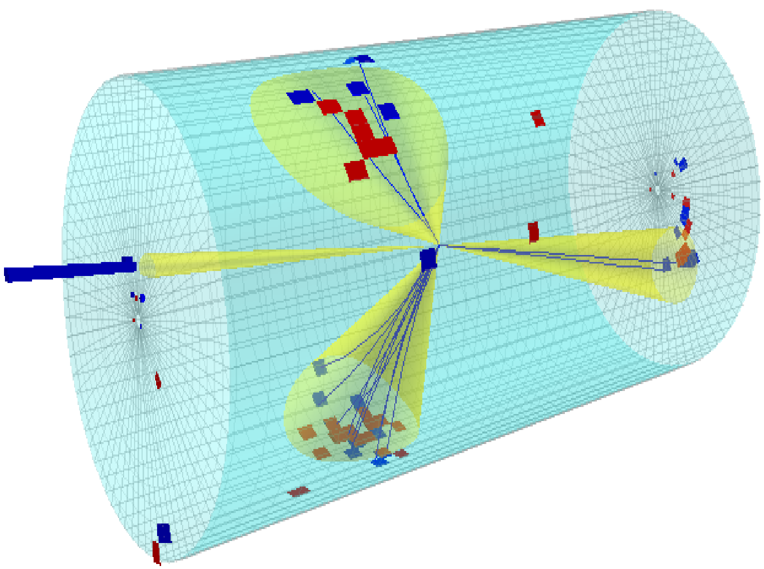
- Website and manual: https://cp3.irmp.ucl.ac.be/projects/delphes

- Paper: JHEP 02 (2014) 057

The Delphes project: Delphes in a nutshell

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

- **Includes**:
  - pile-up
  - charged particle **propagation** in magnetic field
  - electromagnetic and hadronic **calorimeters**
  - **muon** system



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

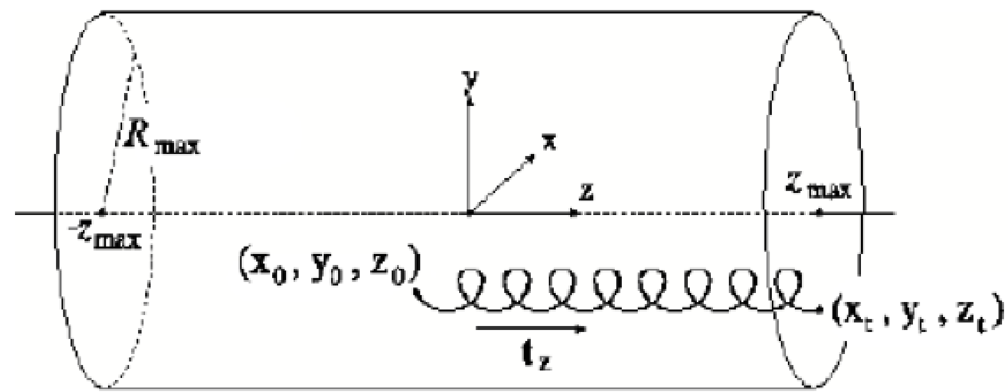
Modules Overview

The modules: Particle Propagation

- **Charged** and **neutral** 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

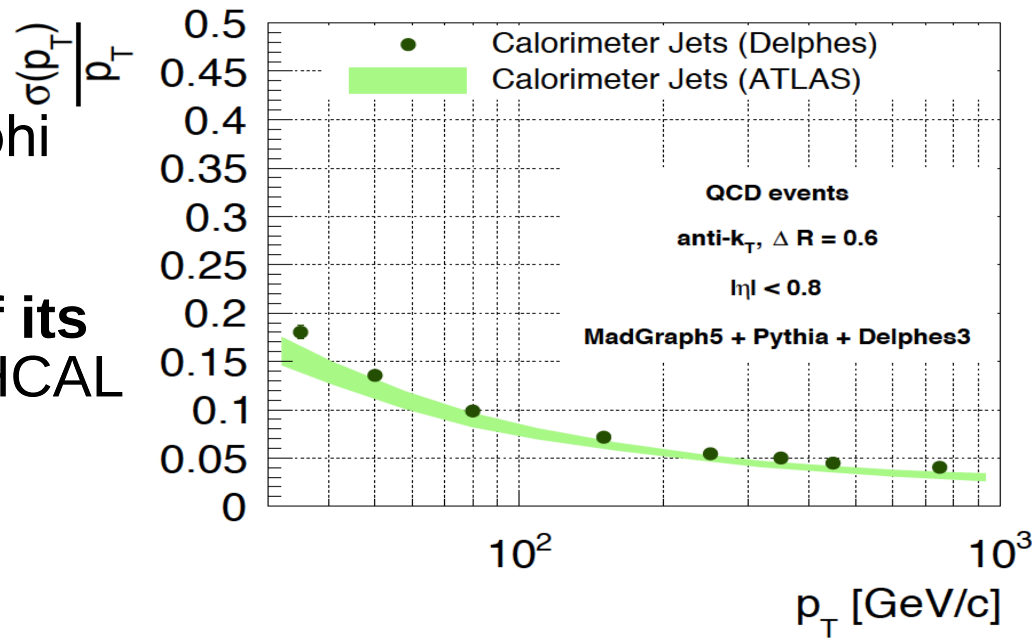
```
# efficiency formula for muons
add EfficiencyFormula {13} {                                        (pt <= 0.1)    * (0.000) + \
                                         (abs(eta) <= 1.5) * (pt > 0.1   && pt <= 1.0)    * (0.750) + \
                                         (abs(eta) <= 1.5) * (pt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

No real tracking/vertexing !!
→ no fake tracks (but can be implemented)
→ no dE/dx measurements

The modules: Calorimetry

- Can specify separate ECAL/HCAL **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

No Energy sharing between the neighboring cells
No longitudinal segmentation in the different calorimeters

The modules: Particle-Flow Emulation

- Idea: Reproduce realistically the performances of the Particle-Flow algorithm.
- In practice, in DELPHES use **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

→ If **σ(trk) < σ(calo) (low energy)**

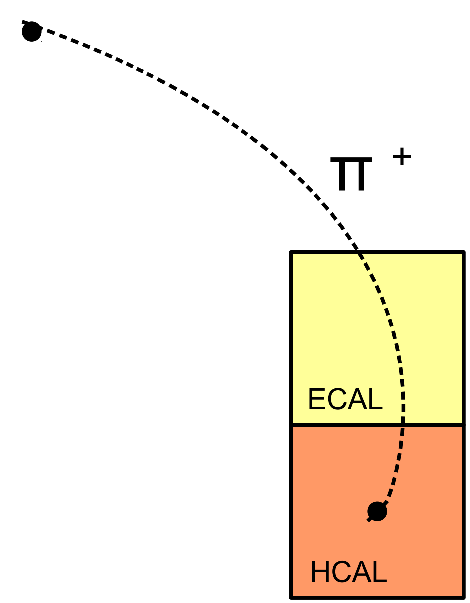
Example: A pion of 10 GeV

$E^{HCAL}(\pi^+) = 9$ GeV

$E^{TRK}(\pi^+) = 11$ GeV

Particle-Flow algorithm creates:

PF-track, with energy $E^{PF\text{-}trk} = 11$ GeV



Separate neutral and charged calo deposits has crucial implications for pile-up subtraction

The modules: Particle-Flow Emulation

- Idea: Reproduce realistically the performances of the Particle-Flow algorithm.

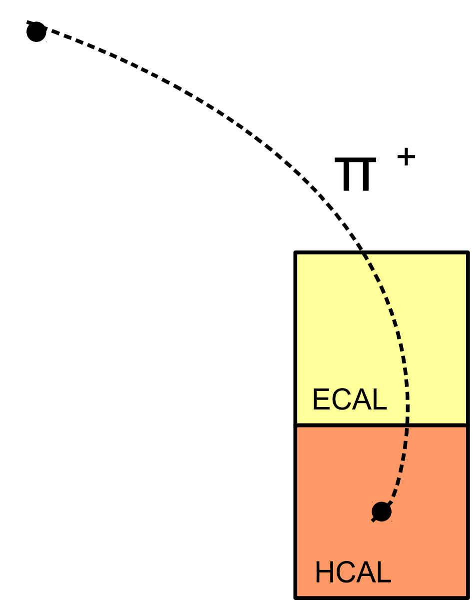
- In practice, in DELPHES use **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

→ If **σ(trk) < σ(calo) (low energy)**

Example: A pion of 10 GeV

$E^{HCAL}(\pi^+) = 15$ GeV

$E^{TRK}(\pi^+) = 11$ GeV



Particle-Flow algorithm creates:

PF-track, with energy $E^{PF\text{-}trk} = 11$ GeV
PF-tower, with energy $E^{PF\text{-}tower} = 4$ GeV

Separate neutral and charged calo deposits has crucial implications for pile-up subtraction

The modules: Particle-Flow Emulation

- Idea: Reproduce realistically the performances of the Particle-Flow algorithm.

- In practice, in DELPHES use **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

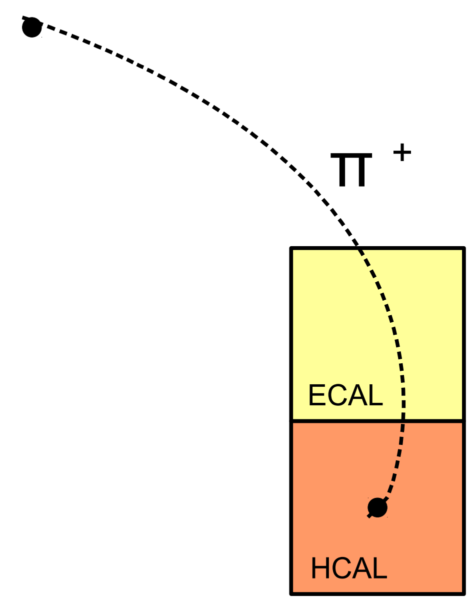
→ If **σ(trk) > σ(calo) (high energy)**

Example: A pion of 500 GeV

$E^{HCAL}(\pi^+) = 550$ GeV
$E^{TRK}(\pi^+) = 400$ GeV

Particle-Flow algorithm creates:

PF-track, with energy $E^{PF\text{-}trk} = 550$ GeV
and no PF-tower



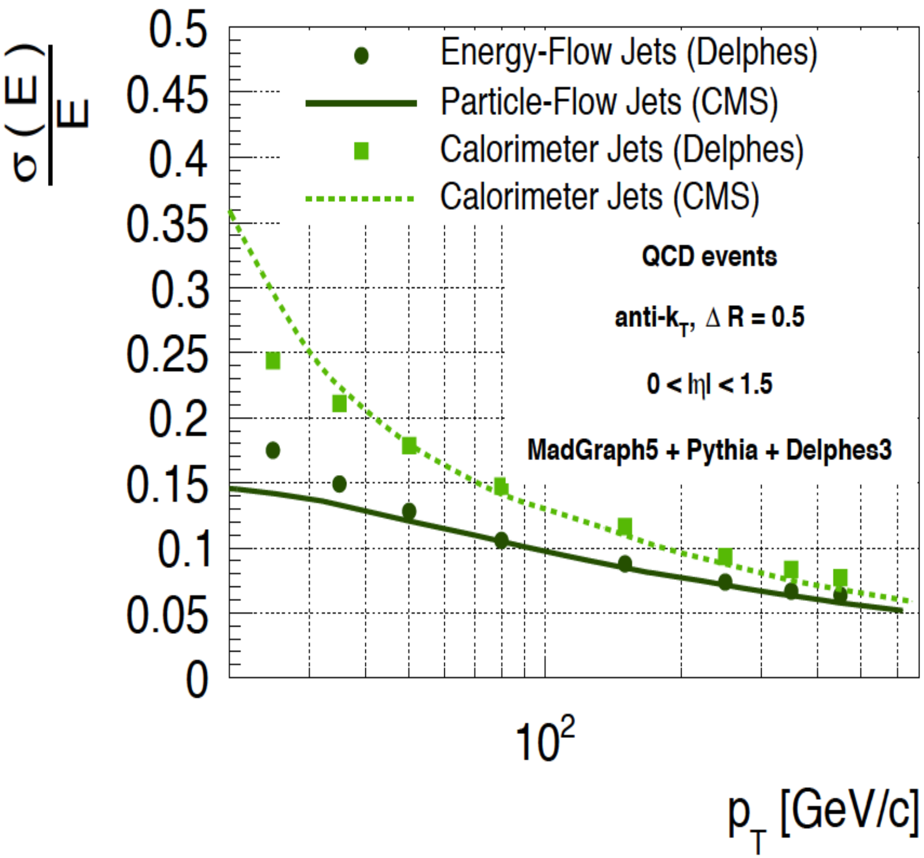
Separate neutral and charged calo deposits has crucial implications for pile-up subtraction

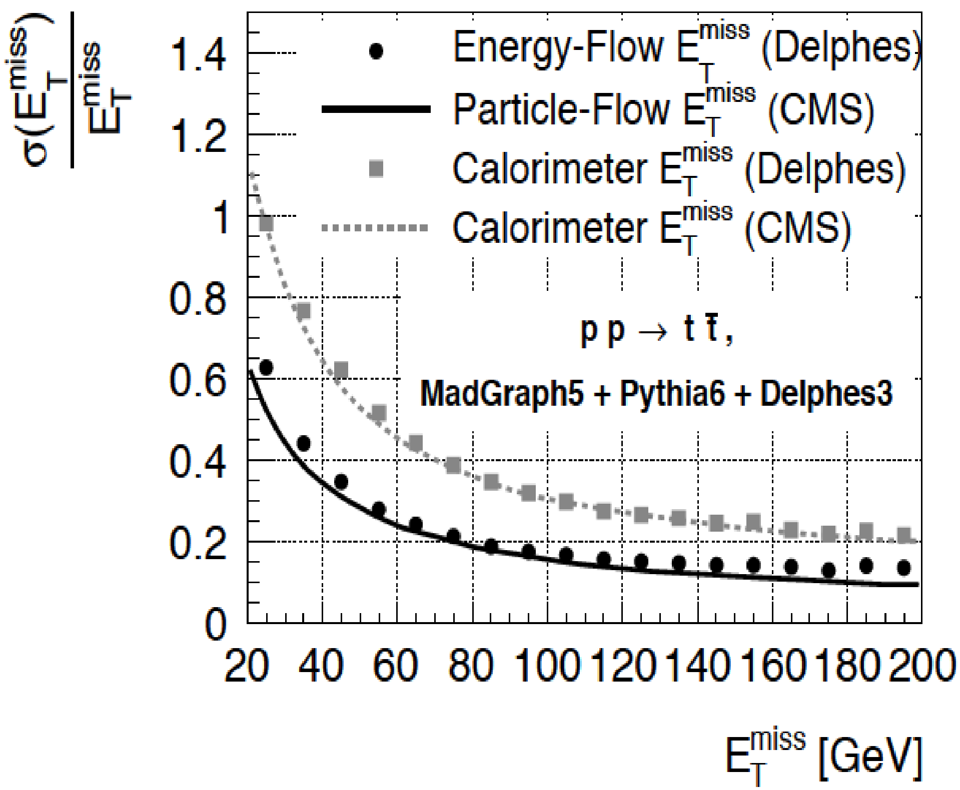
The modules: Jets / E_T^{miss} / H_T

- Delphes uses **FastJet** libraries for jet clustering
- Inputs **calorimeter towers** or "**particle-flow**" objects

```
module FastJetFinder FastJetFinder {
#  set InputArray Calorimeter/towers
  set InputArray EFlowMerger/eflow

  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt, 5 Cambridge/Aachen, 6 antikt
  set JetAlgorithm 6
  set ParameterR 0.7

  set ConeRadius 0.5
  set SeedThreshold 1.0
  set ConeAreaFraction 1.0
  set AdjacencyCut 2.0
  set OverlapThreshold 0.75

  set MaxIterations 100
  set MaxPairSize 2
  set Iratch 1

  set JetPTMin 20.0
}
```

Validation: Particle-Flow





→ **good agreement**

Leptons, photons

- Muons/photons/electrons

  - muons **identified** via their PDG id, do not deposit energy in calo (independent smearing parameterized in p_T and η)
  - electrons and photons reconstructed according to particle-flow

- Isolation:

$$I(P) = \frac{\sum_{i \neq P}^{\Delta R<R,\; p_T(i)>p_T^{min}} p_T(i)}{p_T(P)}$$

→ modular structure allows to easily define different isolation

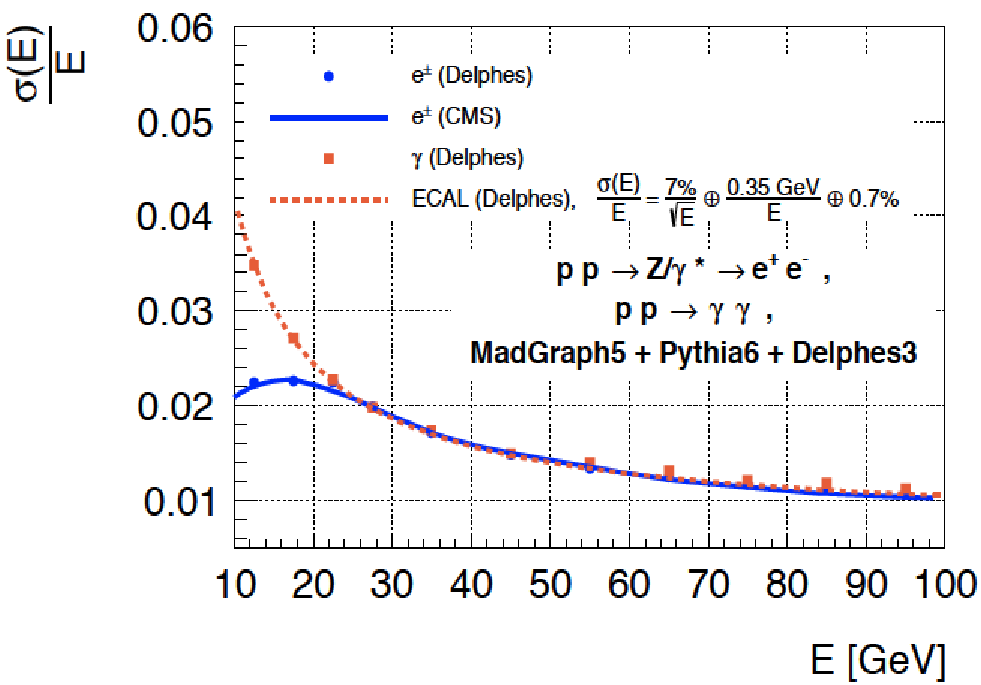
If **I(P) < Imin**, the lepton is **isolated**

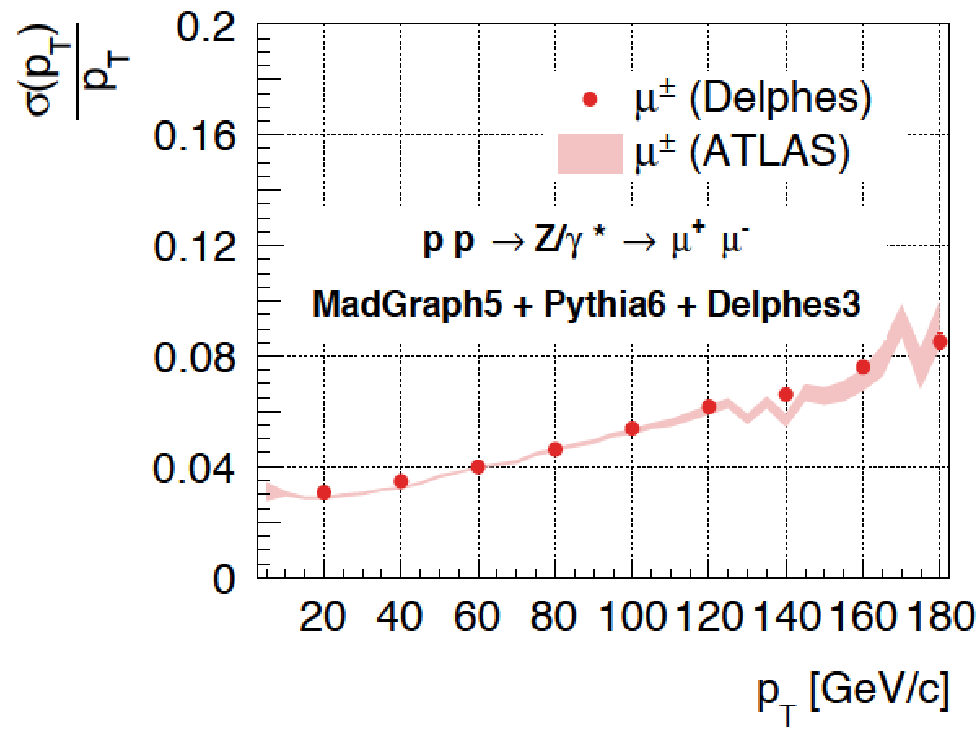
User can specify parameters I_{min}, ΔR, p_T^{min}

- Not taken into account:

  - fakes, punch-through, brehmstrahlung, conversions

Validation: electrons and muons





→ **excellent agreement**

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**

p_T and η dependent efficiency and mistag rate

VALIDATION

Physics example

- Reproduce part of **top mass measurement** in semi-leptonic decay (arXiv:1209:2319)

- **Signal** produced with MG5+Pythia+Delphes3

- Selection criteria:

→ = 1 lepton $p_T > 30$ GeV, $|\eta| < 2.1$
→ ≥ 4 jets $p_T > 30$ GeV, $|\eta| < 2.4$
→ ≥ 2 b-tagged jets, ≥ 2 light jets

eff(Delphes) = 2.8% vs. eff(CMS) = 2.3%

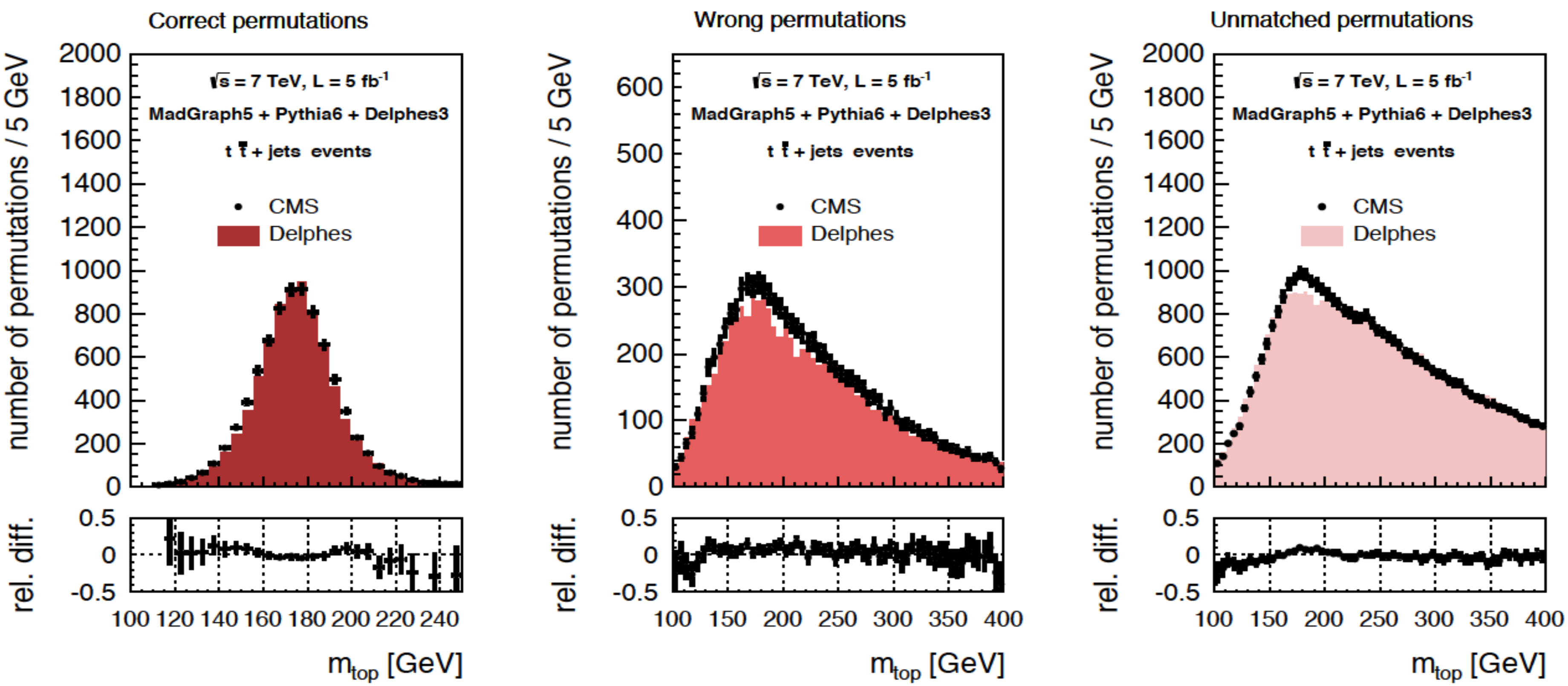
→ **good agreement**

Physics example

Look at **hardest** 2 b-tagged and 2 light jets (à la CMS):

- correct : 4 jets are good, match right b with lights
- wrong : 4 jets are good, match wrong b with lights
- unmatched : at least one of the jets don't match

	CMS	DELPHES
correct	15.5 %	15.8 %
wrong	17.4 %	16.5 %
unmatched	67.1 %	67.7 %

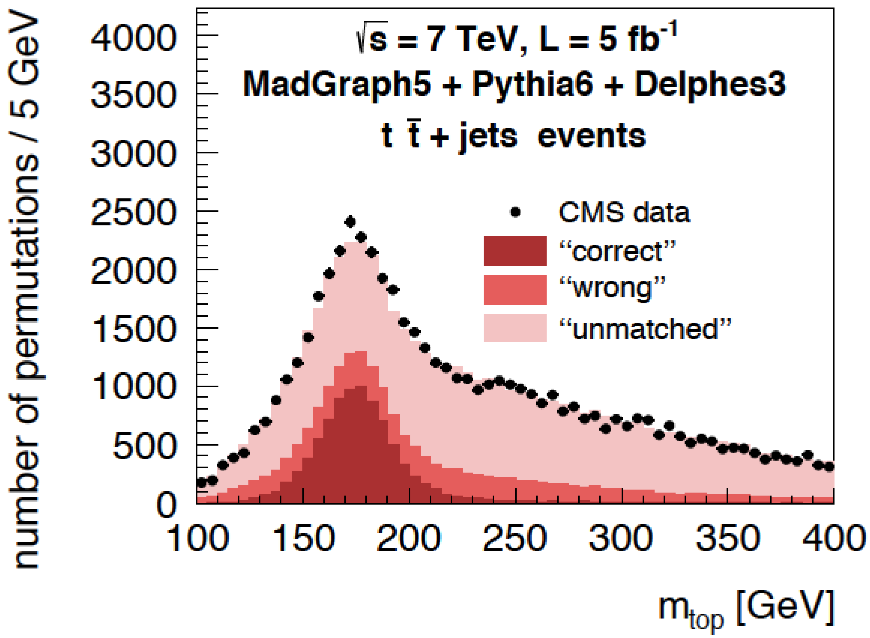


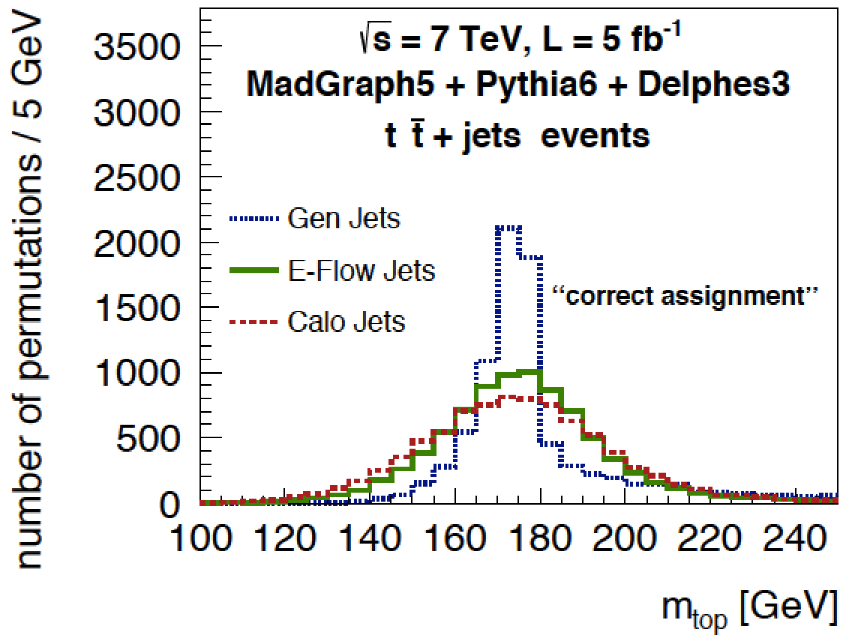
Physics example

Look at **hardest** 2 b-tagged and 2 light jets (à la CMS):

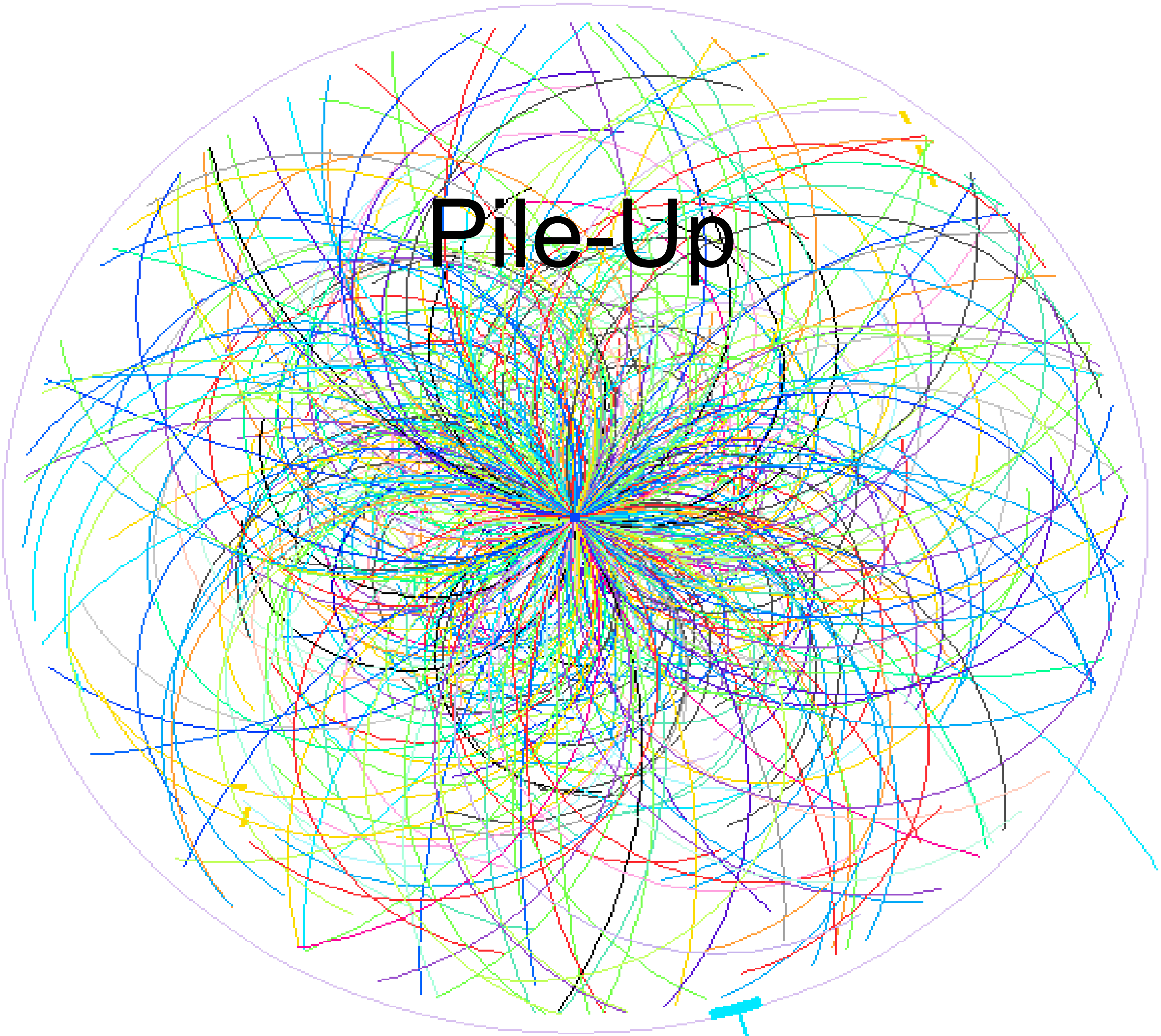
- correct : 4 jets are good, match right b with lights
- wrong : 4 jets are good, match wrong b with lights
- unmatched : at least one of the jets don't match

	CMS	DELPHES
correct	15.5 %	15.8 %
wrong	17.4 %	16.5 %
unmatched	67.1 %	67.7 %





Pile-Up



Pile-Up

- **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
  - rotate event by random angle φ wrt z-axis

- **Charged** Pile-up subtraction (most effective if used with PF algo)

  - 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)

  - allows user to tune amount of charged particle subtraction by **adjusting Z spread/resolution**

- **Residual** eta dependent pile-up substraction is needed for jets and isolation.
  - Use the FastJet Area approach (Cacciari, Salam, Soyez)
    - compute ρ = event pile-up density
    - jet correction : $p_T \rightarrow p_T - \rho A$ (JetPileUpSubtractor)
    - isolation : $\sum p_T \rightarrow \sum p_T - \rho \pi R^2$ (Isolation module itself)

Pile-Up

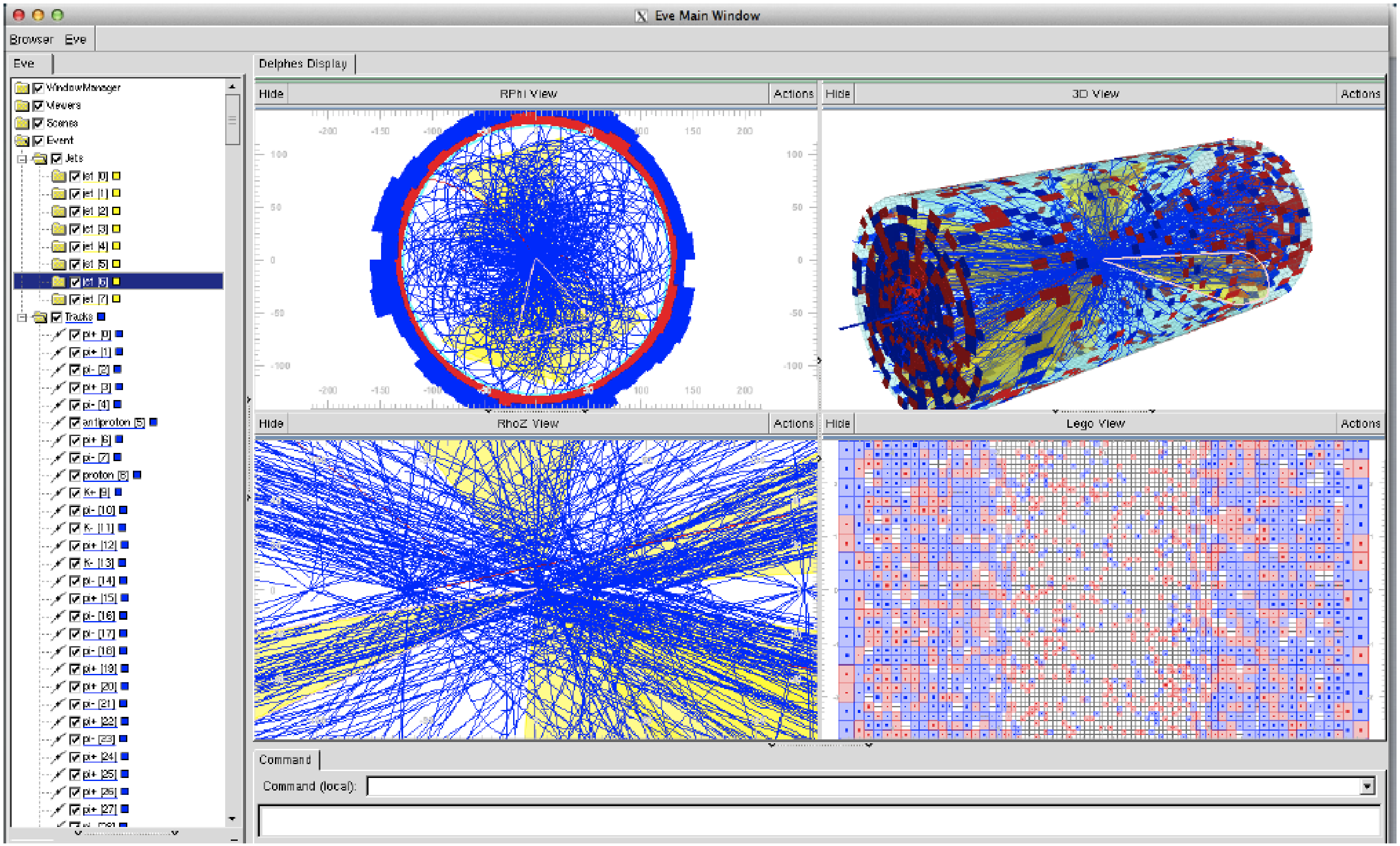
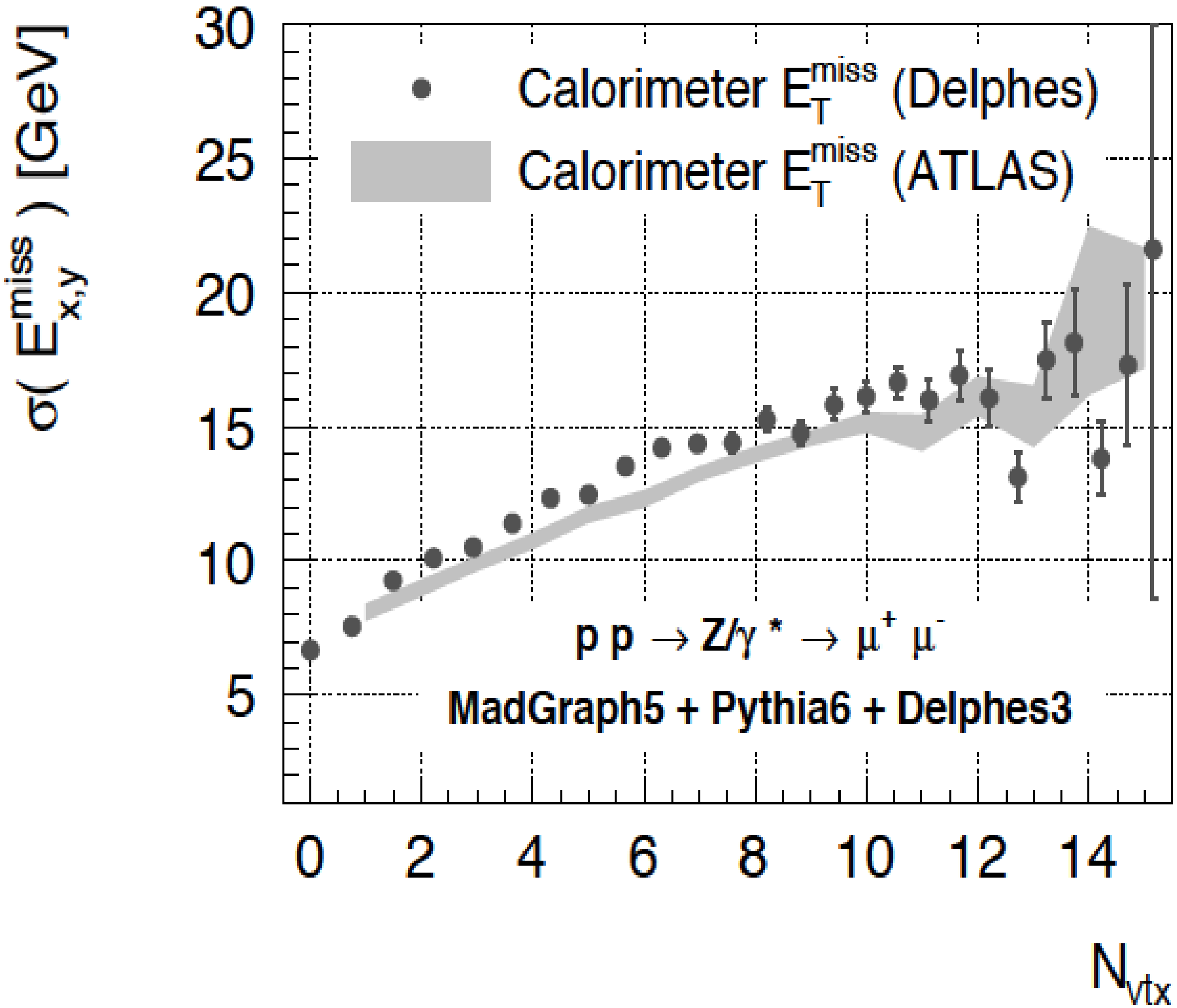


Figure 3. QCD event with 50 pile-up interactions shown with the DELPHES event display based on the ROOTEVE libraries [12]. Transverse view (top left), longitudinal view (bottom left), 3D view (top right), (η,ϕ) view (bottom right).

Validation: Pile-Up



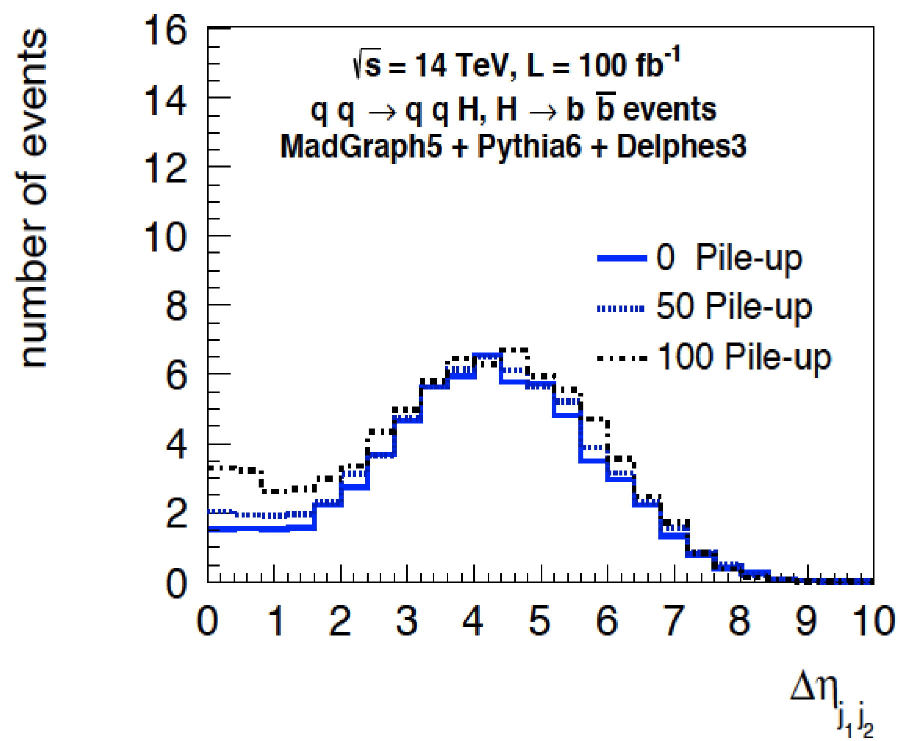
→ **good agreement**

Validation: Pile-Up

- H → bb in **VBF channel** expected to be highly affected by pile-up
- Irreducible background **bb+jets**
- Select >4 jets with pT > 80, 60, 40, 40 (at least 2 b-tagged, at least 2 light)

Emergence of pile-up jets in the central region:

→ **depletion of rapidity gap**



New Features

b-tagging

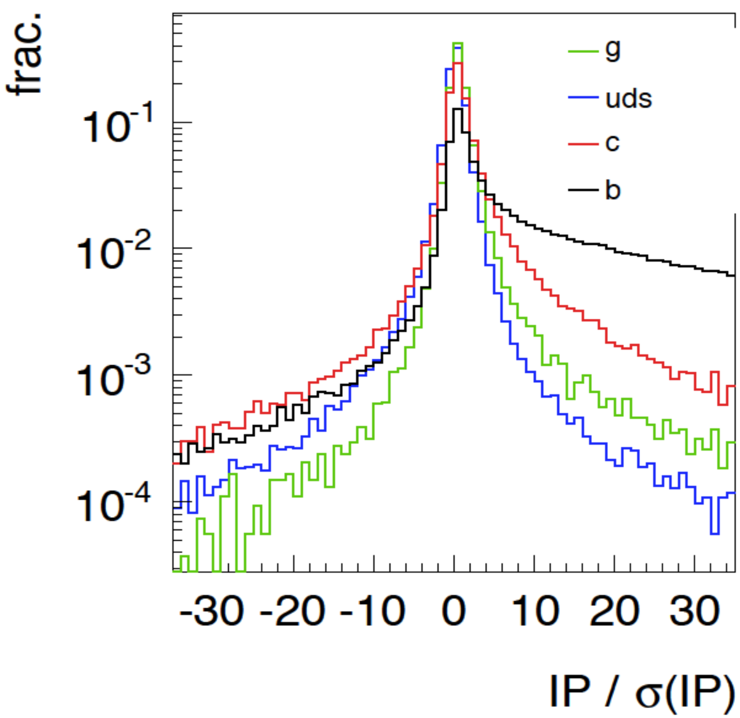
Parametrized **b-tagging**:
- Check if there is a b,c-quark in the cone of size DeltaR
- Apply a **parametrized Efficiency** (PT, eta)

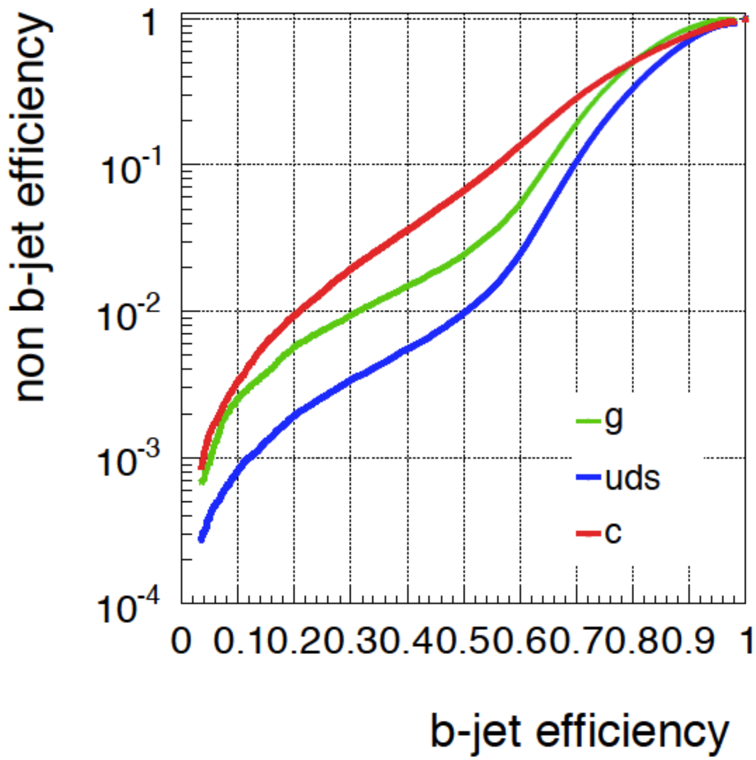
```
module BTagging BTagging {
  set PartonInputArray Delphes/partons
  set JetInputArray JetEnergyScale/jets

  set BitNumber 0
  set DeltaR 0.5
  set PartonPTMin 1.0
  set PartonEtaMax 2.5

  # default efficiency formula (misidentification rate)
  add EfficiencyFormula {0} {0.001}

  # efficiency formula for c-jets (misidentification rate)
  add EfficiencyFormula {4} {                                  (pt <= 15.0) * (0.000) + \
                                               (abs(eta) <= 1.2) * (pt > 15.0) * (0.2*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 15.0) * (0.1*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 2.5)                                  * (0.000)}

  # efficiency formula for b-jets
  add EfficiencyFormula {5} {                                  (pt <= 15.0) * (0.000) + \
                                               (abs(eta) <= 1.2) * (pt > 15.0) * (0.5*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 15.0) * (0.4*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 2.5)                                  * (0.000)}
}
```

→ perfectly reproduces existing performances
→ not predictive

Track counting b-tagging

- Track parameters (p_T, d_{XY}, d_Z) derived from **track fitting** in real experiments
- In Delphes we can **smear** directly $\mathbf{d_{XY}}$, $\mathbf{d_Z}$ according to **($\mathbf{p_T}$, η)** of the track
- **Count tracks** within jet with **large impact parameter** significance.





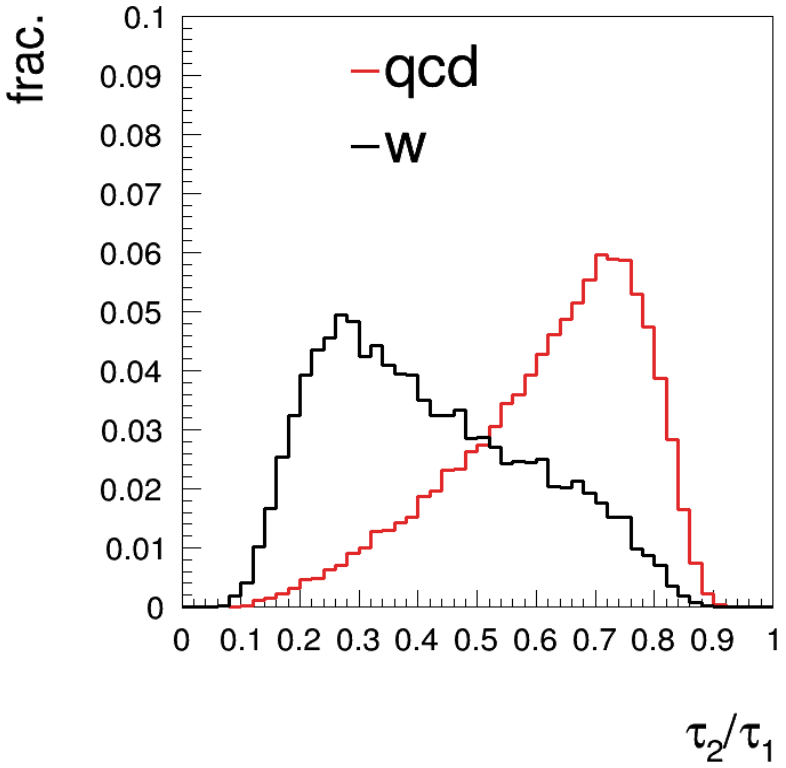
→ although very simple is predictive

→ ignore correlations among track parameters

Jet Substructure

JHEP 1103:015 (2011), JHEP 1202:093 (2012) and JHEP 1404:017 (2014)

- Embedded in FastJetFinder module
- $\tau_1, \tau_2, .. , \tau_5$ saved as jet members (N-subjettiness)
- Trimming, Pruning, SoftDrop ...

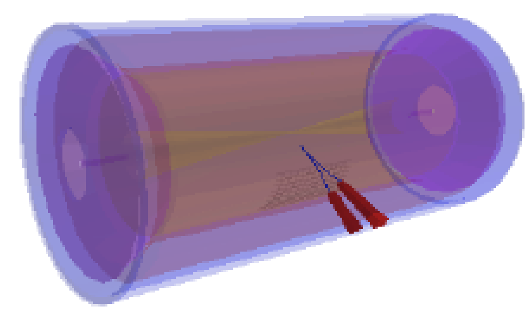



```
###########
# Jet finder
###########

module FastJetFinder FastJetFinder {
#  set InputArray Calorimeter/towers
  set InputArray EFlowMerger/eflow

  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt, 5 Cambridge/Aachen, 6 antikt
  set JetAlgorithm 5
  set ParameterR 1.0

  set JetPTMin 200.0

  set ComputeTrimming true
  set ComputePruning true
  set ComputeSoftDrop true
  set ComputeNsubjettiness true

}
```

Photon Conversions

- probability of converting after distance "Δx"

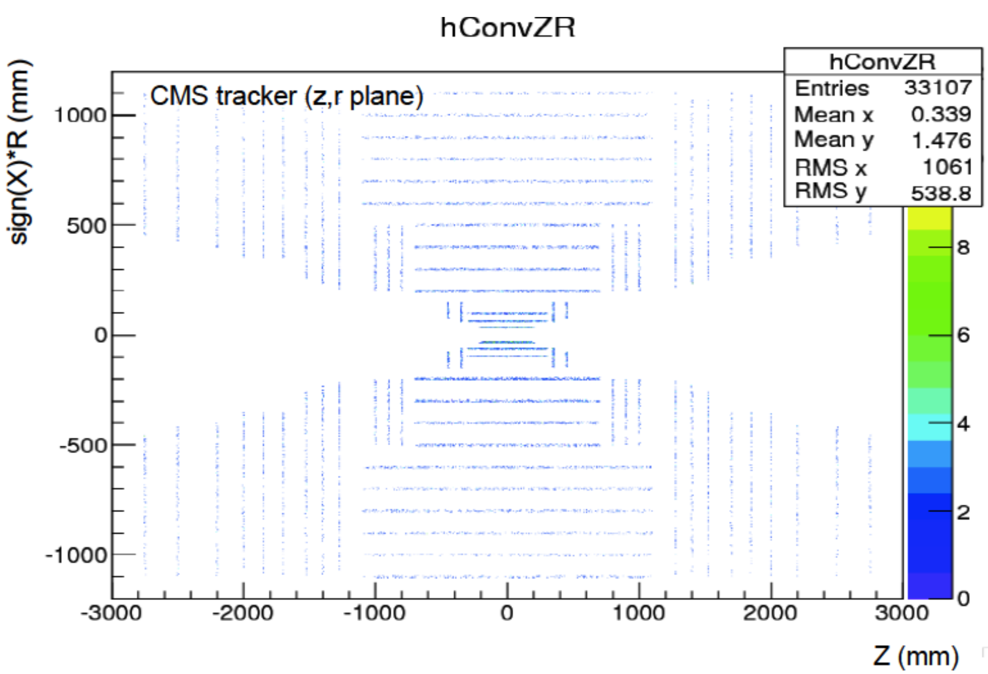
$$P(\text{conv. after } \Delta x) = 1 - \exp(-\Delta x / \lambda)$$



1) material budget map can be provided via

$\lambda^{-1}(r, z, phi)$ = average conversion rate per unit length (m^{-1})

$= 7/9 * \varsigma / X_0$



2) step length "Δx"

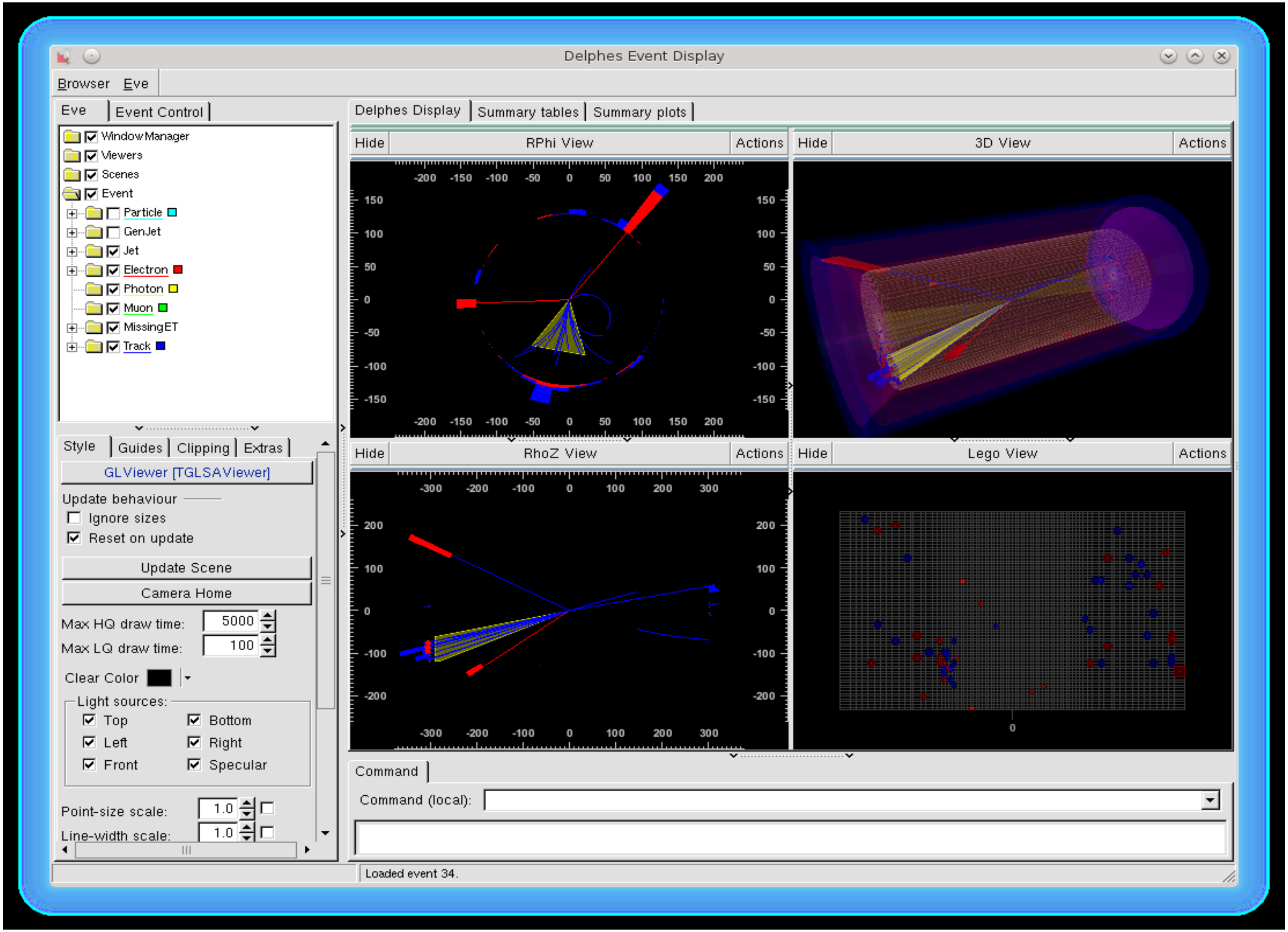
3) the photon annihilation cross-section

$$d\sigma / dx \sim 1 - 4/3\, x(1 - x)$$

More info:
https://cp3.irmp.ucl.ac.be/projects/delphes/raw-attachment/wiki/WorkBook/Modules/delphes_conversions.pdf

Event Display

Useful for educational and debugging purposes ...



Delphes and Future colliders

Delphes and future colliders

- Delphes has been designed to deal with **high number of hadrons** environment**:**
  - Jets, MET and object isolation are modeled realistically
  - pile-up subtraction (FastJet Area method, Charged Hadron Subtraction)
  - pile-up JetId

- Recent improvements
  - **different segmentation** for ECAL and HCAL
  - Impact parameter smearing: allow for **predictive b-tagging** (now parametrized)
  - **jet substructure** for boosted objects (N-(sub)jettiness)
  - Included configuration card for future collider studies
  - Embed Pythia8 parton shower inside Delphes simulation

Delphes and future colliders

Delphes can be used **right-away** for **future colliders** studies ...

What can you do with Delphes?

- reverse engineering

  → you have some target for jet invariant mass resolution
  what granularity and resolution are needed to achieve it?

- impact of pile-up on isolation, jet structure, multiplicities ...

In which context?

- **preliminary physics studies** can be performed in **short time**
- can be used **in parallel** with full detector simulation
- flexible software structure allows **integration** in other frameworks
  (can be called from others programs, see manual)

Run delphes ...

Running Delphes

- Install ROOT (and load environment):

```
source [path-to-root-installation]/bin/thisroot.sh
```

- Download, unpack and install latest Delphes version

```
wget http://cp3.irmp.ucl.ac.be/downloads/Delphes-3.3.1.tar.gz
tar xzvf Delphes-3.3.1.tar.gz
cd Delphes-3.3.1
make -j 4
```

- To run you need an hadron-level input file (produced by MG+Py/Herwig). Delphes accepts both *.hep or *.hepmc format.

  You can download a small example sample from here (or generate one):

```
wget http://cp3.irmp.ucl.ac.be/downloads/z_ee.hep.gz
gunzip z_ee.hep.gz
```

- And run with the default CMS detector card:

```
./DelphesSTDHEP cards/delphes_card_CMS.tcl delphes_output.root z_ee.hep
```

- Follow README file for a quick start tutorial, starting from section “Simple analysis [...]”

Running Delphes

- Install ROOT (and load environment):

```
source [path-to-root-installation]/bin/thisroot.sh
```

- Download, unpack and install latest Delphes version

```
wget http://cp3.irmp.ucl.ac.be/downloads/Delphes-3.3.1.tar.gz
tar xzvf Delphes-3.3.1.tar.gz
cd Delphes-3.3.1
make -j 4
```

- To run you need an hadron-level input file (produced by MG+Py/Herwig). Delphes accepts both *.hep or *.hepmc format.

  You can download a small example sample from here (or generate one):

```
wget http://cp3.irmp.ucl.ac.be/downloads/z_ee.hep.gz
gunzip z_ee.hep.gz
```

- And run with the default CMS detector card:

```
./DelphesSTDHEP cards/delphes_card_CMS.tcl output.root z_ee.hep
```

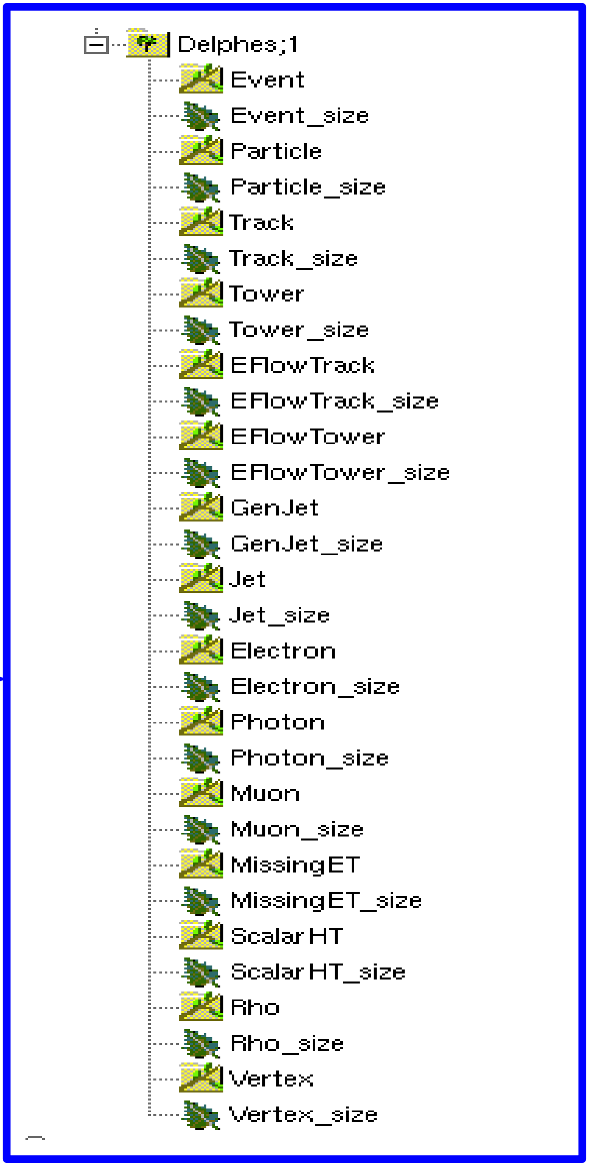
cards/delphes_card_CMS.tcl → detector card

output.root → output file

z_ee.hep → input event file

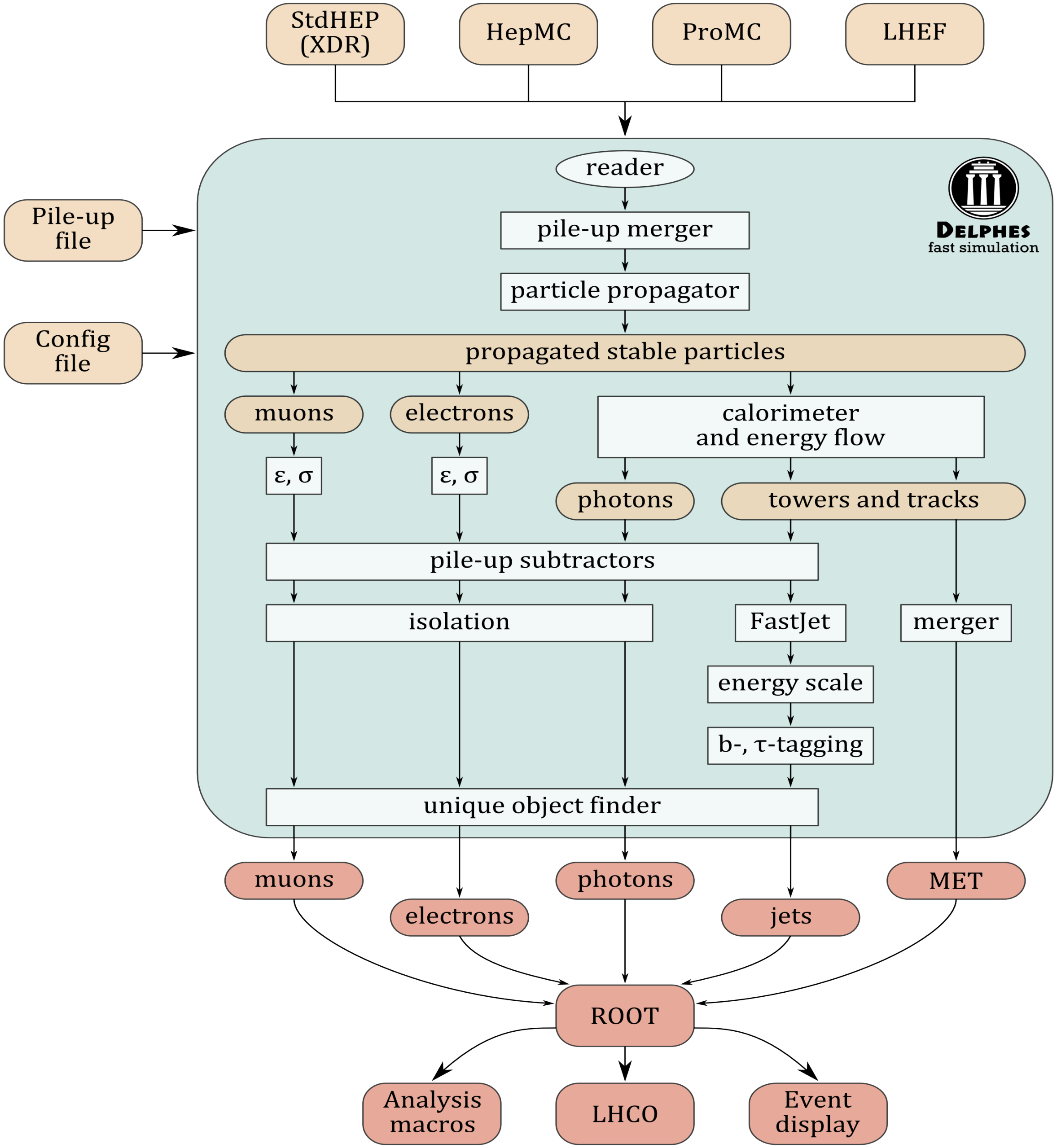
Technical features

- **modular** C++ code, uses ROOT classes
- Input
  - Pythia/Herwig output (HepMC,STDHEP)
  - LHE (MadGraph/MadEvent)
  - ProMC
- Output
  - ROOT trees
- Configuration file
  - define geometry
  - resolution/reconstruction/selection criteria
  - output object collections



default **CMS/ATLAS** configurations and Future Colliders (**FCC-hh, ILC**) are included in any Delphes release

Modularity in action



Conclusions

- **Delphes 3** has been out for two year now, with **major improvements**:

  - modularity
  - pile-up
  - visualization tool based on ROOT EVE
  - default cards giving results on par with published performance from LHC experiments
  - fully integrated within MadGraph5/Py8
  - updated configurations for future e+e- and hh colliders

- Delphes 3 can be used right away for fast and realistic simulation

- Continuous development (IP b-tagging, Jet substructure, New detector cards ...)

Website and manual:

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

People

Jerome de Favereau
Christophe Delaere
Pavel Demin
Andrea Giammanco
Vincent Lemaitre
Alexandre Mertens
Michele Selvaggi

the community ...

Back-up

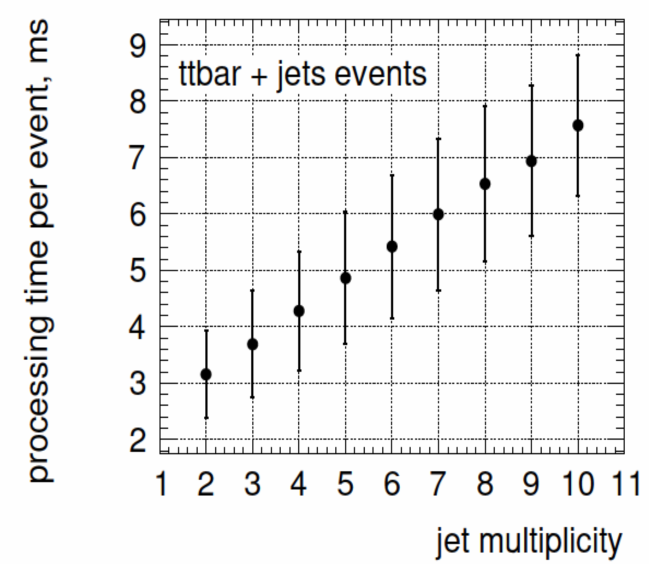
The Delphes Project: CPU time

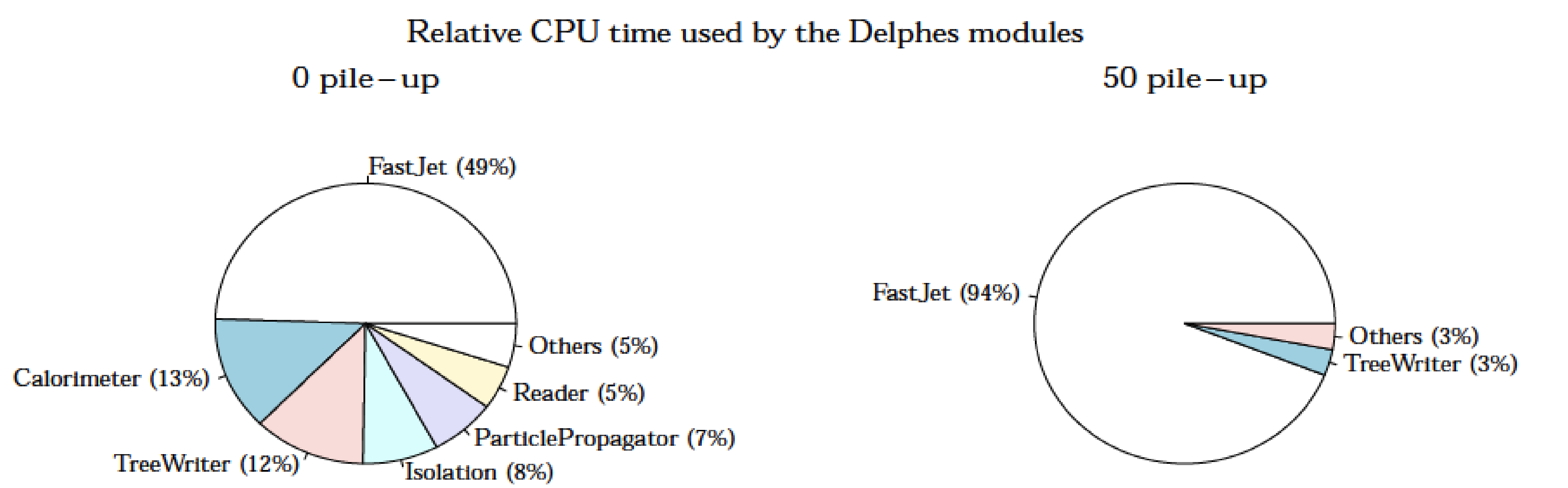
Delphes **reconstruction time** per event:

0 Pile-Up = 1 ms

150 Pile-Up = 1 s

Mainly spent in the FastJet algorithm:





The Delphes Project: disk space

Disk **space** for 10k ttbar events (upper limit, store all constituents):

0 Pile-Up = 300 Mb

100 Pile-Up = 3 Gb

Mainly taken by list of MC particles and Calo towers: