Reconstruction in an imaging calorimeter for HL-LHC

Antonio Di Pilato
University of Bari & INFN

*on behalf of the CMS Collaboration*
High Luminosity LHC project @CERN will provide more accurate measurements of new particles and enable observation of rare processes that occur below the current sensitivity level.

- **High pileup**
  - $7.5 \times 10^{34} \text{ cm}^{-2}\text{s}^{-1}$ peak luminosity
  - up to 200 simultaneous collisions per bunch crossing

- **High radiation levels**
  - 1 year HL-LHC ~ 10 years LHC
The **High Granularity Calorimeter** (HGCAL), will replace the current CMS endcap calorimeters for the High Luminosity LHC during LS3.
3D shower reconstruction
Covers $1.5 < \eta < 3.0$
Full system operates at $-35^\circ C$
600 m$^2$ of **silicon sensors**
- 6M channels
- 0.5 or 1.0 cm$^2$ cell size
- 120/200/300 mm cell thickness
400 m$^2$ of **plastic scintillators**
- 240k channels
- 4-30 cm$^2$ cell size
- SiPM readout
Total weight: 215 ton per endcap

For more information: [The Phase-2 Upgrade of the CMS endcap calorimeter Technical Design Report](#)
CMS High Granularity Calorimeter

CE-E (Electromagnetic Calorimeter)

- 28 layers per endcap
- **Active elements**
  - hexagonal modules based on silicon sensors
- **Absorber type**
  - Cu, CuW, Pb
- **Thickness**
  - $25X_0$
  - $1.3\lambda$

Distinctive feature of the detector
(hexagonal geometry in the reconstruction)

For more information: [The Phase-2 Upgrade of the CMS endcap calorimeter Technical Design Report](https://link-to-report.com)

CMS High Granularity Calorimeter

CE-H (Hadronic Calorimeter)

- 22 layers per endcap
  - 8 full Si layers
  - 14 mixed layers
- **Active elements**
  - hexagonal modules based on silicon sensors
  - Scintillating tiles
- **Absorber type**
  - Stainless steel, Cu
- **Thickness**
  - $8.2\lambda$

For more information: [The Phase-2 Upgrade of the CMS endcap calorimeter Technical Design Report](#)

Expected CPU Trend for the coming years [1]

What we get

- A factor of ~3 in improvements from CPUs evolution

[1] IT Technology and Markets, Status and Evolution
Computational Challenge

Expected Needs from Offline and Computing for CMS [1]

- A factor of ~30 more resources, from Offline and Computing (mainly driven by Monte Carlo simulations)

[1] HL-LHC computational challenge for ATLAS and CMS experiments
A factor of $\sim 10$ still missing!

A new approach is needed!

The software reconstruction of HGCAL cannot be obtained by adapting any existing sub-detector software. It’s a completely novel detector and represents a *unique opportunity* to design the reconstruction with modern architectures and new techniques in mind:

- Heterogeneous Computing
- Machine Learning
HGCAL Reconstruction Chain

- Particles deposit energy and create **RecHits** (the event display above shows **RecHits** in real test-beam data)
- **RecHits** on each layer are clustered together by the CLUE algorithm (*described later in this talk*) into **LayerClusters** (2D objects)
- **LayerClusters** are linked together to form **Tracksters**, collections of **LayerClusters** aligned like tracks (3D objects)

---

The Iterative CLustering

**TICL - The Iterative Clustering:** modular framework for particle reconstruction in HGCAL

- Final purpose is to process HGCAL *Rechits* and return particle properties and probabilities.
- Modules and interfaces are defined so that new developers don’t have to know anything about the CMSSW (official CMS Software) core framework in order to contribute.
- Modules are designed such that new algorithms or techniques (e.g. Machine Learning) can be plugged on top easily.
- More information about TICL Framework available on the [official HGCAL website](#).

This talk will cover **3 main aspects** of TICL Reconstruction:

1. **Description of a TICL iteration to produce Tracksters**
2. **2D Clustering with CLUE Algorithm**
3. **Particle Identification and Energy Regression on a Trackster**
TICL Components

Rechits → 2D Clusters → Seeding Region → Tracksters → PFTICLProducer

Tracks → PID → Denoising

Timing → Energy Regression

MIP Tracksters → Trackster Splitting
1. Description of a TICL iteration to produce *Tracksters*
2. 2D Clustering with CLUE Algorithm
3. Particle Identification and Energy Regression on a *Trackster*
A TICL Iteration

• **LayerClusters** belonging to physics objects are masked-out and won’t be available for the next iteration
• Restrict Masked **LayerClusters** to a seeding region
• Pattern Recognition (PR) algorithms are designed with parallelism in mind
• Linking, Cleaning and Classification, possibly exploiting Machine Learning
• Make use of timing information when possible

**Existing iterations**
- Track-seeded
- MIP
- EM
- HAD
The Pattern Recognition algorithm that has been implemented inside TICL is the **Cellular Automaton**.

1. Start from a Layer$_N$ and consider a specific LayerCluster
2. Open a window in the [$\eta$, $\phi$] space around it and project it onto the next Layer$_{N+1}$
3. Consider all the LayerClusters inside this region and try to establish a **Doublet** connection between the LayerClusters on the two adjacent Layers
4. Apply some compatibility criteria to decide if the LayerClusters should be linked or not (i.e. geometry constraints, energy, timing compatibility, ...)
5. Repeat this same procedure for all the LayerClusters on Layer$_N$
6. Repeat this same procedure for all pairs of contiguous Layers [Layer$_K$, Layer$_{K+1}$]

- At the end of this process, the LayerClusters will be linked by Doublets if they satisfy some configurable requirements
- The set of all connected Doublets forms a **graph** and is the building block of a **Trackster**

If properly configured, the CA can also establish links between non-adjacent layers, allowing for consecutive missing LayerClusters.
CLUstering by Energy (CLUE)

Features
- based on energy density of hits
- \( n = 10^5 \) hits \( \rightarrow k = 10^4 \) clusters
- small clusters \((n/k \sim 10)\)
- fast
- GPU-friendly

CLUE algorithm consists of several steps:
1. Build Spatial Index
2. Calculate Local Density
3. Calculate “Nearest Higher”
4. Promote Seeds and Demote Outliers
5. Assign Cluster ID

- CLUE algorithm was officially presented at [CHEP2019 in Adelaide, Australia](https://chepp2019.cern.ch) (credits to Z.Chen for the slides).
- The full talk is available [here](https://chepp2019.cern.ch).

CLUstering by Energy (CLUE) on GPU

**STEP 1: Build “Grid Spatial Index”**

- Grid tiles small compared to the size of HGCAL layer (exploiting high granularity)
- Each tile hosts indices of hits inside it
- Points inside a tile can be directly accessed.
- Complexity of query d-neighborhood is $O(1)$, given that $d$ is small (*fast-query*)

\[ \Omega_d(i) = \{ j : j \in \text{tiles touched by square window } (x_i \pm d, y_i \pm d) \} \]

\[ N_d(i) = \{ j : d_{ij} < d \} \subset \Omega_d(i) \]

**STEP 2: Calculate Local Energy Density**

- The Local Energy Density is defined as:

\[ \rho_i = \sum_{j:j \in N_d(i)} \chi(d_{ij}) w_j \]
The "Nearest Higher" is defined as the nearest point with higher local density:

\[ nh_i = \begin{cases} 
\arg \min_{j \in N'_{d_m}(i)} d_{ij}, & \text{if } |N'_{d_m}(i)| \neq 0 \\
-1, & \text{otherwise}
\end{cases} \]

- \( d_m = \max(\delta_s, \delta_o) \) where \( \delta_s \) and \( \delta_o \) are algorithm parameters for seed promotion and outlier demotion
- \( N'_d(i) = \{ j : j \in N_d(i), \rho_j > \rho_i \} \)
- Calculate \( \delta_i = \text{dist}(i, nh_i) \)
Each hit has a list of followers defined as:

\[ F_i = \{ j : n_{h_j} = i, \delta_j < \delta_o \} \]

Clusters obtained by pushing the cluster ID from seeds to its chain of followers iteratively

STEP 5: Assign Cluster ID

- Each hit has a list of followers defined as:
  \[ F_i = \{ j : n_{h_j} = i, \delta_j < \delta_o \} \]
- Clusters obtained by pushing the cluster ID from seeds to its chain of followers iteratively

STEP 4: Promote Seeds and Demote Outliers

- Promote hit as seed if \( \rho_i > \rho_c \) and \( \delta_i > \delta_s \)
- Demote hit as outlier if \( \rho_i < \rho_c \) and \( \delta_i > \delta_o \)
CLUE Performance

- CLUE algorithm has about 30x speedup over previous clustering in CMS Software framework (CMSSW)
  - designed with a thinking-parallel approach
  - overcomes the performance of previous clustering that relies on KD-Tree for neighborhood query and has an $O(n^2)$ loop and density sorting \[1\]

\[1\] The previous clustering algorithm is fully described on the official HGCAL website

---

Hardware

Intel i7-4770K (1 Thread)
A **GPU version** of CLUE was developed and tested within CMSSW
- The optimized GPU version gives an additional **6x speedup** on top of CLUE CPU version
- Includes operations that can be shared by other GPU reconstruction steps and be partial hidden if multiple CUDA streams work on different events → **30x speedup** over CPU if considering CLUE steps execution only

**Hardware**
- Intel i7-4770K (1 Thread) + NVIDIA GTX1080
**ParticleID & Energy Regression**

**Tracksters** are the target objects on which **Particle Identification** and **Energy Regression** are performed.

**Strategy**
- Single **Convolutional Neural Network** (CNN) to perform both tasks
- **Tracksters** «images» are built as **input** to a CNN
  - each pixel represents a **LayerCluster**
  - image size: $50 \times 10 \times 3$
    - 50 HGCAL Layers per endcap
    - 10 clusters on a Layer
    - 3 features (energy, $\eta$, $\phi$)
  - **LayerClusters** are sorted by decreasing energy in each layer
  - Zero padding applied for Layers with less than 10 **LayerClusters**
  - low energy **LayerClusters** lost in Layers with more than 10 **LayerClusters**
- **PID score** (probabilities) and **energy value** as **output**
  - classes under discussion

![Sample image for an electron/positron](image_url)
PID&ER Preliminary results

- Performance study outside CMSSW (preliminary) on a 4-classes model
- Very promising
  - works still ongoing to add more classes and improve performance
  - this study must follow the evolution of TICL framework in parallel
- Confusion between photons and electrons can be solved by adding information from the Tracker
- Improvements are needed for charged hadrons (especially at higher energies)
Conclusion

• High Granularity Calorimeter for the HL-LHC is a very ambitious project
  ▪ A “tracking” device with high hit multiplicity and precise time information
  ▪ Developing a reconstruction that fully exploits the features of the detector is a challenging and creative task

• TICL is being designed with throughput and modern software techniques
  ▪ A unique opportunity since it is impossible to reuse software
  ▪ Fertile ground for the exploitation of Neural Networks developed outside HEP

• Development in CMSSW: both online and offline computing will benefit from it
  ▪ Enable running heterogeneous reconstruction in CMSSW on HPC machines
Backup
Two Shower Separation

Reconstructed $e^+e^-$ showers from converted $\gamma$

high energy core visible in the layers with maximum energy deposition

separation of nearby showers
ParticleID & Energy Regression

Sample images

- $e^\pm (E = 77.21$ GeV
- $\gamma (E = 123.03$ GeV
- $\pi^\pm (E = 209.45$ GeV
- $\pi^\pm (E = 323.37$ GeV

ParticleID & Energy Regression

Sample images

- $\mu^+ (E = 159.41 \text{ GeV})$
- $\mu^- (E = 290.70 \text{ GeV})$
- $K^0 (E = 171.39 \text{ GeV})$