

Clustering and Reconstruction in HGICAL

FCC Software Meeting

Marco Rovere for the HGICAL DPG



Introduction

- Will cover few, key areas of **HGCAL** reconstruction
 - ▶ General Overview
 - ▶ **TICL**
 - ▶ Clustering (Layer Clustering)
 - ▶ Software R&D Activities



Overview

HGCAL

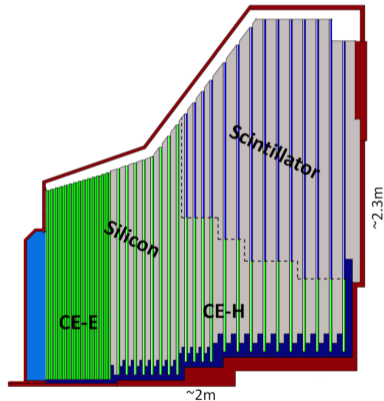
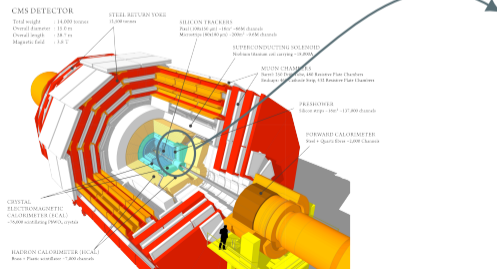
High Granularity Calorimeter: fine grain for 3D(5D?) shower reconstruction

⇒ Electromagnetic calorimeter (CE-E): Si, Cu & CuW & Pb absorbers, 28 layers, $25 \chi_0$ & $\sim 1.3\lambda$

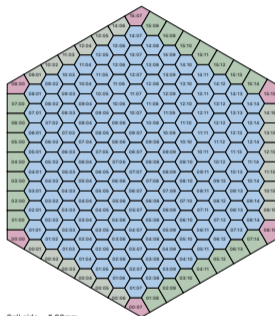
⇒ Hadronic calorimeter (CE-H): Si & Scintillator, stainless steel & Cu absorbers, 22(8+14) layers, $\sim 9.5\lambda$

Key Parameters:

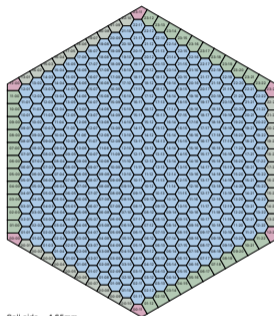
- HGCAL covers $1.5 < |\eta| < 3.0$
- Total size $z=2\text{m}$, $r=2.3\text{m}$. Total weight $\sim 215\text{ t}$ per endcap
- 620 m^2 of Silicon sensors (120/200/300 μm). $0.5\text{-}1.0\text{ cm}^2 \Rightarrow 6\text{M}$ channels
- 400 m^2 of plastic scintillators with SiPM readout. $4\text{-}30\text{ cm}^2 \Rightarrow 240\text{k}$ channels
- Intrinsic timing capabilities ($\sim 25\text{ps}$ resolution)



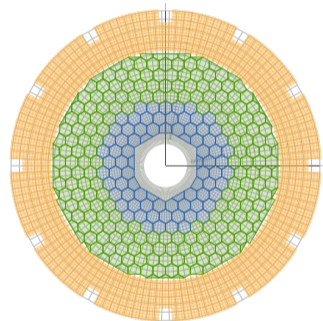
HGCAL Sensors



(a) Low density Silicon sensors



(b) High density Silicon sensors



(c) Layout of a layer(38) with silicon and scintillator sensors



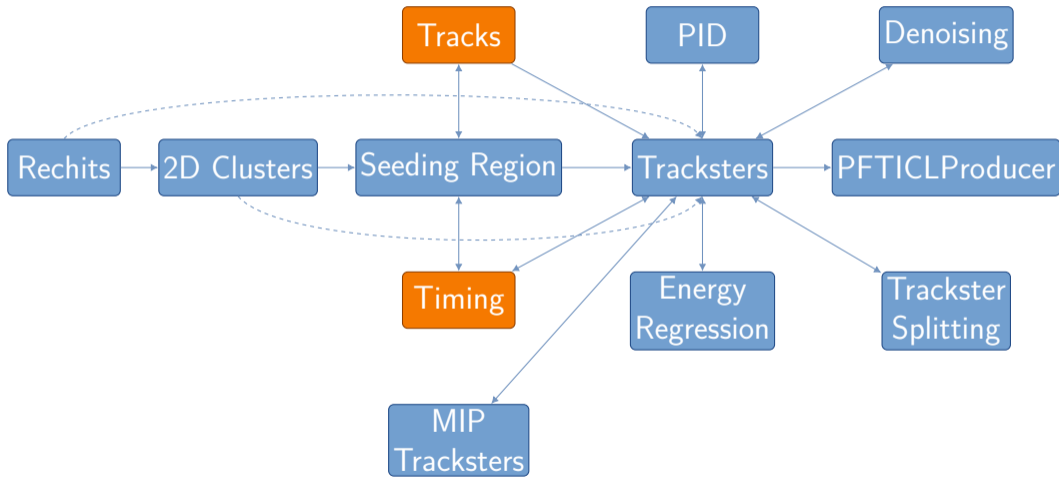
Reconstruction \diamond Overview

What is TICL?

- **TICL** (T-he I-terative CL-ustering) is a **modular framework** integrated in the **CMSSW** reconstruction
- Its 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** 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.
- *Mostly geometry independent*
- Algorithms are designed as plugins (within CMSSW)
- There is a dedicated R&D project inside the CERN R&D activities whose purpose is to abstract TICL and package it into a standalone library



TICL Components



A TICL Iteration

- $O(10^5)$ hits and $O(10^4)$ LC/event
- Layer Clusters belonging to already identified physics objects are masked-out
- **Pattern Recognition** is restricted to available Layer Clusters inside a **Seeding Region**
- Pattern Recognition algorithms are designed with parallelism in mind
- **Linking, Cleaning and Classification** to assign ID **with a given probability**, may exploit Machine Learning
- Use timing information, when possible

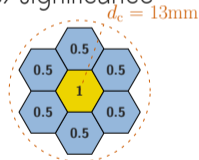


Reconstruction \diamond Clustering

The 2D layer clustering algorithm

- First introduced in May 2016, with significant coding and algorithmic changes and developments since then
- A key characteristic:
 - ▶ “Energy density” rather than individual cell energy is used to define ranking, seeding threshold, etc...
 - ▶ This concept appropriate for clustering cells that are small compared to shower size: “energy density” gives a better indication of the importance/significance of the cell

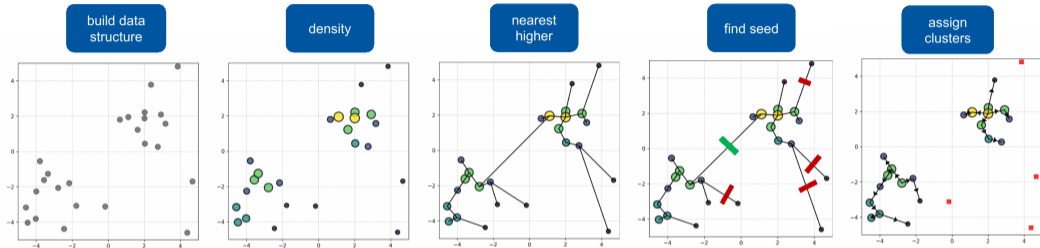
$$\rho_i = \sum_j E_j \times f(\Delta r_j); \quad f(\Delta r_j) = \begin{cases} 1, & \text{if } i = j \\ k, & \text{if } 0 < \Delta r_j \leq d_c \\ 0, & \text{if } \Delta r_j > d_c \end{cases}$$






- When used in seed selection, this definition picks out a local maximum cell as the seed
- And is more sensitive to a multicell blob of energy than just a single cell local maximum



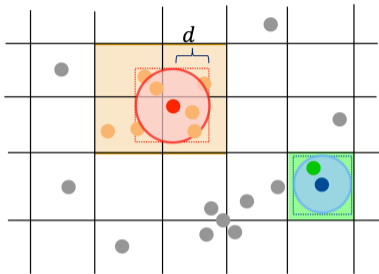
CLUE Block Algorithm



- Original Paper from which CLUE has been derived: 
- CLUE paper: 
- CLUE standalone code repository: 



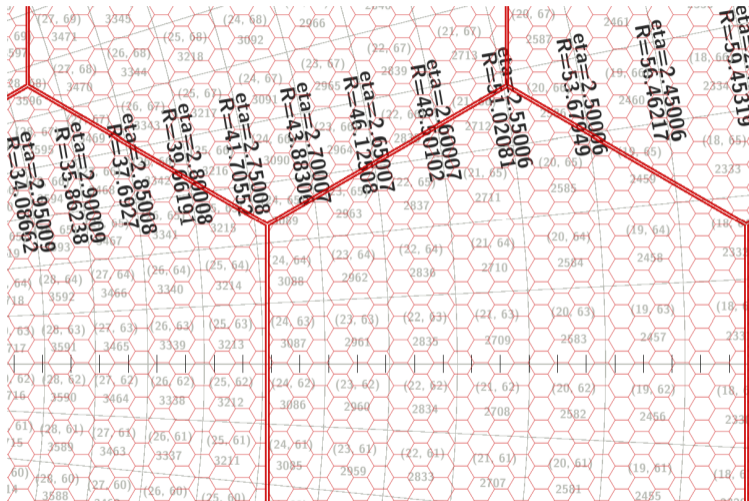
CLUE Tile data Structure



- Querying neighborhood N_d is a *frequent* operation in density-based clustering: \Rightarrow **need fast N_d query**
 - **d-searchBox**: $\Omega_d(i) = \{j : j \in \text{tiles touched by square window}(x_i \pm d, y_i \pm d)\}$
 - **d-neighborhood**: $N_d(i) = \{j : d_{ij} < d\} \subset \Omega_d(i)$
 - To query N_d , we only need to loop over hits in Ω_d
- Build **Grid Spatial Index** for hits on each layer (η, ϕ space)
 - ▶ Grid tiles are small compared to the size of **HGCAL** layer
 - ▶ Each tile in the grid hosts indices of hits inside it and has a fix length of memory to store the hosted indices.
 - Building spatial index is highly parallelizable on GPU.

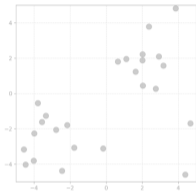


CLUE Tile data Structure

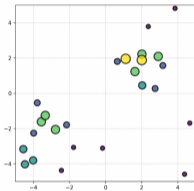


CLUE Density

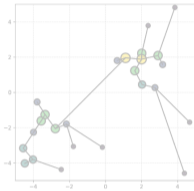
build data structure



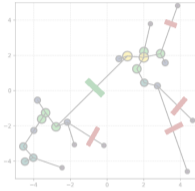
density



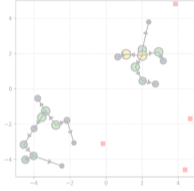
nearest higher



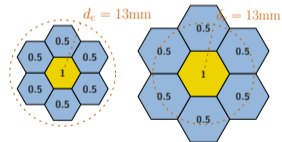
find seed



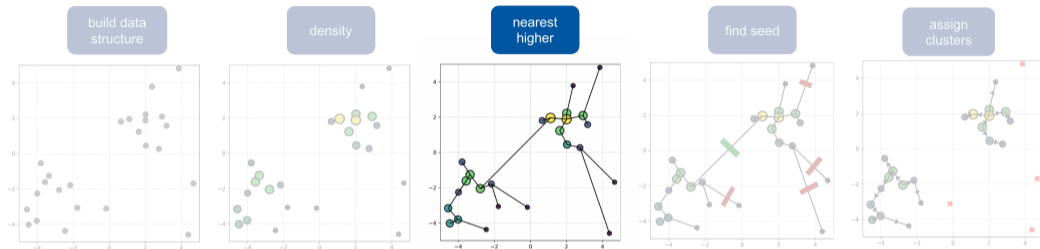
assign clusters



$$\rho_i = \sum_{j \in N_d(i)} E_j \times f(d_{ij}); f(d_{ij}) = \begin{cases} 1, & \text{if } i = j \\ k, & \text{if } 0 < d_{ij} \leq d_c \\ 0, & \text{if } d_{ij} > d_c \end{cases}$$



CLUE Nearest Higher

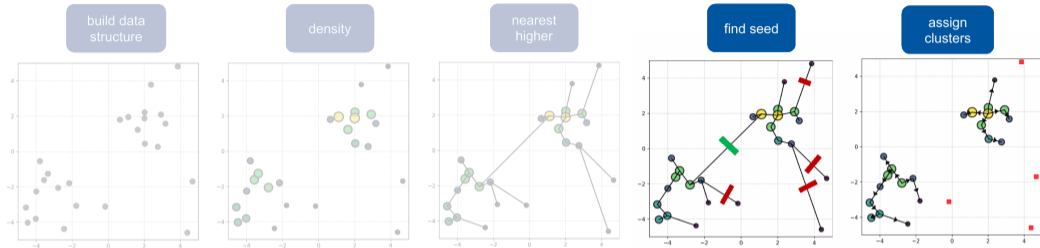


- Define $d_m = \max(d_s, d_o)$, d_s and d_o parameters for seed promotion and outlier demotion
- Within $N_{d_m}(i)$, find the **nearest point with higher density**
- Calculate $d_i = \text{dist}(i, nh_i)$

$$nh_i = \begin{cases} \operatorname{argmin}_{j \in \hat{N}_{d_m}(i)} d_{ij}, & \text{if } |\hat{N}_{d_m}| \neq 0, \hat{N}_{d_m}(i) = \{j : j \in N_{d_m}(i), \rho_j > \rho_i\} \\ -1, & \text{otherwise} \end{cases}$$



CLUE Find and Assign Clusters

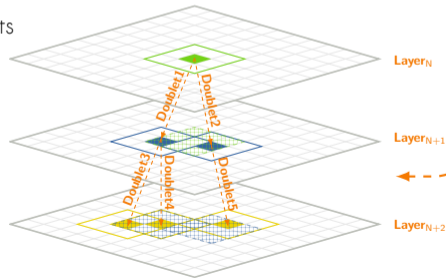
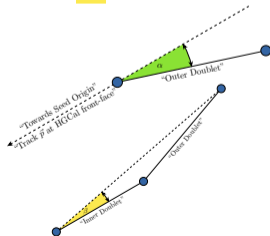
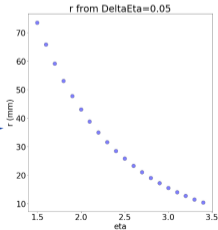


- **Promote as seed** if $\rho_i > \rho_c, \delta_i > d_s$
- **Demote as outlier** if $\rho_i < \rho_c, \delta_i > d_o$
- Assign unique, progressive clusterId to each cluster
- Push the clusterId from seeds to other hits through the reversed chains of nearest higher (followers)



Trackster construction using Cellular Automaton

- For each 2D layer cluster in a **Layer_N**, open a search window in **Layer_{N+1}**
 - ⇒ Search uses a 2D histogram in η, ϕ
 - Bin size is 0.05 (i.e. ~ 70 mm at $\eta = 1.6$ and 20 mm at $\eta = 2.8$)
 - ⇒ Search window is (3×3) $[(5 \times 5)]$ in the region $\eta < 2.1$ $[\eta \geq 2.1]$ bins centred on the bin in which the layer cluster sits
 - ⇒ A layer cluster in this search window will make a **doublet** with the original layer cluster
- Doublets are linked if timing and two angular requirements are satisfied:
 - ⇒ A requirement on the direction (α) of each doublet wrt the vertex (or wrt a track direction if this is a track seeded iteration)
 - ⇒ A requirement on the angle (β) between the doublets



TICL Iterations

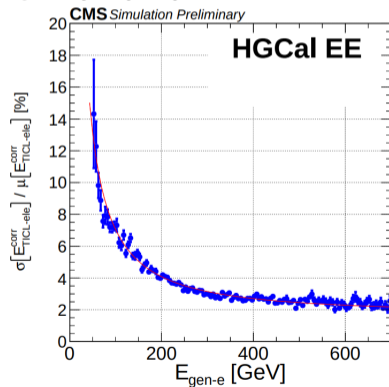
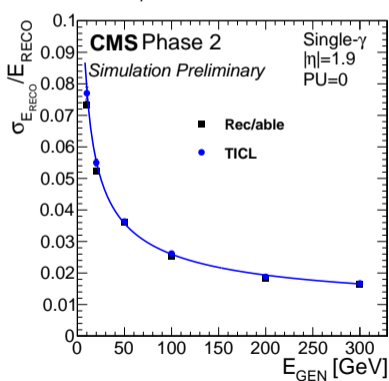
The following iterations has been implemented inside **TICL**

- **track-seeded** iterations (targetting electrons and charged hadrons)
- **electromagnetic** (electrons, photons)
- **hadronic** (neutral hadrons)
- **MIP-like** (muons)
 - ▶ This will require some *modification* to the CLUE algorithm in order to have high efficiency also for **single cell** (anti)-clusters.
 - ▶ Work is in an advanced state



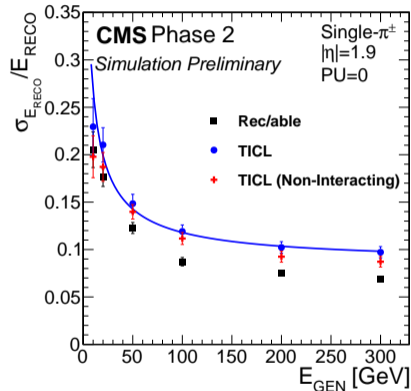
Electrons and photons

- TICL has been tested on *electromagnetic* showers
- Preliminary results are extremely encouraging
- A full blown *electron reconstruction* with *bremsstrahlung* recovery being worked out by EGAMMA POG using TICL tracksters.



Hadronic iteration

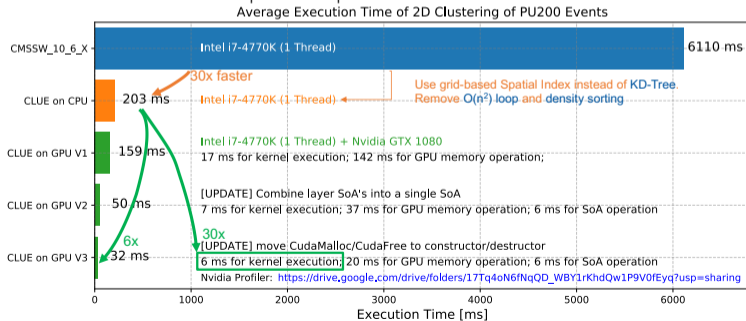
- Unlike em showers, hadronic showers vary in shape, they often resemble a branching tree of blobs or sub-clusters
- Testing an approach which attempts to collect the whole structure as a single trackster
- Initial results very promising.



R&D Activities \diamond Heterogeneous Computing

Heterogeneous HGICAL Reconstruction

- The complexity of the detector and the extrapolated computing needs of CMS pose challenges from the computation point of view
- Heterogeneous approaches are actively being explored and implemented
- 2D clustering has been fully ported and validated on GPUs
 - ▶ Order of magnitude in speed-up achieved with identical results
 - ▶ In addition to a similar speed-up in the CPU version



Conclusions

Summary

- The reconstruction in High Granularity Calorimeters at future colliders has many unprecedented challenges:
 - ▶ A “tracking” device with high hit multiplicity and precise time information
- **TICL** is being designed with throughput and modern sw techniques
 - ▶ A unique opportunity since it is impossible to reuse software
 - ▶ Fertile ground for the exploitation of neural networks
 - ▶ Integrated in **CMSSW** but loosely coupled to it
 - ▶ Allows to exercise the code frequently, validate it and profit from the CMS software development infrastructure
 - ▶ **TICL** is being developed with parallelism in mind:
 - many parts have been/are being ported to GPUs
- Making it a common library would benefit experiments at future colliders and contribute to **TICL**'s improvement by using it with different topologies/geometries
- Increase the interaction with groups developing similar PF-like reconstruction



Questions?



Backup

CLUE pseudo-code: Main

Algorithm 1: HGCal clustering

```
for layer  $\in$  HGCal-Layers do
  build tiles
  calculateLocalDensity()
  calculateDistanceToHigher()
  findAndAssignClusters()
end
```



CLUE pseudo-code: Density

Algorithm 1: HGCal clustering: calculate ρ

```
 $d_c$  = value for EE, FH or BH
for  $i \in$  points do
  for  $j \in$  tiles.searchBox( $i, d_c$ ) do
    if  $dist_{i,j} < d_c$  then
      |  $i.rho += j.weight \times (i==j ? 1:0.5)$ 
    end
  end
end
end
```



CLUE pseudo-code: Nearest

Algorithm 1: HGCal clustering: calculate δ

d_c = value for EE, FH or BH

for $i \in$ points do

 delta = MaxFloat

 nearestHigher = -1

 for $j \in$ tiles.searchBox($i, 2d_c$) do

 isHigher = ($j.rho > i.rho$) or ($j.rho == i.rho$ and $j > i$)

 if $dist_{i,j} < delta$ and isHigher then

 nearestHigher = j

 delta = $dist_{i,j}$

 end

 end

$i.delta = delta$

$i.nearestHigher = nearestHigher$

end



CLUE pseudo-code: Find and Assign Clusters

Algorithm 1: HGCal clustering: find seeds and assign clusters

```
 $d_c$  = value for EE, FH or BH
nClusters = 0
buffer
for i in points do
   $\rho_c = \kappa * i.\text{sigmaNoise}$ 
  isSeed = (i.rho >  $\rho_c$ ) & (i.delta >  $d_c$ )
  isOutlier = (i.rho <  $\rho_c$ ) & (i.delta >  $2d_c$ )
  if isSeed then
    i.clusterID = nClusters
    nClusters ++
    buffer.push(i)
  else
    if !isOutlier then
      nh = i.nearestHigher
      nh.followers.push-back(i)
    end
  end
end
end
while ! buffer.empty do
  i = buffer.back
  buffer.pop-back
  for fl  $\in$  i.followers do
    fl.clusterID = i.clusterID
    buffer.push-back(fl)
  end
end
end
```



Computing Motivations for Software R&D for CMS

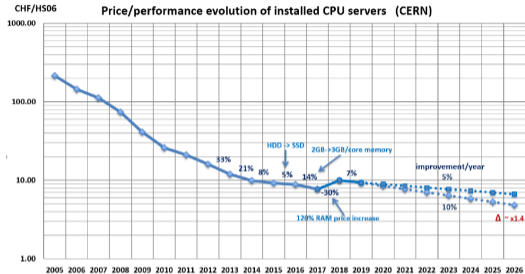


Figure: Expected CPU Trend for the coming years

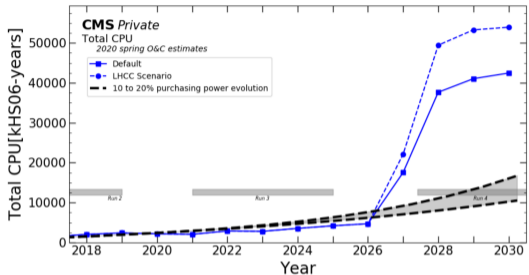


Figure: Expected Needs from O&C for CMS

- A factor of 2 in improvements from CPUs evolution
 - A factor of 10+ more resources, from Offline and Computing
- ⇒ A factor of 5 still missing

