Università degli Studi di Milano Bicocca

Scuola di Scienze Matematiche, Fisiche e Naturali

Corso di laurea magistrale in fisica

Towards a time-aware global event interpretation at the CMS experiment

Candidate:
Aurora Perego
842894

Supervisor:
Prof. Pietro Govoni

Co-supervisor:
Dott. Felice Pantaleo

Academic year 2022/2023

"I grew up with an ambition and determination without which I would have been a good deal happier. I thought a lot and developed the faraway look of a dreamer, for it was always the distant heights that fascinated me and drew me to them in spirit. I was not sure what could be accomplished with tenacity and little else, but the target was set high and each rebuff only saw me more determined to see at least one major dream to its fulfillment."

Earl Denman, *Alone to Everest*

Contents

Introduction

The Large Hadron Collider (LHC) [1] at CERN is the most powerful particle accelerator ever built. It has been operating since 2008 and has led to one of the biggest discoveries of the latest decades, the Higgs Boson [2, 3]. Four experiments are located along the LHC: ATLAS [4], CMS [5], ALICE [6] and LHCb [7]. The first two are general-purpose experiments, meaning that they cover a wide range of physics studies, while ALICE and LHCb are focused respectively on heavy ions physics and b physics to investigate differences between matter and antimatter.

In July 2022 Run 3 started and produced proton-proton collisions at an unprecedented energy of 13.6 TeV. At the end of this run the accelerator and its experiments will undergo major upgrades to get ready for the next phase of operations: the High-Luminosity LHC (HL-LHC) [8]. The plan of the High-Luminosity phase is to collect 3000 fb^{-1} of data, ten times the amount of data expected to be collected by the initial LHC configuration. To reach this goal both luminosity and pileup will be increased. The instantaneous luminosity, i.e. the number of collisions per unit of area and time, will go from 2 to $5 \cdot 10^{34}$ $\text{cm}^{-2}\text{s}^{-1}$. The pileup, i.e. the average number of collisions per bunch crossing, will go from 60 to 140. If the detectors can accept it, the luminosity could go up to $7.5 \cdot 10^{34}$ $\text{cm}^{-2}\text{s}^{-1}$ and the pileup up to 200. This would allow the collection of up to 4000 fb^{-1} of data before 2040. The massive detector upgrade planned will allow us to cope with the higher radiation levels and the increased event complexity presented by the High-Luminosity phase while maintaining the current performance.

The Compact Muon Solenoid (CMS) experiment [9, 10], one of the two general-purpose detectors at the LHC, has a broad physics program that will gain huge benefits from the upgrade of the LHC, ranging from precise measurements of the Higgs boson properties to direct searches for Beyond Standard Model physics (BSM). The amount of data collected will allow us to reach percent-level precision for most Higgs coupling measurements, while the current precision is about 20%. Studies on the Higgs self-coupling and measurements of the Higgs coupling to charged leptons (muons and taus) will also be possible. To face these unprecedented challenges CMS will introduce significant upgrades both in the sub-detectors and in the software.

One novel strategy that CMS has decided to adopt is the introduction of time

information in the collisions. To obtain this, the electronics of the current electromagnetic calorimeter (ECAL) [11] will be upgraded to provide time information with a precision of 30 ps for showers above 50 GeV. The endcap calorimeters will be completely replaced and the new High Granularity Calorimeter (HGCAL) [12] will provide excellent transverse and longitudinal segmentation and excellent timing resolution, designed for 5D shower topology identification. Its precision will be of the order of 25 ps for high energy deposits and will go up to a hundred of ps for the less energetic ones. In addition, a new dedicated detector for precision timing of charged minimum ionizing particles (MIPs), the MIP Timing Detector (MTD) [13] will be built. MTD will surround the tracker and with its time resolution of about 35 ps will add precision timing information to tracks.

This novel information opens possibilities for a time-aware reconstruction in CMS and enlarges its physics reach, giving new insights into phenomena like searches for long-lived particles (LLP), that would produce delayed signals in MTD, and heavy ion collisions studies, thanks to particle identification among low momentum charged hadrons, such as pions, kaons, and protons, using the time-of-flight between the collision vertex and MTD. The time information will be also crucial to mitigate pileup effects. The first part of this work is focused on integrating the time information in the event reconstruction and on evaluating its impact on the overall reconstruction performance.

After a general introduction to HL-LHC and the planned CMS upgrades given in Chapter 1 and to tracks and Particle Flow reconstruction in Chapter 2, the third chapter will be focused on the Monte Carlo simulation of events. This is a fundamental step that provides a benchmark against which to compare the reconstruction algorithms and therefore requires particular attention to realize objects that contain all the necessary information for this task. Chapters 4 and 5 present a first approach to a time-aware reconstruction and the effects of the presence of time information on electron isolation. The former is focused on studying the time information provided by the new sub-detectors, namely MTD and HGCAL, and on combing these two to improve the linking between tracks and energy deposits in the calorimeters, leveraging the fact that pileup signals will not be in time with the hard scattering ones. This is the same idea behind the use of timing in electron isolation studies, i.e. a technique used to check if a reconstructed electron is prompt or is more likely to come from pileup or even be a jet wrongly reconstructed as an electron. Timing information helps clean the region around the electron from pileup signals, increasing the possibility of identifying correctly its real nature.

CMS will bring substantial upgrades also to the data acquisition system because the increased event and detector complexity will need a higher processing power. To keep costs and power consumption contained, the processing power needed cannot be provided by CPUs only. Therefore CMS has decided to leverage heterogeneous computing, i.e. the integration of architectures designed to perform specific computational tasks much

faster and more efficiently with respect to CPUs. The CMS trigger is divided into two steps: the Level-1 Trigger and the High Level Trigger. The former reduces the data rate from 40 MHz, i.e. the collision rate, to about 110 kHz in Run 3, while the latter further reduces this to about 5 kHz. During Phase 2, the output rate of the Level-1 Trigger will be from 500 to 750 kHz and that of the HLT will increase accordingly.

The second part of this work is focused on the High Level Trigger, where Graphic Processing Units (GPUs) are used to perform part of the event reconstruction since the beginning of Run 3. To avoid writing multiple versions of the same code for each of the different architectures supported, performance portability libraries have been explored. These libraries allow the user to write a single version of the code and build it to run on different devices. As many portability libraries are available for this purpose, dedicated studies have been performed to evaluate the one that best fits the needs of the CMS software. My work in this context, presented in Chapter 6, has been focused on testing one of these libraries: SYCL/oneAPI [14, 15]. SYCL is a standard developed by the Khronos Group and oneAPI is the Intel implementation of SYCL specifically designed to target CPUs, Intel GPUs, and FPGAs, with extensions that allow running also on NVIDIA and AMD GPUs, covering all the main architectures and therefore making it of particular interest. The code for the pixel tracks and vertices reconstruction has been rewritten in SYCL and its physics and computing performance have been tested.

Chapter 1

The CMS experiment at the High Luminosity LHC

1.1 The Large Hadron Collider

The Large Hadron Collider (LHC) [1] is the largest and most powerful particle accelerator in the world. The LHC is a two-ring circular collider, with a circumference of about 27 kilometers, located underground at the European Organization for Nuclear Research (CERN) near Geneva, Switzerland. It accelerates two beams of particles, protons or heavy ions, in opposite directions, which then are made to collide at specific points. Protons in the beams are packed in so-called bunches that reach energies up to 6.8 TeV per beam and collide once every 25 ns. At this energy collisions happen between proton constituents, partons, and therefore the energy of the collision is not fixed, making it possible to study a broad range of energies.

There are four interaction points, one for each experiment: A Toroidal LHC ApparatuS (ATLAS) [4], the Compact Muon Solenoid (CMS) [5], A Large Ion Collider Experiment (ALICE) [6] and Large Hadron Collider beauty (LHCb) [7]. ATLAS and CMS are general-purpose detectors, ALICE is focused on heavy-ion physics and is designed to study the quark-gluon plasma, a phase of matter resulting from strongly interacting matter at extreme energy densities, while LHCb is dedicated to b-physics, i.e. physics of mesons containing the beauty quark, and studies on CP violation parameters. As already mentioned, ATLAS and CMS are called general-purpose experiments because they can cover a wide range of physics goals, including studies on the Higgs boson and its properties, searches for new particles and phenomena beyond the Standard Model, investigating the nature of dark matter and dark energy, exploring the behavior of quarks and gluons in the quark-gluon plasma, and the matter-antimatter asymmetry.

The LHC operations started in 2008 and are still ongoing, alternating periods of activity to shutdowns used to bring upgrades to the detectors and the collider. Three

main data-taking periods have been scheduled, Run 1 (2010-2013), Run 2 (2015-2018), and Run 3 (2022-2025) [16].

A useful value to understand the power of an accelerator is the luminosity, an indicator of the number of collisions that happen in the detector. The instantaneous luminosity quantifies the number of particle interactions per unit of time and unit of cross-sectional area, while the integrated luminosity is its integral over time. The instantaneous luminosity is defined as:

$$\mathscr{L} = \frac{N_1 N_2 f}{4\pi\sigma_x\sigma_y} \tag{1.1.1}$$

and is measured in $\text{cm}^{-2}\ \text{s}^{-1}$. N_1 and N_2 are the number of protons in the two bunches ($\sim 10^{11}$), f is the frequency of the collision (40 MHz), σ_x and σ_y are the sections of the bunches along the x and y direction respectively ($\sim 10\ \mu$m). The nominal value for the instantaneous luminosity at the LHC is $2 \cdot 10^{34}\ \text{cm}^{-2}\ \text{s}^{-1}$.

The integrated luminosity represents the data collected over a certain data-taking period and is measured in inverse femtobarn (fb^{-1}). It can be easily used to estimate the number of signal events expected multiplying the integrated luminosity by the cross-section of the process of interest. At the end of LHC operations, about 400 fb^{-1} of data will be collected [17].

Another important concept is the pileup (PU), i.e. the number of collisions overlapping with the one of interest, the hard scattering. The average (in-time) pileup during LHC operations went from less than 5 at the beginning of Run 1 to around 60 in Run 3. This number indicates the average number of proton-proton collisions when two bunches collide. Since collisions happen every 25 ns (50 ns in Run 1), pileup can come also from previous and subsequent bunch crossing and in this case is called out-of-time (OOT) pileup.

1.2 The Compact Muon Solenoid experiment

The Compact Muon Solenoid (CMS) detector is one of the two general-purpose experiments located along the LHC. It is located about 100 m below the ground. It is composed of different sub-detectors arranged in an onion-like structure, as shown in Figure 1.1, with a diameter of 15 m and a length of 28 m, with a total weight of 14000 ton. The main feature of CMS is a superconductive solenoid magnet with a magnetic field of 3.8 T. This magnetic field bends the paths of charged particles as they pass through the detector, making it possible to determine their charge and transverse momentum.

Inside the solenoid, the sub-detector closest to the beamline is the tracker. It consists of layers of silicon detectors, which enable precise tracking and reconstruction

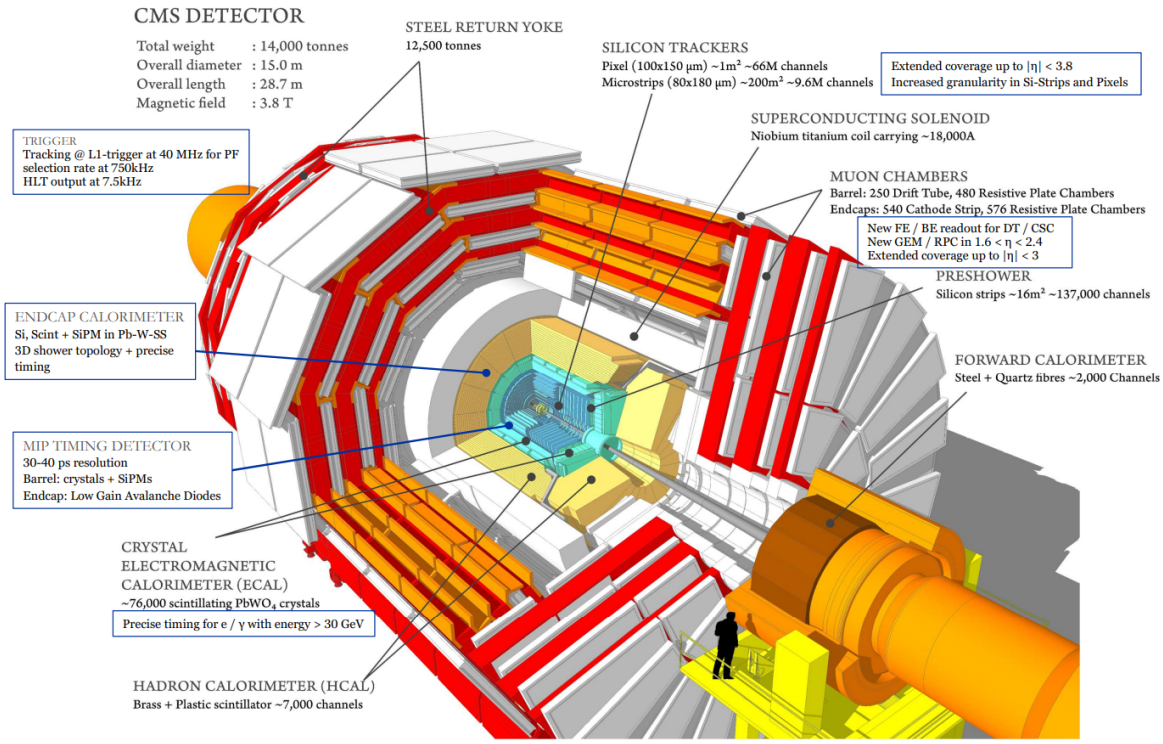


Figure 1.1: Scheme of the Compact Muon Solenoid (CMS) detector at the Large Hadron Collider with the Phase 2 upgrades highlighted in blue [18]

of the trajectory of charged particles emerging from collisions, providing valuable information about the particles' momenta and vertices.

Between the tracker and the solenoid lie the electromagnetic and hadronic calorimeters. These calorimeters measure the energy deposited by different particles. In particular, the electromagnetic calorimeter is responsible for detecting photons and electrons, while the hadronic calorimeter is dedicated to measure the energy of hadrons.

Finally, on the outermost layer of the CMS detector, there are the muon chambers. Muons produced during the collisions are highly penetrating and can travel through several meters of material, being the only type of particle, except for neutrinos, that can reach these chambers and exit the detector.

Not all the data produced at each collision in the detector can be saved nor need to be saved. For this reason CMS has a trigger system that selects the collisions of interest. It is divided into two levels, the Level-1 Trigger (L1) and the High-Level Trigger (HLT). The L1 Trigger is the initial stage of the trigger system and makes decisions within 4 μs to identify potentially interesting events within the enormous amount of collision data generated at the LHC. It is built with FPGAs and dedicated electronics and analyzes only the data coming from the calorimeters and muon detectors. It reduces the data rate from 40 MHz to about 100 kHz.

The High-Level Trigger is implemented in software, running on a farm of commercial computers equipped with x86 CPUs and, since the beginning of Run 3, NVIDIA GPUs. The HLT runs reconstruction algorithms on the full event data. The selection has to be made in a few hundreds of ms and the data rate goes from 100 kHz to about 5 kHz. The events that pass all the selections are saved for offline reconstruction and analysis.

In Figure 1.2 there is a representation of the coordinates system in CMS, right-handed and with the origin set to match the nominal collision point. The x-axis points to the center of the LHC rings, the y-axis points up and the z-axis is along the anticlockwise beam direction. The azimuthal angle (ϕ) is measured anticlockwise from the positive x-axis in the range $[-\pi, \pi]$, while the polar angle (θ) is measured from the positive z-axis. Since in a collision the rest frame is not known, it's important to define variables that do not depend on the boost. When a particle with momentum $\vec{p}$ is produced, the only component that can be measured is the one in the transverse plane, i.e. the transverse momentum $\vec{p_{\mathrm{T}}}$. Another coordinate of interest is the pseudorapidity which is defined as:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right)$$

It goes from $-\infty$ in the negative z-axis to $+\infty$ in the positive z-axis. It evaluates to 0 for $\theta = 90°$, to 0.88 for $\theta = 45°$, to 2 for $\theta = 15°$ and so on.

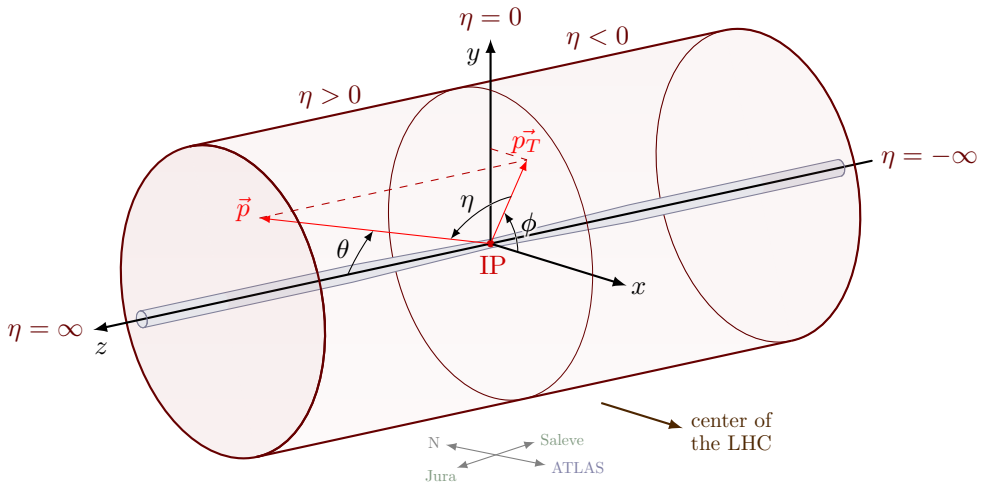


Figure 1.2: CMS coordinates system: right-handed and with the origin set to match the nominal collision point. The x-axis points to the center of the ring, the y-axis points up and the z-axis is along the anticlockwise beam direction. The azimuthal angle (ϕ) is measured anticlockwise from the positive x-axis in the range $[-\pi, \pi]$, while the polar angle (θ) is measured from the positive z-axis. The pseudorapidity η goes from $-\infty$ in the negative z-axis to $+\infty$ in the positive z-axis.

1.3 The High Luminosity Large Hadron Collider

The High-Luminosity Large Hadron Collider (HL-LHC) [8] project was approved in 2011 and will start in 2029, as shown in Figure 1.3, after a long shutdown during which the experiments will bring the necessary upgrades to the detectors. It will extend the LHC program with Run 4 (2029-2032), Run 5 (2035-2038), and Run 6 (2040-2041) and foresees two scenarios: Nominal and Ultimate. In the Nominal Scenario, the instantaneous luminosity will be $5 \cdot 10^{34}$ cm^{-2} s^{-1} and the integrated one at the end will be around 3000 fb^{-1}. In the Ultimate Scenario, the instantaneous luminosity will be $7.5 \cdot 10^{34}$ cm^{-2} s^{-1} with up to 4000 fb^{-1} of data collected.

The main reason for this upgrade is the need for more data, the higher the statistics, the more precise measurements can be. On one hand, HL-LHC will produce more than 150 million Higgs bosons enabling up to percent-level precision measurements for most Higgs coupling (the current precision is about 20%), allowing to explore the Higgs potential through HH production and probing Higgs coupling to fermions. On the other hand, the high statistics will allow studies of rare processes: Beyond Standard Model (BSM) searches, like exotic decays, Long Lived Particles, and Dark Matter as well as measurements of electroweak processes with small cross-section [20, 21].

The increase in luminosity, looking at Equation (1.1.1), can be obtained by increasing

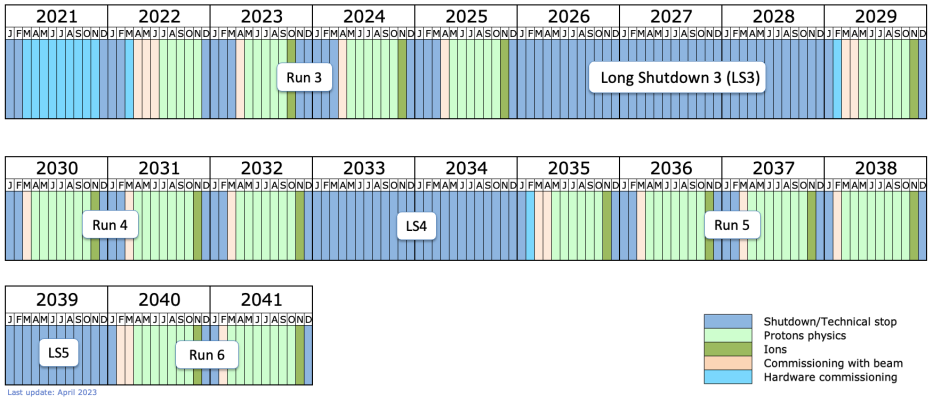


Figure 1.3: Long-term LHC schedule: after the end of Run 3, during the three-year long shutdown the collider and the experiments will do the necessary upgrades before the beginning of the High-Luminosity phase. HL-LHC will start in 2029 with Run 4 which will last four years. After another shutdown, Run 5 will begin and last until 2038, followed by Run 6 in 2040-2041 [19]

either the number of protons per bunch or the frequency. In Run 3 there are around 2800 bunches in each beam and the number cannot be increased more. Therefore the only possibility is to increment the number of protons per bunch and consequently the pileup. It will be around 140 in the Nominal Scenario and 200 in the Ultimate Scenario.

The presence of pileup complicates various stages of data processing. It results in numerous hits in the detector, leading to misidentified or mismeasured tracks. Additionally, pileup adds extra energy to the calorimeter measurements associated with the collision containing the hard scatter. This poses challenges to the trigger system, responsible for identifying and selecting interesting events in real time. Pileup events can interfere with the accurate determination of trigger thresholds, affecting the efficiency of event selection. Furthermore, pileup affects offline reconstruction and interpretation of events. The noise and complexity added make it more difficult to extract meaningful information from the collision data.

During HL-LHC the majority of data read out will be coming from pileup collisions rather than from the hard scattering and the volume of information that will need to be processed and analyzed will substantially increase [22]. As a consequence, the presence of pileup will also impact the execution time for event reconstruction, both in the High-Level Trigger and in offline analysis. The increased complexity arising from pileup necessitates more computational resources and longer processing times to accurately reconstruct and analyze the events. Strategies to mitigate the effects of pileup will be crucial for accurate event reconstruction and interpretation during the High Luminosity Phase of the LHC.

Another consequence of the luminosity increase is the rising of the radiation level the detector will go through. The radiation induced by the particles created in the collisions both in the material and in the electronics of the detector can cause significant damage and in the long term result in a degradation of the detector performance. Designing more radiation-hard detectors will be crucial to get to the end of the HL-LHC minimizing performance and resolution degradation,

1.4 Compact Muon Solenoid upgrades for the HL-LHC

To face the challenges posed to the detectors due to the increase in pileup and radiation during HL-LHC, CMS will bring upgrades both to the hardware and to the software [23].

Several parts of the detector will be replaced with new and more radiation-hard components since the expected annual dose delivered to the detector per year during the HL-LHC era will be similar to the total dose of all LHC operations. In addition, to mitigate the high pileup issue, improved granularity will be sought wherever possible.

In particular, the Tracker and Endcap Calorimeter systems will be completely replaced. The new tracker will be more radiation-tolerant and more granular to cope with the occupancy increase due to pileup. Its acceptance will be extended from $|\eta| = 2.5$ to $|\eta| = 4$ to enhance the physics capabilities [24]. Finally, tracking information will become part of the Level-1 Trigger reconstruction [25]. The Electromagnetic and Hadronic Endcap Calorimeters will be replaced with the High Granularity Calorimeter (HGCAL) [12] that will have an excellent spatial resolution and also a very good time resolution.

The front-end and very front-end electronics of the Electromagnetic Calorimeter (ECAL) [11], i.e. the on-detector electronics devoted to the first stages of signal processing, will be upgraded to reduce the shaping time of the signal. The benefits include a reduction of noise and out-of-time pileup contamination, a better resolution in the measurement of the time of arrival of the signal, and a bigger difference between scintillation and spike signals. The latter originated from particles that have deposited energy directly in the Avalanche Photodiodes (APDs) causing large signals that must be rejected before the online trigger [26].

To mitigate the in-time pileup, novel approaches, such as the use of dedicated timing detectors with an excellent timing resolution, are being explored. A new detector devoted to such measurements will be added between the tracker and the calorimeters, the MIP Timing Detector (MTD) [13].

The introduction of high-resolution time measurements will help disentangle the huge complexity derived from the high luminosity environment, being the only information that can help discriminate particles overlapped in space but not in time.

Regarding the muon system [27], the drift tube (DT) and Cathode strip chambers (CSC) electronics degraded by the radiation will be replaced and with the new one, there will be an increase in the trigger rate capability and performance.

To ensure a good performance in muon identification and reconstruction, a sufficient number of detector hits must be measured for each muon track. In the endcap regions, the high muon and background rates and the reduced magnetic bending, need an increase in the redundancy of the muon system to obtain a robust track reconstruction and reject the wrongly reconstructed tracks. For this reason, new Gas Electron Multiplier (GEM) detectors and Resistive Plate Chambers (RPC) will be added to cover a region up to $|\eta| < 2.4$.

Finally, to meet the requirements of High Luminosity, an upgrade of the Trigger and Data Acquisition (DAQ) [28] systems is necessary as well to ensure that the system can handle the influx of data while maintaining its optimal performance.

1.4.1 The MIP Timing Detector

The MIP Timing Detector (MTD) will be placed between the Outer Tracker and the calorimeters in a 4 cm wide layer and will cover a region up to $|\eta| = 3$. MTD will provide precise timing measurement of particles with a resolution of 30-40 ps, which will degrade to 50–60 ps by the end of HL-LHC operation due to radiation damage. Hits left by particles that go through MTD are associated with tracks and used to estimate the time of each track at the point of closest approach to the beam line. This time information reduces the number of tracks that are incorrectly associated with the hard interaction vertex by more than a factor 2 assuming an efficiency of 85% in associating time to a track. In addition, with the time compatibility constraint, the density of tracks along the beam line reduces from 1.9 mm^{-1} (in PU200) to 0.8 mm^{-1} (analogous to PU80), with the potential to almost recover the Phase-1 operations conditions.

The CMS physics program at the HL-LHC will benefit from the presence of MTD considering its important role in maintaining good resolution and reconstruction efficiency for the physics objects. The overall efficiency gain in measurements such as di-Higgs boson production or Higgs production in association with other particles will be about 20–30%. The b-tagging performance will be improved by the removal of pileup contributions, with effects on all the relevant di-Higgs boson final states (bbbb, bb$\tau\tau$, bb$\gamma\gamma$, bbWW, bbZZ). In addition, the use of timing will reduce the rate of pileup jets by a factor of two and improve the quality of the p_{T}^{miss} reconstruction, increasing the Vector Boson Fusion (VBF) tagging performance and also increasing the sensitivity to physics beyond the Standard Model (BSM) [13].

The novel track-time reconstruction will open new possibilities for CMS allowing it to look for phenomena that are outside the capability of the current detector. Searches for long-lived particles (LLP) will be tackled since heavy particles that travel in the

tracking volume before decaying into lighter standard model particles produce delayed signals in the MTD [29]. Moreover, the time of flight between the collision vertex and the MTD can be used to differentiate among low momentum charged hadrons, such as pions, kaons, and protons, providing new opportunities in Heavy Ion collisions studies [30].

To reduce time jitter, noise, and dark current rate MTD will operate at a temperature of -30°C. It will be divided into two sub-detectors that differ in realization and operating conditions: the Barrel Timing Layer (BTL) up to $|\eta| = 1.5$ and the Endcap Timing Layer (ETL) in $1.6 < |\eta| < 3$, A representation of the detector in GEANT can be seen in Figure 1.4.

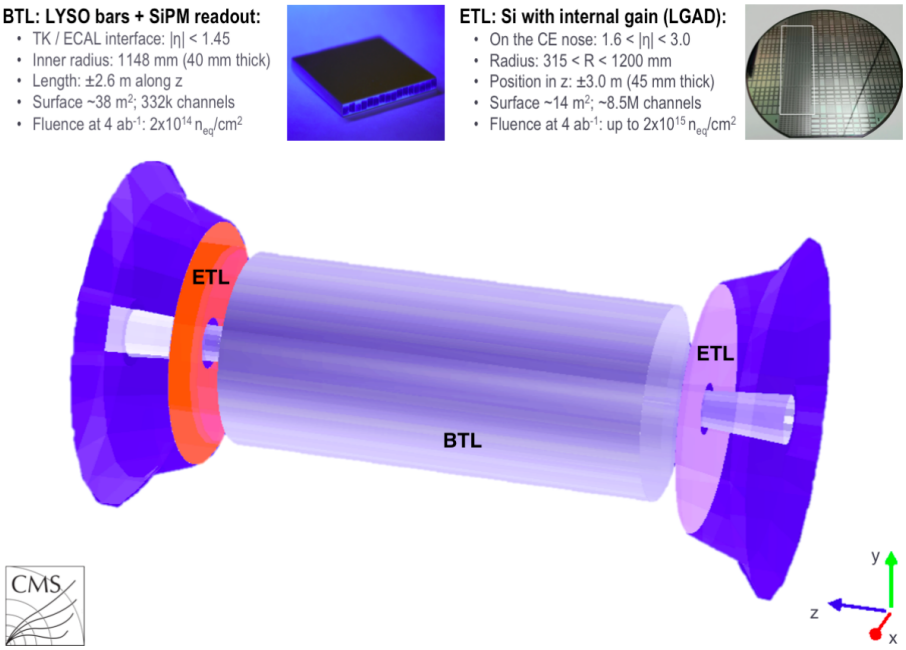


Figure 1.4: Representation of the MIP Timing Detector in GEANT. The Barrel Timing Layer, composed of LYSO:Ce crystal bars, is in the central region, and the Endcap Timing Layer, made of LGADs, on the two sides [13]

BTL will be composed of LYSO:Ce crystal bars of about 5.7 cm in length oriented along the ϕ direction in CMS, a width of 3.0 mm along the z direction and a radial thickness that varies along the barrel from 2.4 to 3.7 mm so that particles coming from the interaction point will travel approximately the same depth in the detector. A Silicon Photo Multiplier (SiPM) is placed at each end of the bars to provide two measurements of the time of arrival that can be combined to have a more precise estimate of the particle time and position.

The read-out is done by a dedicated ASIC, the TOFHIR (Time-of-flight, High Rate) chip. Each chip reads 32 SiPMs and performs discrimination of the leading edges of

their pulses followed by measurement with a time-to-digital converter (TDC). It also measures the amplitude of the pulse to correct the time walk.

Photons produced by a particle passing through BTL arrive at the SiPM following an exponential distribution with a decay time corresponding to the scintillation light decay time of the LYSO:Ce crystal, i.e. 40 ns. The rise time of the signal is about 14 ns and the decay time is about 300 ns. The discriminator threshold is between 5 and 50 photoelectrons, so only the initial photons contribute to the time measurement.

ETL will go through a radiation level substantially higher than BTL and therefore requires a different technology.

It will consist of two disks of silicon sensors, the Low Gain Avalanche Detectors (LGAD). The LGAD sensors have a gain of 10–30 that helps reduce noise sources and achieve a low-jitter fast-rising pulse edge. To achieve a good time performance at low gain the cell size must be small (< 2 mm^2). Therefore, a large number of pads are required. Each pad will contain an array of 16x16 LGAD. Each module is read out by two dedicated ASICs, called Endcap Timing Readout Chips (ETROCs). As for BTL, the readout chip uses the timing of the leading edge of the pulse to measure the time-of-arrival (TOA) and time-over-threshold (TOT) of each particle to measure the pulse height for time walk correction.

Contribution to the time resolution comes from the jitter, variations of the signal amplitude and shape due to fluctuations of the ionization process, signal distortion, and time to digital converter (TDC) binning.

The MTD reconstruction starts by clustering the energy deposits left by a particle in adjacent cells of the detector. This is relevant for BTL where a particle can cross more than one crystal, especially at high η. In ETL instead, the majority of the particles cross a single sensor. The cluster position is computed as the average of the hits' positions weighted on their energy, while the cluster time is the average of the hits' times weighted on their resolution. To reject pileup hits in a cluster are required to be time-compatible. The next stage is the propagation of the tracks trajectories to the MTD surface to look for spatially matched clusters. Tracks with time are used to reconstruct 4D vertices and different mass hypotheses (pion, proton, kaon) are tested to obtain the best association between track and vertex.

1.4.2 The High Granularity Calorimeter

The High Granularity Calorimeter (HGCAL) is a sampling calorimeter that will replace the existing Electromagnetic and Hadronic calorimeters in the Encap region of CMS. These calorimeters were designed for an integrated luminosity of 500 fb^{-1} and going beyond this will lead to an unacceptable loss of physics performance.

A sampling calorimeter works by stopping incident particles in absorbers made of lead or stainless steel and by measuring the energy released in the form of a shower of

secondary particles and radiation. The energy of showers is sampled with active layers alternated with the absorber layers.

The studies performed for the Phase 2 upgrade demonstrated that silicon sensors could tolerate high radiation levels and therefore these have been chosen as the active material for the high-radiation area of HGCAL.

In particular, the electromagnetic part (CE-E) will be composed of 26 samplings layers of silicon cells (∼0.5-1 cm^2), while the hadronic part (CE-H) will have 21 samplings with silicon cells (∼0.5-1 cm^2) and 14 layers of plastic scintillators (∼4-30 cm^2) read out by SiPMs in the low-radiation area. The result is a total of 47 sensor layers and 6 million channels to readout. A scheme of the detector is in Figure 1.5.

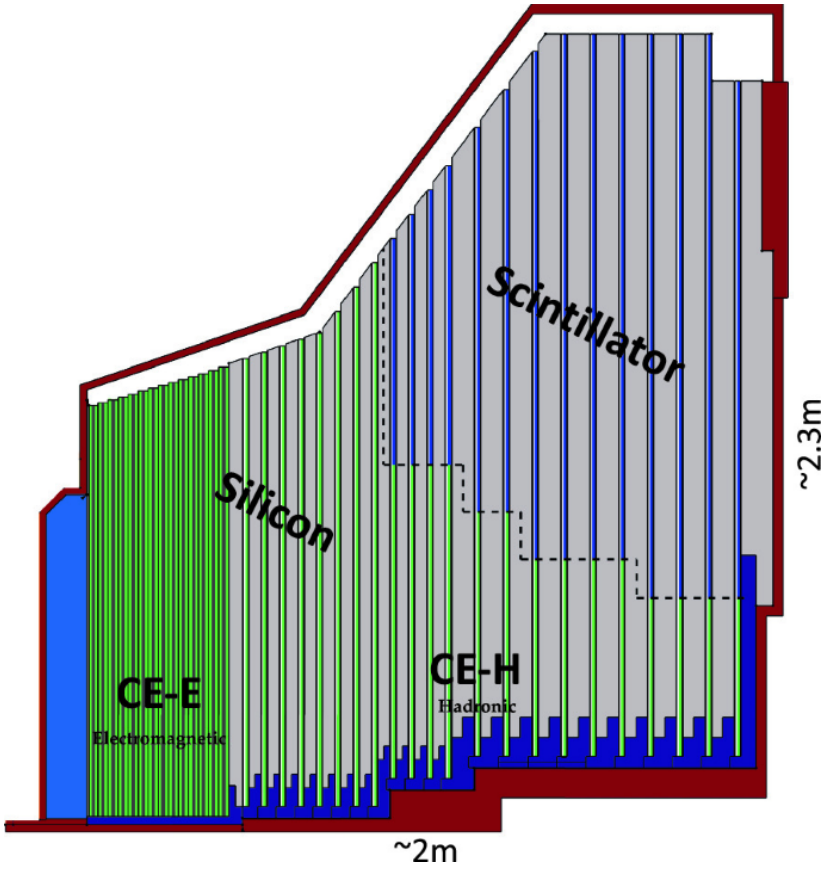


Figure 1.5: Longitudinal view of the High Granularity Calorimeter. The CE-E part is composed of 26 layers of silicon sensors and the CE-H part is composed of 21 layers of silicon sensors of which 14 layers are partially of scintillators. The light-blue element in front of HGCAL is a neutron moderator, about 20 cm wide. The first layer is at $|z| \simeq 322$ cm and the detector covers the region $1.5 < |\eta| < 3$ [12]

Each silicon sensor is composed of four layers: a baseplate, a Kapton-gold sheet, the

sensor, and the printed circuit board (PCB) with front-end electronics. The baseplate for the CE-E is made of WCu and is an important part of the absorber material, while carbon fiber will be used for the CE-H. The Kapton foil is used to provide a high-voltage connection to the silicon sensors. The pads of the silicon sensors or the SiPMs are connected to the input of the front-end ASIC, called HGCROC, that lies on the PCB.

Like MTD, to reduce the noise and the radiation damage, HGCAL will operate at a temperature of -30°C.

Thanks to its excellent granularity both in the longitudinal and lateral planes, HGCAL will provide precise measurements of energy deposits. In addition, it will also give time measurements for high energy deposits, allowing a 5D reconstruction (x, y, z, energy, time) of the showers.

This high granularity and excellent time resolution have huge benefits for the reconstruction of final state particles, the determination of their properties, particle-flow calorimetry, vertex identification, and pileup mitigation, providing good energy resolution even in a high pileup environment.

Precise time measurements are possible due to the fast response time of the silicon sensors and the design of the front-end electronics. The time is measured based on the energy deposited in a cell, with a threshold at 12 fC. Therefore a cell with significant deposited energy can give a precise time stamp. For example, an energy deposit of 50 fC has an expected time resolution of 25 ps.

For the silicon sensors, the charge measurement is performed with an ADC and with the time-over-threshold (ToT) technique with a TDC. ADC and ToT are combined to provide a single charge measurement. The pre-amplified pulse is also sent to a discriminator and the time-of-arrival (ToA) is measured with a TDC.

The HGCAL reconstruction starts from the clustering of reconstructed hits that lie on the same detector layer to create the so-called layer clusters. The algorithm used is CLUE [31], short for clustering of energy, a fast parallel algorithm that performs an energy-based clustering. The following step is the clustering along different layers, performed by a 3D version of CLUE, to create the so-called tracksters. An example is shown in Figure 1.6. An additional linking step between tracksters is performed, to obtain the final candidates for the particle flow algorithm.

1.4.3 The Trigger system

During the HL-LHC, the amount of data produced during a bunch crossing will substantially grow with respect to the LHC operations due to the increase in pileup and detector complexity. To keep the current 100 kHz output rate of the Level-1 Trigger it would be necessary to increase the trigger thresholds, with a huge impact on the physics reach. On the other hand, with the same threshold as Phase 1, the rate will increase up to six times the current one. The upgraded Level-1 Trigger will have about

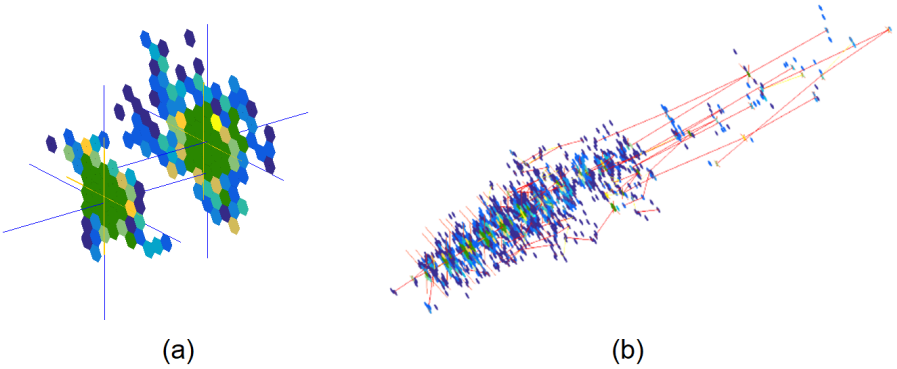


Figure 1.6: First two steps of the HGCAL reconstruction: (a) Example of two layer clusters built by CLUE coming from a 400GeV photon shower and (b) the corresponding trackster with the links found by CLUE3D between the layer clusters highlighted in red

12.5 μs to identify which collisions could be of interest and should be read out. The main upgrade will be the use of tracking information and particle-flow algorithms and it will select events at a rate up to 750 kHz to maintain the current physics performance. This will be possible thanks to the use of advanced field programmable gate array (FPGA) processors and high-speed optical links to ease the collection of data from the entire detector [25].

As a consequence, the High-Level Trigger input rate will increase and more processing power will be required to reconstruct and select the events in the available time, i.e. few hundred milliseconds.

1.4.4 The reconstruction software

Figure 1.7 shows projections of the offline computing resources needs of CMS in the upcoming years. The Baseline Scenario is the extrapolation of the current computing model to the HL-LHC era, with no additional developments. The Weighted Probable Scenario includes the impact of the ongoing R&D activities in optimizing the code both in simulation and reconstruction.

Looking also at the pie chart in Figure 1.8, the majority of CPU consumption comes from reconstructing the events, meaning that effort in improving the reconstruction algorithm will have a great impact on the overall CPU requirements.

To reconstruct and select the events, CMS employs a framework called CMSSW, namely CMS software [33]. The same framework is used for the online reconstruction at HLT and for offline reconstruction, simulation, and analysis. This allows the reuse of a large fraction of the reconstruction code developed for the offline reconstruction also online and to migrate novel offline algorithms and selection strategies to the online

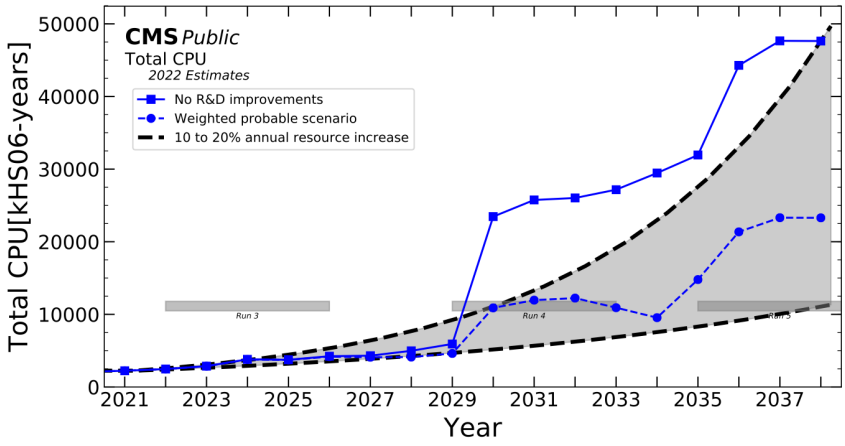


Figure 1.7: Prospects of the CPU requirement for offline data processing and analysis activities compared with the available CPU resources based on the level of R&D effort. This projection highlights the importance of developing strategies to reduce CPU usage, to remain within the budget [32]

environment.

The core of the execution in CMSSW is the Event Data Model (EDM) which is centered around the concept of `Event`. An `Event` is a C++ object that contains all the raw data and information related to a collision event. CMSSW is modular, meaning that each step of the reconstruction is implemented by a separate module that can be plugged into the configuration. Modules are implemented by C++ classes and compiled into plugins, shared libraries that can be loaded on demand. The framework maps modules to plugins and loads the ones that are needed by the configuration.

Each module can perform a different operation based on its type. To give some examples, there are `EDProducer` modules that can read data from the event and insert new data products, `EDAnalyzer` modules that are only allowed to read event data products, and `EDFilter` modules that can decide whether a trigger path should continue processing an event or not. The `Event` is used to pass data from one module to the next and is the only way the modules have to access or modify data. Some modules might require additional information to process events like the detector geometry and calibrations, these are provided through the `Event Setup` system.

CMSSW has been using since Run 2 the Intel Threading Building Blocks (TBB) library [34] to support a multi-threaded execution mode for module-level and event-level parallelism, meaning that independent modules within the same event and different events can be processed simultaneously. This is achieved by creating multiple `EDM`

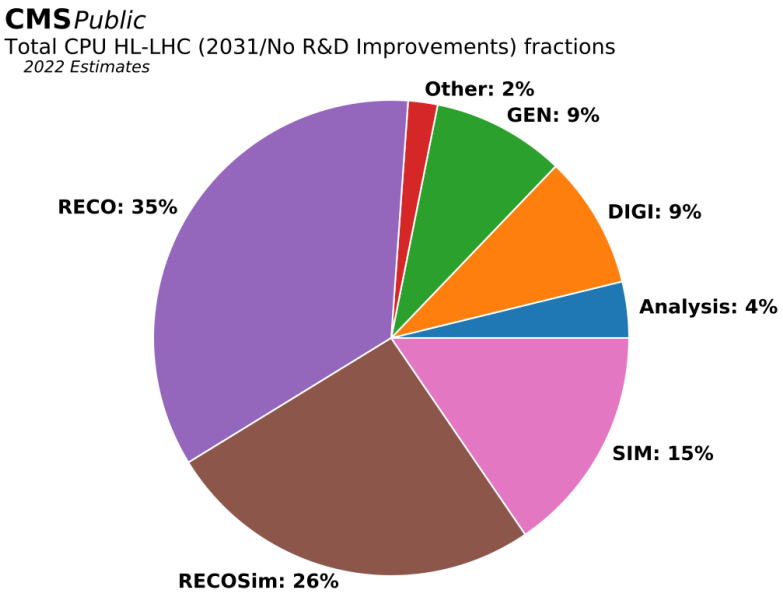


Figure 1.8: Breakdown of the CPU usages that includes CMS offline processing activities: prompt reconstruction, Monte Carlo simulation, data re-reconstruction and all phases of analysis activities [32]

`streams`. Each of them executes the entire workflow on a different event and the framework supports the creation and execution of multiple streams simultaneously.

At the HLT, the higher luminosity, pileup and input rate will require a processing power larger by more than an order of magnitude with respect to Phase-1, requiring an alternative approach to the CPU-only one. CMS has identified heterogeneous computing as the way to go to face this challenge. More details on heterogeneous computing and how it has been integrated into CMSSW will be given in Chapter 6, but the idea is that part of the work can be offloaded to co-processors specifically designed to handle parallel execution of specific tasks. Matching each job to its most appropriate architecture will allow it to achieve higher throughput and better energy efficiency. At the HLT, part of the reconstruction will be offloaded to graphic processing units (GPUs). GPUs have been in use since the beginning of Run 3 and have demonstrated an increase in throughput, i.e. number of events processed per second, with respect to the use of CPUs only. The fundamental redesign of the algorithms required to fully exploit the parallelism of GPUs also led to better physics performance. Currently, the following algorithms run on GPU: pixel unpacking, local reconstruction, tracks and vertices, ECAL unpacking and local reconstruction, and HCAL local reconstruction. Work is ongoing to run also the primary vertex and Particle Flow reconstruction on GPU, while part of the HGCAL reconstruction for Phase 2 has already a GPU version.

Chapter 2

Particle Flow reconstruction in CMS

After a general introduction to CMS and its upgrades for the High-Luminosity Phase of the LHC, this chapter dives into the current reconstruction algorithms, with a focus on tracks and Particle Flow reconstruction that are the most relevant for the content of this thesis.

2.1 Introduction to the Particle Flow Reconstruction

Event reconstruction in high-energy physics experiments is a complex and essential process that transforms raw detector data into a coherent and insightful understanding of the particle interactions occurring during collisions. A detailed and accurate reconstruction forms the basis for all the subsequent analyses, including the search for rare events, precision measurements, and the identification of novel physics signatures.

During a collision, an intricate array of particles is produced and they interact with the various sub-detectors. The raw data collected from these detectors, need to be integrated to obtain a comprehensive description of the event's fundamental constituents.

Traditional reconstruction techniques for physics objects were focused on localized information from a given sub-detector, e.g. using only ECAL for electrons or the muon chambers for muons. The advent of innovative algorithms like Particle Flow (PF) has revolutionized this approach. The Particle Flow algorithm is based on the concept of global event interpretation and goes beyond isolated subsystem analyses, correlating inputs from the entire detector to identify each final-state particle, and combining the corresponding measurements to reconstruct the particle properties. An illustration of this is in Figure 2.1.

After the local reconstruction, the reconstruction of each particle proceeds with a link algorithm that connects the PF elements from different sub-detectors producing the so-called PF block of elements, that are associated either directly or indirectly

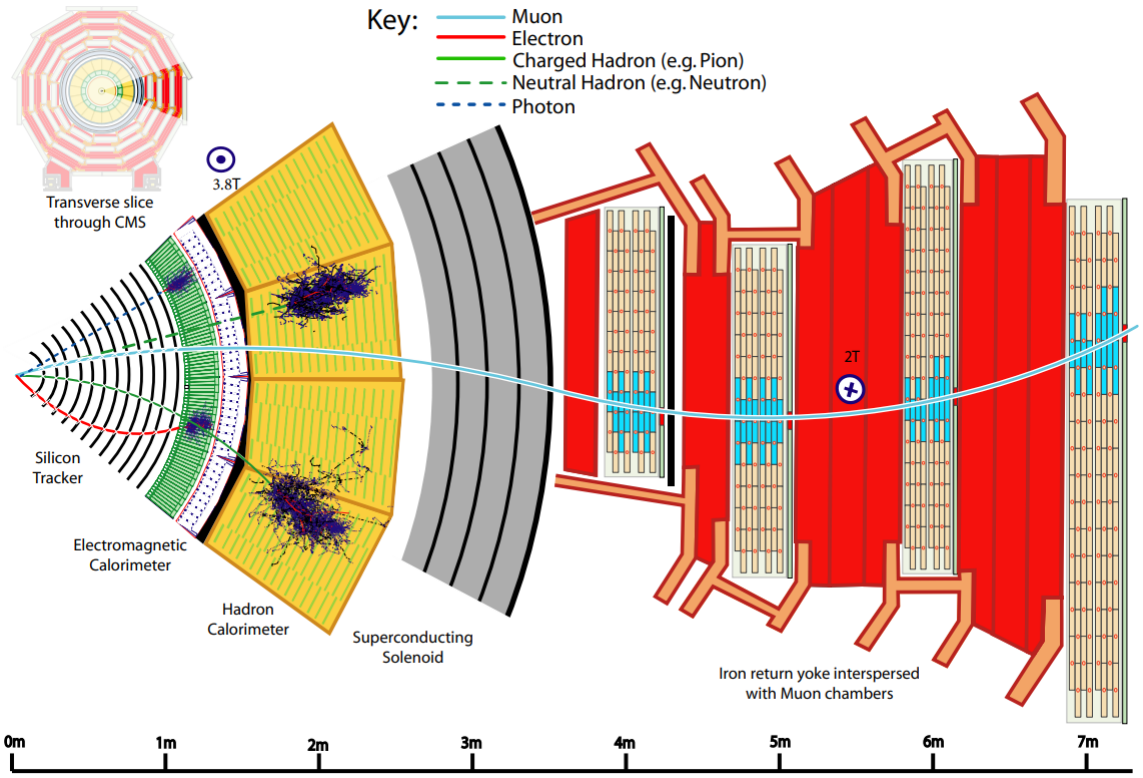


Figure 2.1: Transverse view of the CMS detector with the signals left by different particles that interact with the different sub-detectors. A photon and an electron leave energy deposits in the electromagnetic calorimeter and the electron, being charged, leaves also a track in the tracker. The situation is similar for neutral and charged hadrons, with the difference that most of the energy is deposited in the hadronic calorimeter. The muon, in turn, leaves a track both in the tracker and in the muon system. Reconstructing these particles based only on the information coming from a single sub-detector would result in a huge loss of information [35]

through other elements in common. In each PF block, the initial focus is on the reconstruction of muon candidates. Muons are identified as a track in the tracker linked with the corresponding track in the muon detector. Once a candidate has been reconstructed, the associated PF elements are removed from the PF block. Subsequently, the reconstruction process continues with the electrons, including the collection of energy originating from Bremsstrahlung photons, and with the identification of energetic and isolated photons, converted or unconverted. Electrons are identified by linking a track with a cluster in the electromagnetic calorimeter and with no signal in the hadronic calorimeter, while photons do not leave a track. The remaining constituents within the PF block undergo a meticulous procedure of cross-identification to categorize charged hadrons, neutral hadrons, and photons. Also secondary particles originating from nuclear interactions are subject to the reconstruction process, contributing to a comprehensive depiction of the particle interactions within the given event. Charged hadrons are identified by linking the track to one or more calorimeter clusters, while neutral hadrons are identified by calorimeter clusters not linked with a track [36].

The PF algorithm has been used for the first time at the ALEPH experiment of the LEP collider and will be of fundamental importance for the design of future experiments, considering its performance in obtaining physics objects with higher efficiency, resolution and reduced misidentification rate.

At the CMS experiment, the Particle Flow algorithm [35] was introduced around 2010 and its performance is enhanced by the presence of a strong magnetic field that can separate neutral from charged particles, a fine spatial granularity of the tracker, highly-segmented calorimeters that allow good separation of energy deposits from different particles and an excellent muon tracking system. The most evident benefits from Particle Flow are in Jet and p_{T}^{miss} performance, identification of hadronic taus and the input to pileup mitigation strategies like PUPPI (pileup per-particle identification) [37].

2.2 Tracks reconstruction in CMS

In CMS the measurement of the trajectories of charged particles is performed by the tracker. The CMS tracker is divided into an Inner Tracker (IT) and an Outer Tracker (OT). The Inner Tracker is composed of silicon pixels and has the goal of resolving the origin of the tracks and providing good seeds for the track reconstruction, while the Outer Tracker is composed of silicon microstrips and sacrifices some of the precision of the Inner Tracker to cover a larger volume and to extend the tracking capabilities [38].

Track reconstruction in CMS involves several stages to identify and reconstruct the path of charged particles created during the collisions. A track is the result of the fit of a set of hits left in the tracker, that provides an estimate of the particle trajectory. In a homogeneous and solenoidal magnetic field, the trajectory of a charged particle can be parameterized with a helix. A reference point useful for the parameterization is the

point of closest approach of the track to the beamline, $\vec{v}$. There are five parameters: the curvature as the inverse signed momentum (q/p in GeV^{-1}), the polar and the azimuthal angles, θ and ϕ respectively, of the track in $\vec{v}$, the signed distance between $\vec{v}$ and the beamline and the signed distance between $\vec{v}$ and the beamline on the $x-y$ plane.

The following is an overview of the track reconstruction process in CMS [39]:

- hit reconstruction: the first step is the identification and reconstruction of hits in the detector. Hits are the signals left by charged particles as they pass through the detector's sensitive modules. This step includes clustering information from strips or pixels to estimate the position of the hits and their associated uncertainty;
- seed generation: seeds are the starting point for the pattern recognition in the tracker and are built starting from two hits located on two different layers of the pixel tracker;
- pattern recognition or trajectory building: based on a combinatorial track finder relying on the Kalman Filter method, performs a first fit of the hits to obtain the trajectories;
- ambiguity resolution: ambiguities arise when a track is reconstructed starting from different seeds or when a seed results in more than one trajectory candidate. These ambiguities must be resolved to avoid double counting of tracks;
- final track fit: Once the most likely track candidates are identified, the tracks are refitted to obtain precise measurements of the track parameters.

Clustering in strips and pixels to obtain hits is seeded by sensors with a charge that is at least three times greater than the strip noise. Starting from each seed, neighboring strips are included in the cluster if the strip charge over its noise ratio is more than 2. To produce a hit, its total charge must exceed the cluster noise by at least a factor of 5. This step is known as local reconstruction. The proper track reconstruction starts with seeding which is done only in the pixel layers. The occupancy, i.e. the number of sensors activated in an event divided by the total number of sensors, in the pixel detector is much lower than in the Outer Tracker. In addition, spatial measurements in the pixel are three-dimensional and have a better resolution also because the particles have interacted with a smaller amount of material. Track seeding requires either two hits and a point along the beam spot or three hits [40]. Due to energy loss and interaction with the material of the detector the track fit with a pure helix parameterization is not optimal and other methods must be used. The CMS tracking strategy is based on the Combinatorial Kalman Filter [41]. At each layer, the Kalman Filter adjusts the track parameters and predicts the following layer until the propagation reaches the outermost layers. If no hit is matched on a given layer, the Kalman Filter can skip

it and propagate to the next one, unless the maximum number of skipped layers is reached. Once the track building is complete, ambiguities are addressed considering the fraction of hits that are shared between two trajectories. If this fraction is above a certain threshold the track with the least number of hits is discarded, or, if they have the same number of hits, the track with the highest χ^2 value is discarded. In the end, to avoid possible biases introduced during the seeding stage, the trajectory is refitted using a least-squares approach, implemented as a combination of a standard Kalman Filter and smoother [42].

2.3 Particle Flow in Phase 2

In the context of the High Luminosity Phase at the LHC, the biggest challenge that Particle Flow algorithms will have to face is the impact of high pileup scenarios that can significantly degrade performance if not effectively addressed. The presence of particles stemming from pileup interactions introduces supplementary tracks and energy deposits within the calorimeters. This results in an augmented contribution to the transverse momentum (p_{T}) and energy of reconstructed particles, potentially culminating in overlapping deposits that complicate the task of differentiating distinct particle signatures.

In the same vein, pileup tracks contribute to an increased line density, i.e. the density of collision vertices within a given unit of length. The association of a track to a vertex is done by looking at the point of the closest approach; to consider also tracks from displaced sources, the optimal selection window is about 1 mm. However, in a high pileup scenario, the density of tracks goes above 1 mm^{-1} (about 1.2 mm^{-1} in PU140 and 1.9 mm^{-1} in PU200). Notably, the performance of the current Particle Flow algorithm is degraded under such conditions of high line density because pileup tracks can fall into the 1 mm selection window.

The Phase 2 upgrades of the CMS detector coupled with the necessary Particle Flow software developments will be crucial to tackle the challenges posed by up to 200 pileup interactions. The fine-grained spatial resolution of the High-Granularity Calorimeter (HGCAL) will be important to disentangle signals emitted by distinct particles. Meanwhile, the precision timing information, combined from HGCAL and MTD, will play an instrumental role in pileup mitigation strategies. To give an example, the use of timing will play a crucial role in recovering the Phase-1 Particle Flow performance in the high pileup scenario with a line density greater than 1 mm^{-1}. The vertices spread along the beam line is about 180-200 ps and the timing resolution of 30-40 ps delivered by the new MIP Timing detector will allow to select only tracks compatible in time, resulting in an effective pileup comparable to the Run 3 conditions in each time window [13].

2.3.1 The Iterative CLustering framework

A candidate for CMS Phase 2 Particle Flow reconstruction is The Iterative CLustering (TICL) framework [43], an innovative modular software framework currently under development within CMSSW. Designed for the High-Granularity Calorimeter (HGCAL) reconstruction, TICL receives in input the rechits (hits left by the particles in the detector) and yields a comprehensive output that includes particle properties and probabilities for individual showers.

TICL aims at fully exploiting the fine granularity and the other significant features of the detector, such as particle identification and precision timing, strategically tailored to mitigate the effects of pileup. It also uses information from the other sub-detectors to perform seeding, clustering, energy calibration, and particle identification. Being modular, it offers maximum flexibility, permitting the amalgamation of various modules in diverse combinations. These modules can be chained together iteratively, thereby fostering adaptability to diverse reconstruction scenarios.

The framework has been developed with heterogeneous computing in mind: both the algorithms and their data structures are designed to be executed on Graphics Processing Units (GPUs). In addition, geometry-agnostic data structures have been realized to facilitate rapid navigation and efficient search capabilities.

The processes that take place during the TICL reconstruction are listed in Figure 2.2 and can be divided into three main configurable parts:

- Seeding Region Production and Layer Clusters Selection: involves the definition of seeding regions and the subsequent creation of layer clusters;
- Pattern Recognition: employs an algorithm to connect layer clusters to produce tracksters;
- Linking: dedicated to the task of connecting tracksters and tracks originating from the same particle, performing energy regression and particle identification.

The future plan is to extend TICL to the barrel region of the CMS detector, to provide a uniform Particle Flow framework for the whole detector.

2.3.1.1 Layer clusters creation

The first step in TICL is the clustering of adjacent hits lying on the same layer of HGCAL. The algorithm used for this task is CLUE, short for Clustering of Energy [31]. CLUE is a *fast parallel clustering algorithm for high granularity calorimeters in high-energy physics*. It follows a density-based approach with some specific optimizations implemented to give it a greater expression of parallelism. The input data of the algorithm are the hits, each having its corresponding coordinates (x, y, and $layer$) and $energy$. For each point, two key variables are calculated: the local density ρ and

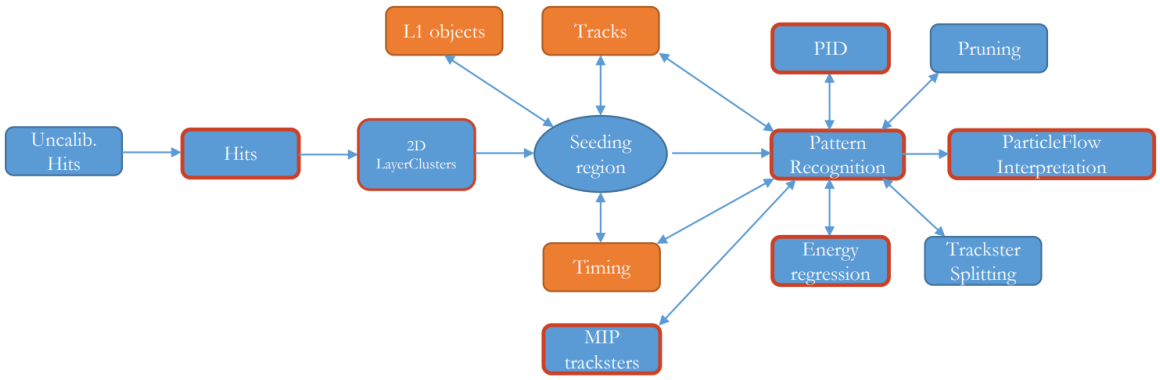


Figure 2.2: Different stages of the TICL reconstruction: it starts from the reconstructed hits, performs 2D clustering to obtain layer clusters, uses pattern recognition algorithms to join 2D clusters on different layers, and executes a final linking step to connect tracksters and tracks originated from the same particle. In addition to that performs Particle Identification (PID) and energy regression to obtain the final objects fed to the Particle Flow algorithm [43]

the separation δ. The first one represents the energy density in the area of the hit, while the second variable corresponds to the distance from the hit and the nearest hit with higher local density. From these two parameters, it is possible to cluster points depending on arbitrary thresholds set on a case-by-case basis.

The points are indexed with a fixed grid, which divides the space into rectangular bins, resembling the characteristic tessellation of sensor cells in the calorimeter. This indexing choice allows us to fast query the neighborhood of multiple points at the same time to better express parallelism on GPUs and when running with multiple threads.

CLUE requires three parameters that can be chosen at runtime:

- the critical distance, d_c: is the cut-off distance used for the calculation of the local density, can be tied to the expected shower size and the granularity of the detectors (possibly differing between silicon-based layers and scintillators);
- the critical density, ρ_c: is the minimum local density to promote a point as a seed or the maximum density to demote it as an outlier, can be tuned to maximize the signal-to-noise ratio;
- the outlier delta factor, odf: is a multiplicative constant applied to the critical distance to obtain the maximum distance between a point and its nearest point with a higher density over which the point is excluded from the clustering process, can be chosen considering the shower separation.

The use of configurable parameters allows CLUE to be more flexible at clustering different events for the specific desired goals of physics.

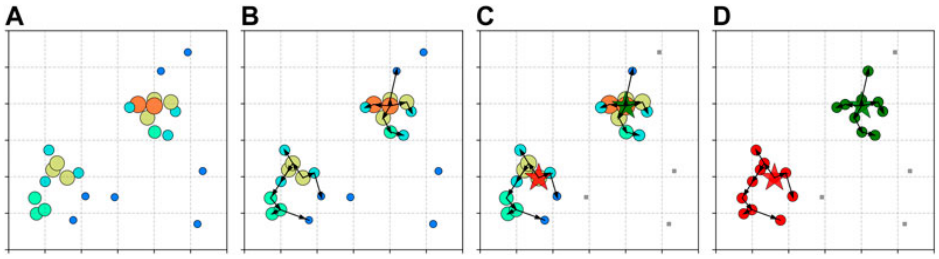


Figure 2.3: Schematic representation of CLUE clustering procedure: (A) creation of fixed spacial grid and computation of the local density for each point considering the neighboring points (B) find nearest point with higher density to each point (C) promote to seeds points with a local density and separation distance greater than the corresponding thresholds and mark not linked points as outliers (D) create the final clusters [31]

The clustering procedure of CLUE is shown in Figure 2.3. First, a fixed grid is constructed and each point is arranged in the bin mapped to its coordinates. Then, CLUE calculates the local density ρ for each point by querying its neighborhood, as shown by the different-sized and colored dots (A). Afterward, each point gets assigned a nearest higher, defined as the nearest point with higher local density, and their distance δ is calculated (B). Points with ρ higher than the critical density ρ_c and δ greater the critical distance d_c are promoted as seeds, while points with lower density and larger separation ($\delta > d_c \times odf$) are demoted to outliers. All the other points that are neither seeds nor outliers are marked as followers of their respective nearest highers (C). As a consequence of this logic, each seed has a chain of followers which can be iteratively navigated to build a cluster (D). At the end of the procedure, only clusters and outliers remain, with the latter points having no followers and not being followers of any other point.

CLUE reduces the problem size by one order of magnitude from rechits to clusters and is already available to be executed on GPU.

2.3.1.2 Tracksters creation with pattern recognition

Layer clusters are connected to produce tracksters using pattern recognition algorithms. Tracksters are Direct Acyclic Graphs connecting the Layer Clusters which serve as nodes. Within the framework of CMSSW, several plugins have been implemented to serve the purpose of pattern recognition. These include CLUE3D (the default one), Cellular Automaton, FastJet, SimTracksters, and MIP.

The created tracksters can be associated with seeding objects if needed and are filled with the indices of the layer clusters they are composed of, the barycenter position, its estimated direction using an energy-weighted PCA, the regressed energy values, the probabilities assigned to various particle identification classes and precise timing data.

Particle identification, energy regression, pruning, energy flow, shower shape, and

missing energy studies are performed on tracksters which then become the starting point for linking and ParticleFlow Interpretation.

2.3.1.3 Linking and TICL Candidates creation

The final step is the linking and Particle Flow interpretation, where, starting from the tracksters produced by the pattern recognition, all the information is gathered and combined to build the final objects. The CLUE3D pattern recognition algorithm is tuned to reconstruct extremely pure tracksters, at the price of splitting the shower into multiple energy clusters. The linking algorithm aims at connecting these energy clusters originating from the same particle to reconstruct the full shower. It also tries to connect tracksters with tracks and therefore distinguishes charged from neutral candidates.

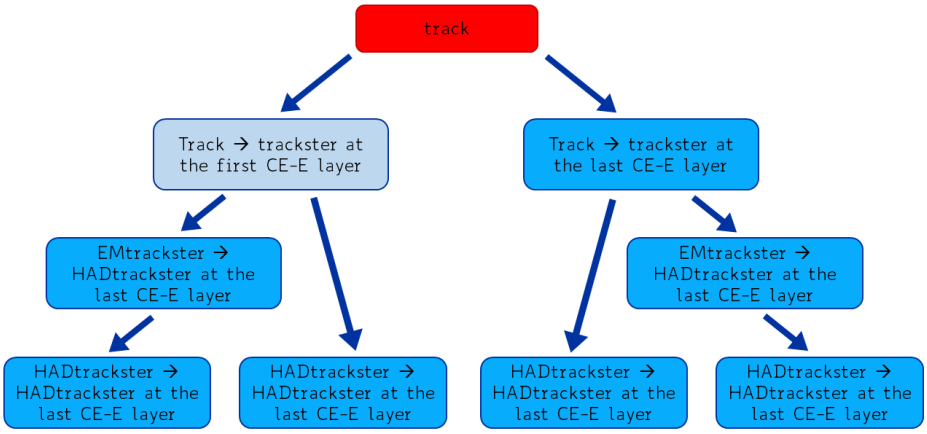


Figure 2.4: Linking for charged candidates: starting from the track, the linking track-trackster at the first and last CE-E layer is probed, then the same check is done between tracksters, first between one electromagnetic and one hadronic and then between two hadronics.

The possible link between track and trackster or between tracksters is probed at two different layers: at the first CE-E layer and at the last CE-E layer. The tracksters are divided into two categories based on the position of the barycenter. If the barycenter lies in the CE-E they are marked as electromagnetic, otherwise as hadronic. Tracks, electromagnetic and hadronic tracksters are propagated to the two layers where the linking will be attempted. The procedures are different for charged and neutral candidates and are sketched in Figure 2.4 and Figure 2.5 respectively.

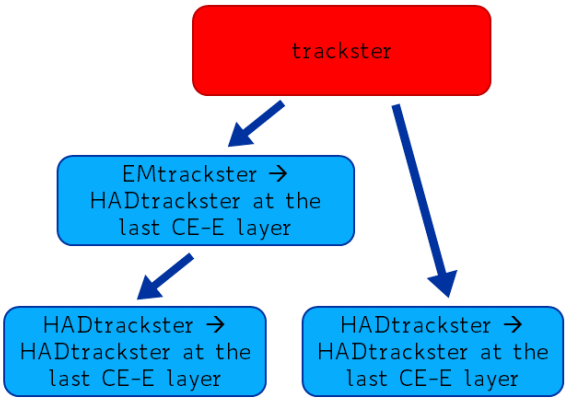


Figure 2.5: Linking for neutral candidates: the absence of a track marks the tracksters as neutral. The link is probed between tracksters only at the last CE-E layer firstly between one electromagnetic and one hadronic trackster and then between two hadronic

Tracks without linked tracksters are directly promoted to charged candidates. A detailed description of the linking algorithm is presented in Section 4.2.1. In addition to that, the energy regression and particle identification are run on the final tracksters using a Convolutional Neural Network (CNN) to obtain the energy and the most probable particle hypothesis of the candidates. All this information is included in the so-called TICL Candidates, which will be given in input to the Particle Flow algorithm.

Chapter 3

Simulation studies

Before looking at the reconstruction, this chapter presents studies that concern the simulation, a fundamental step to have a better understanding of the physics processes and the best reconstruction strategies to adopt.

Monte Carlo (MC) methods are computational techniques that model complex systems and processes through the generation of random numbers. In high energy physics experiments, Monte Carlo methods play a crucial role in understanding and predicting the behavior of particle interactions, detector response, and background processes. These are some examples where MC simulation is used:

- data analysis: simulation can be used as a benchmark to compare the expected outcome with the actual one given from the analysis of the observed data. It can also be used to estimate background contributions that mimic the signal of interest and can help in determining selection criteria to enhance the signal-to-background ratio. Differences between simulated and real data can lead to physics discoveries;

- event reconstruction: the data produced with the simulation can be compared with the reconstructed data and act as a benchmark for the reconstruction, being a reference to test and optimize the algorithms developed;

- detector design and calibration: to optimize the design and configuration of detectors to maximize the physics potential and calibrate detectors' response to different particles and energies to enable precise measurements.

The Monte Carlo simulation of an event is divided into two steps. The first one is the simulation of the trajectories and interactions of particles based on theoretical models. These interactions include the creation, scattering, hadronization, and decay of particles. Examples of tools that are used in these steps are MADGraph [44] and Pythia [45].

The former generates parton-level events: it can calculate the matrix elements for a wide range of particle interactions, as well as for Beyond Standard Model processes. Then Pythia is used for full event generation, which includes hadronization and decay processes.

The following step is the modeling of particles' interactions with the detectors. This involves understanding how particles generate signals in detectors, undergo energy loss, and produce various observable effects. This is done with Geant4 [46], a software toolkit developed by the high-energy physics community to simulate the interactions of particles with matter. Geant4 allows us to model the behavior of particles as they pass through different materials and detectors, providing insights into particle interactions, energy deposition, and the response of experimental setups.

Eventually, also pileup events are generated and superimposed on the hard scattering collision.

3.1 Monte Carlo truth information for MTD

During the Geant4 step simulated vertices, tracks and hits left in the detectors are saved in the event. The hits are then clustered together to obtain the simulated energy deposits to have all the elements that allow to design, optimize, and validate event reconstruction algorithms.

3.1.1 The MTD Truth Accumulator

The "truth accumulator" is a component within CMSSW, the CMS software, responsible for accumulating truth-level information related to a specific detector. A `CaloTruthAccumulator`, for hits deposited in the calorimeters, and a `TrackingTruthAccumulator`, for hits deposited in the tracker, were already integrated in CMSSW and have been used since the beginning of Phase 1. Between the tracker and the calorimeter in the simulation there is an imaginary boundary that separates the tracker volume from the calorimeter volume. Simulated tracks (`simTracks`) in the two regions are treated differently because saving all of them would be too expensive. The boundary sets the limit after which they are not saved anymore and hits produced in the calorimeters are associated with the last track that has crossed the boundary.

For the new MIP Timing Detector (MTD) a new truth accumulator is needed. As already said in the introduction, MTD has two different sub-detectors: the Barrel Timing Layer (BTL) and the Endcap Timing Layer (ETL). Unfortunately, even though the MTD simulated hits use the same C++ class as the tracker's hits, BTL lies in the tracker volume while ETL lies in the calorimeter volume, so they are treated in slightly different ways.

There are a couple of options that have been considered to address this situation: the first one was to write two different accumulators, but this would have led to two different objects, i.e. `TrackingParticle` for BTL and `CaloParticle` for ETL. Another option could have been to move the boundary after ETL, to have both sub-detectors in the same volume. Ultimately, the decision has been to use a similar approach as for the calorimeter for both sub-detectors. Therefore, the `MtdTruthAccumulator` takes inspiration from the `CaloTruthAccumulator`, but takes different objects in input and concentrates on the time instead of on the energy.

In the first step of the accumulator all the useful information is extracted from simulated hits (`simHits`) and saved for the next steps. These data are saved in maps where the keys are a combination of the detector ID (`DetId`) and the local position of a sensor (crystal for BTL and pixel for ETL) within a module in the form of a row and a column. The `DetId` alone is not enough for ETL because it is associated with a module, but each module contains 16x16 sensors. Therefore the keys are 64-bit integers where the first 32 bits store the `DetId`, the next 16 the row, and the last 16 bits the column. A map associates each key with the simulated hit time and if more than one `simHit` is left on the same sensor only the one with the smallest time is kept.

The next step is the creation of a decay graph of the simulated particles. When particles interact new simulated vertices are created and these correspond to the nodes, while the edges are the simulated tracks. This graph is used to create the so-called `CaloParticles` and `simClusters`. A `CaloParticle` is the particle that exits from the collision vertex, while the `simClusters` are created from the `simHits` left in the detectors.

In the last step of the accumulator `simClusters` are filled with the information extracted from the `simHits` at the beginning and then each `CaloParticle` is associated with the `simClusters` that originated from it.

The final result is two collections: one of `CaloParticles` and one of `simClusters`. The `simClusters` are the starting point to create new objects that are more suitable for all the subsequent studies.

3.1.2 MTDSimLayerClusters and MTDSimTracksters

Starting from the `simClusters`, two new collections have been produced: the `MTDSimLayerClusters` and the `MTDsimTracksters`. The `simClusters` contain all the `simHits` associated with the same `simTrack`, even if the particle has interacted creating other `simTracks` after the boundary or if it has done back-scattering in the calorimeters and crossed again MTD. This can happen in particular in ETL because all the `simTracks` created by a particle after it has crossed the boundary are not saved and are linked to the last `simTrack` that crossed the boundary. In addition, `simClusters` contain hits from both ETL layers, that are 4 cm apart. An example of what can happen is shown in Figure 3.1.

These objects are therefore very different from the clusters reconstructed by MTD. To have objects comparable with the reconstruction `MTDSimLayerClusters` have been created. They are obtained by applying to the `simHits` contained in a `simCluster` the same strategy used during the reconstruction. To belong to the same `MTDSimLayerCluster`, two hits must be in the same layer, module and be in adjacent sensors, so either the same row and adjacent column or the same column and adjacent row. These new objects will be employed to validate the current reconstruction algorithms and eventually to optimize them. In addition, they can be used to check the efficiency of linking the clusters to the reconstructed tracks. These studies are shown in Section 3.2.

The `MTDsimTracksters`, the other new collection, is more similar to the initial `simClusters`, but contains more information. Each `MTDsimTrackster` knows which `MTDSimLayerClusters` it contains and knows also the position and the time at which the particle entered for the first time MTD. These objects will be useful to study the link between HGCAL energy deposits and MTD clusters from the simulation point of view and for Particle Flow studies.

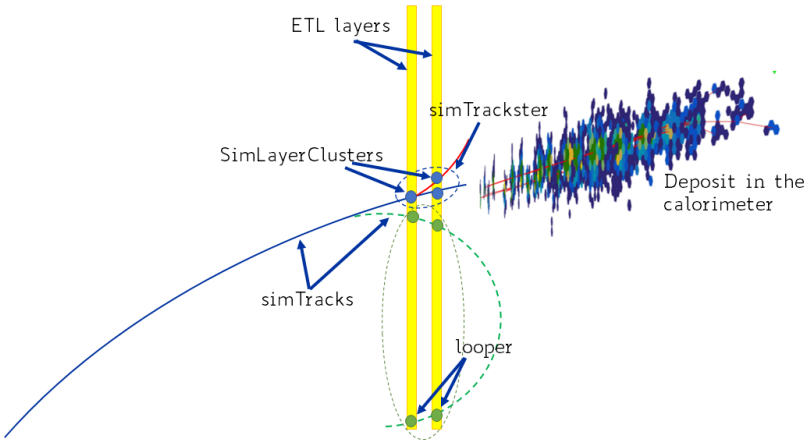


Figure 3.1: `MTDSimLayerClusters` and `MTDsimTracksters`: the simulated track (`simTrack`) enters MTD and leaves hits in the two ETL layers. New tracks may be created (e.g. the red one) and if they are after the boundary the hits are associated with the original `simTrack`. The particle can also do backscattering and go back to ETL (e.g. the green one) again. Hits associated with the same `simTrack` finish in the same `MTDsimTracksters` and then within a `MTDsimTrackster` they are divided into the `MTDSimLayerClusters`.

3.2 Studies on MTD efficiencies

Having Monte Carlo truth information of the hits and clusters left by the particles that crossed MTD, it is possible to check the efficiency MTD has in registering the passage of these particles, creating clusters, and assigning these clusters to a track.

This is done through the so-called associators, maps that link each reconstructed cluster with its corresponding simulated one and vice versa. The reco-to-sim associators have been realized by looping over the reconstructed clusters collection and associating each of them with a `MTDSimLayerCluster` on a hits matching base, namely looping on all their hits and searching for the `MTDSimLayerCluster` with the corresponding simulated hits. If no simulated cluster is found, the reconstructed one is probably noise. For the sim-to-reco associators, the reverse approach has been employed. If there was no corresponding reconstructed cluster for the simulated one, it indicates that this cluster was not reconstructed.

All the studies that follow have been done for pions and electrons and are focused on the Endcap Timing Layer (ETL).

The first check that can be done is indeed determining the cluster efficiency, i.e. the probability that a cluster has been reconstructed. To do so the sim-to-reco association map is used and the efficiency is computed as the number of correctly reconstructed clusters over the number of simulated clusters. The efficiency has been plotted as a function of p_{T}, η and ϕ for pions in Figure 3.2 and electrons in Figure 3.3 shot from the vertex. In both cases, the efficiency is more or less flat and is around 80%, a bit higher for pions. This can result from two distinct causes: either the rechits are not reconstructed or they are reconstructed but not clustered. Running the same plots to assess the rechits reconstruction efficiency revealed that the primary factor is the former, meaning that about 15% of hits are not reconstructed.

Another important study is the check of the correctness of the cluster time reconstructed by MTD. This can be easily done by comparing the reconstructed time with the simulated time of the associated `MTDSimLayerCluster`. The residuals are computed as:

$$\texttt{residual} = t_{sim} - t_{reco} \tag{3.2.1}$$

and as can be seen in Figure 3.4 they are centered in zero with a σ of 39 ps, in line with the average MTD resolution.

The next study concerns the link between MTD clusters and tracks. During the reconstruction, each track is propagated until MTD and then clusters are searched in a small region around the track. If a cluster is found, it is associated with the track and its time is propagated back to the point of closest approach to the vertex assuming the particle is a pion. For ETL, since there are two layers, up to two clusters can be associated with a track. If this happens the time of the second cluster is projected on the time of the first one and they are averaged weighted on the time error to obtain

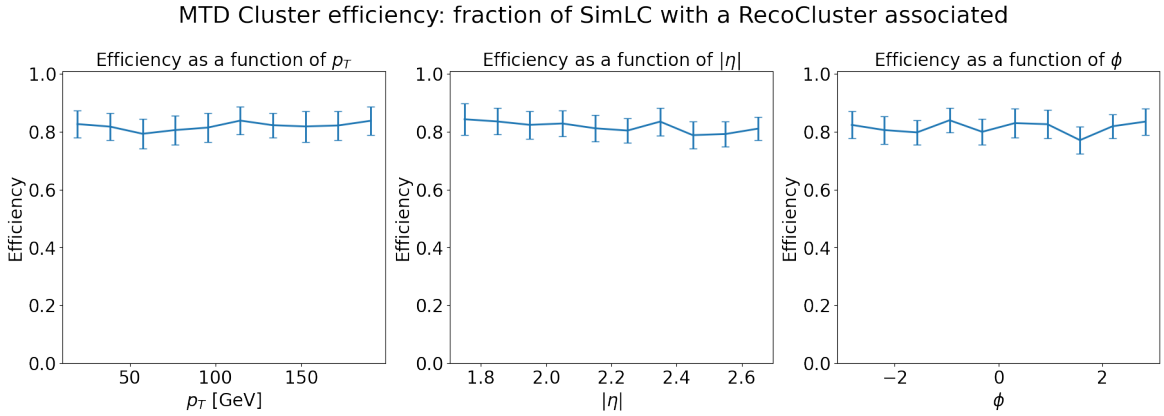


Figure 3.2: MTD clusters efficiency for pions: considering all the simulated layer clusters, the efficiency is the fraction of reconstructed clusters among all the simulated ones. The efficiency is shown as a function of the pion p_{T}, η and ϕ and is more or less constant and slightly above 80%

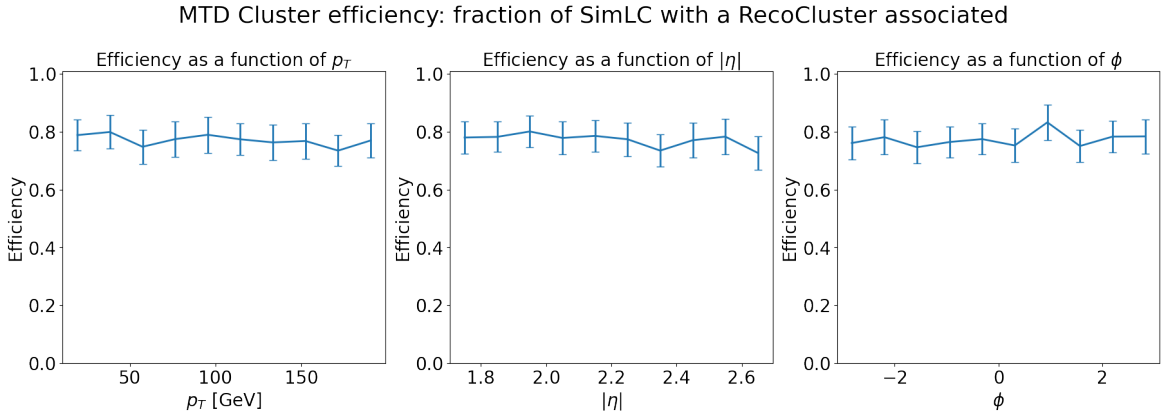


Figure 3.3: MTD clusters efficiency for electrons: considering all the simulated layer clusters, the efficiency is the fraction of reconstructed clusters among all the simulated ones. The efficiency is shown as a function of the electron p_{T}, η and ϕ and is more or less constant and slightly below 80%

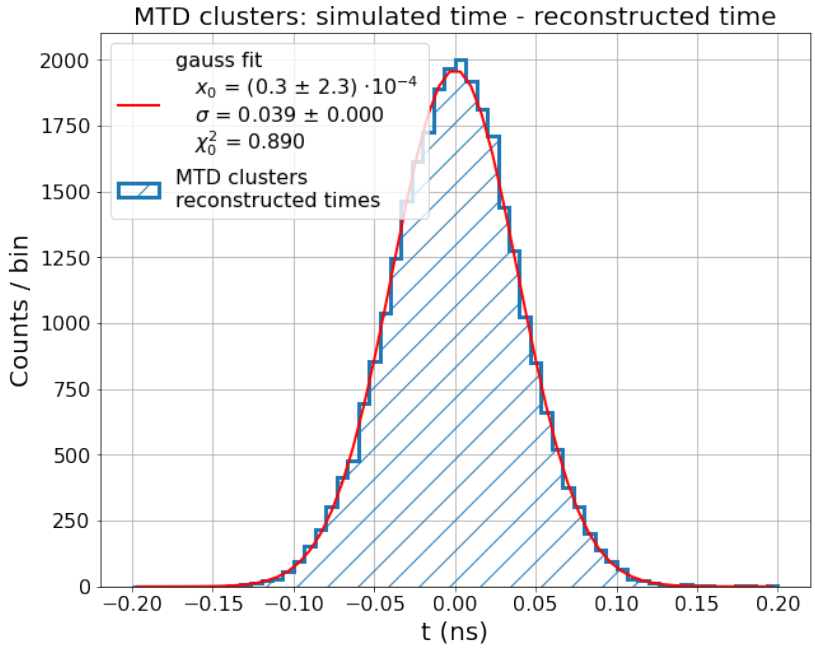


Figure 3.4: MTD clusters times residuals: the histogram shows the differences between the simulated and reconstructed times of the clusters and highlights a good cluster time reconstruction

only one time measurement for the track. Then, when reconstructing the vertices another iteration is done on the times, and other mass hypotheses are probed to find the one that gives the best fit with the vertex. The quality of the association of the time with the track is given by an MVA.

One cluster in ETL is enough to assign time to the track, but having two of them is better because the time estimation will be more accurate. For this reason, two different studies have been performed. In the first one, the efficiency is computed as the fraction of reconstructed tracks that have at least one MTD cluster correctly associated with them through the MTD reconstruction. For the second one instead the efficiency is computed by considering individually all the reconstructed clusters correctly associated with the tracks.

For each case, the efficiency has been computed by comparing the simulation with the reconstruction. Looking at Figure 3.5 there are two ways to link a `MTDSimLayerCluster` with its corresponding reconstructed cluster. The first one is with the sim-to-reco associators introduced previously, while the second one is through the MTD reconstruction. A `MTDSimLayerCluster` is associated with a `simTrack` that in turn can be associated with a reconstructed track. If the MTD reconstruction has linked a cluster with this track, it is possible to check whether or not the cluster is the same obtained directly with the sim-to-reco associator.

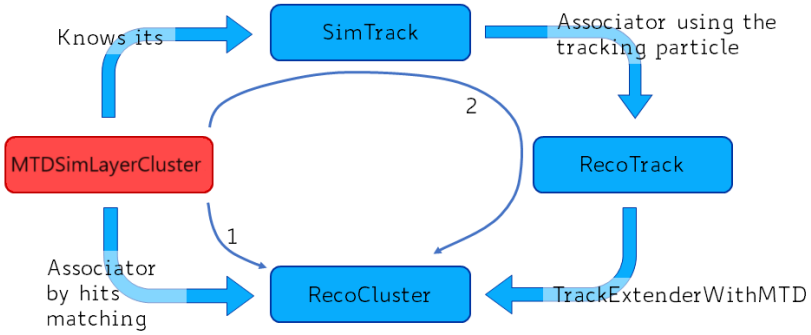


Figure 3.5: Association between a `MTDSimLayerCluster` and its corresponding reconstructed cluster (1) through the sim-to-reco associators and (2) through the `simTrack`, the associator `simTrack` to reconstructed track and the MTD reconstruction itself. If the clusters obtained with the two methods are the same the reconstruction was correct

Starting from the single pions shot from the vertex, Figure 3.6 shows the efficiency of associating at least one cluster to each track as a function of the pion p_{T}, η and ϕ. The light-blue one represents the efficiency using the sim-to-reco associators, computed as follows:

$$\mathtt{eff} = \frac{\begin{array}{c}\mathtt{simLayerCluster\ associated\ with\ a} \\ \mathtt{reconstructed\ track\ and\ to\ a\ reconstructed\ cluster}\end{array}}{\mathtt{simLayerCluster\ associated\ with\ a\ reconstructed\ track}} \tag{3.2.2}$$

The red points instead do the additional check that the cluster has to be correctly associated with the track and the efficiency is:

$$\texttt{eff} = \frac{\begin{array}{c}\texttt{simLayerCluster associated with a reconstructed track} \\ \texttt{that is linked to the correct reconstructed cluster}\end{array}}{\texttt{simLayerCluster associated with a reconstructed track}} \tag{3.2.3}$$

Each track is counted once even if there are more `MTDSimLayerClusters` associated with it.

The differences between the two curves are the clusters that have been reconstructed but not linked to the track and this is where the reconstruction algorithm can be optimized. The difference between the two curves is very small, meaning that the reconstruction algorithm is good at associating the clusters to the tracks.

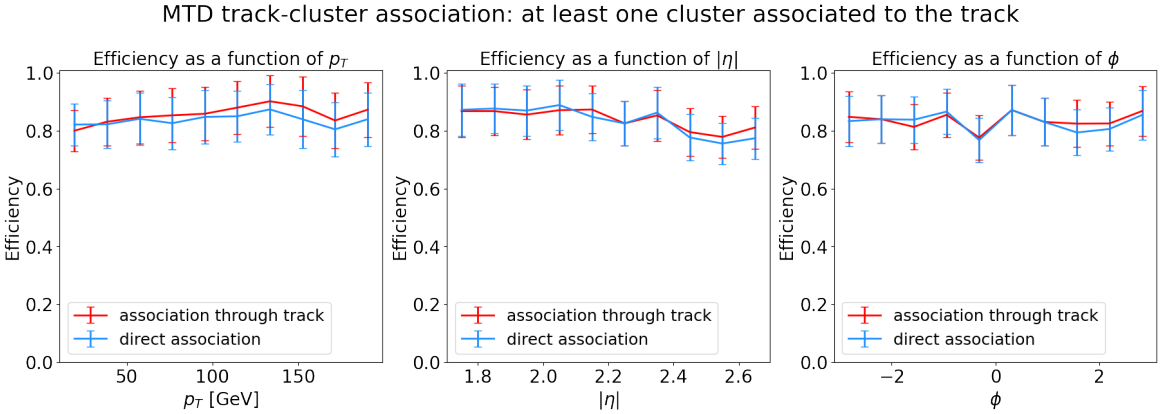


Figure 3.6: MTD clusters-track association efficiency for pions: considering all the reconstructed tracks, the efficiency in red is the fraction of reconstructed tracks that have at least one MTD cluster correctly associated with them through the MTD reconstruction. The efficiency in light-blue instead is the fraction of reconstructed tracks that have at least one MTD cluster associated with them through the simulation. The efficiency is shown as a function of the pion p_{T}, η, and ϕ and the two methods show compatible results, meaning that the reconstruction is performing well

Figure 3.7 shows the second study performed, the one that considered all the clusters individually. The efficiency definitions are the same for the light-blue (Equation (3.2.2)) and the red (Equation (3.2.3)) curves, with the difference that if there are more `MTDSimLayerClusters` associated with a track, it is counted more than once. In this case, the gap between the light-blue curve and the red curve is bigger, meaning that even if the reconstruction algorithm works well in assigning at least one cluster to the track, it's a bit worse in collecting all of them.

The same studies have been conducted on single electrons shot from the vertex. The results are shown in Figure 3.8 and Figure 3.9 and the trend with respect to pions

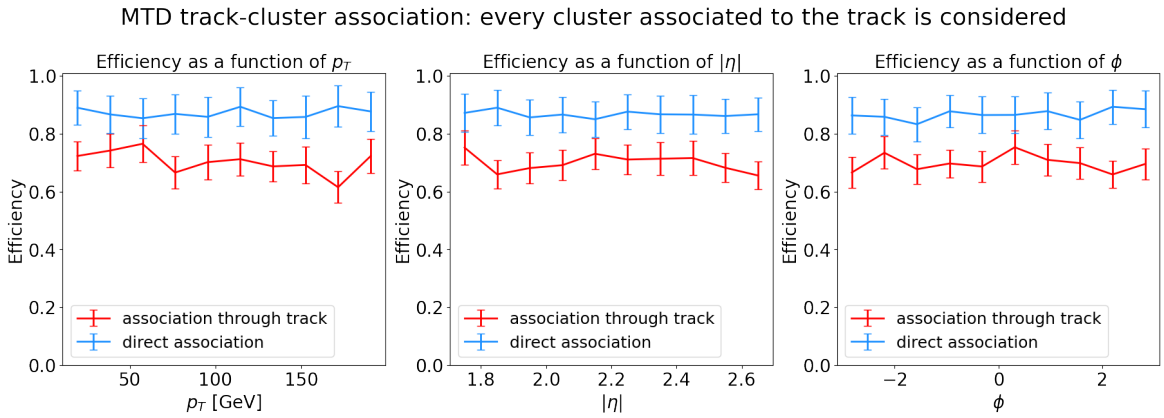


Figure 3.7: MTD clusters-track association efficiency for pions: considering all the reconstructed tracks, the efficiency in red is the fraction of reconstructed MTD clusters that have been correctly associated with the tracks through the MTD reconstruction. The efficiency in light-blue instead is the fraction of reconstructed MTD clusters that have been associated with the tracks through the simulation. The efficiency is shown as a function of the pion p_{T}, η and ϕ and the one obtained using the MTD reconstruction is slightly worse with respect to using the simulation

is overall worse. This is because electrons need a dedicated track reconstruction due to the different way they lose energy.

Charged particles can lose energy either through radiation or through ionization. The radiation process is called Bremsstrahlung and happens when a particle emits a photon exchanging a soft photon with a heavier nucleus nearby. The energy lost as a function of the distance traveled in the medium, i.e. the dE/dx, is a decreasing exponential with the distance at the exponent and is inversely proportional to the square of the mass of the particle. Instead, the energy loss by ionization happens when charged particles interact electromagnetically with the electrons of an atom in the medium and transfer energy to them in a process that results in ionizing the atom. The dE/dx is given by the Bethe-Block formula that depends on the speed of the particle and some properties of the medium. At high energy, the energy loss by radiation dominates, while the ionization becomes important at low energy. However, since Bremsstrahlung goes with the inverse square of the mass, it is relevant only for electrons.

As described in Section 2.2, when reconstructing tracks a combinatorial track finder based on the Kalman Filter is used [41]. It is employed iteratively for track fitting and predicts the expected state of a particle's trajectory based on its previous state and the known detector geometry. When actual measurements are obtained from the detector, the Kalman Filter updates the predicted state using the measured data, adjusting the estimate based on the consistency between prediction and measurement. However, the

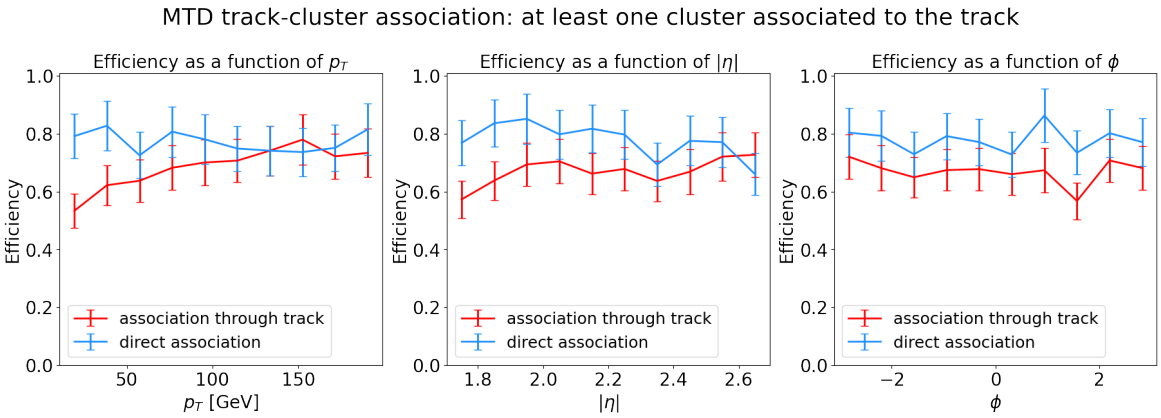


Figure 3.8: MTD clusters-track association efficiency for electrons: considering all the reconstructed tracks, the efficiency in red is the fraction of reconstructed tracks that have at least one MTD cluster correctly associated with them through the MTD reconstruction. The efficiency in light-blue instead is the fraction of reconstructed tracks that have at least one MTD cluster associated with them through the simulation. The efficiency is shown as a function of the electron p_{T}, η, and ϕ and the one obtained using the MTD reconstruction is slightly worse than using the simulation

Filter considers only the energy loss by ionization.

For the electrons, there is a dedicated track reconstruction performed using the Gaussian Sum Filter (GSF) [47] that models the various sources of uncertainty using a mixture of Gaussian distributions. It is more suitable for the electrons because it takes into account the trajectory changes due to Bremsstrahlung. At the end of the reconstruction each electron has two tracks associated with it, one using the Kalman Filter and the other using the Gaussian Sum Filter. At the moment the MTD reconstruction is performed only on the tracks obtained with the Kalman Filter, therefore the time associated with the electrons may be not optimal for this reason. A dedicated MTD reconstruction for electrons that uses GSF tracks could significantly improve the results, as can be seen from the light-blue lines in the plots that show the efficiency achievable with a perfect reconstruction.

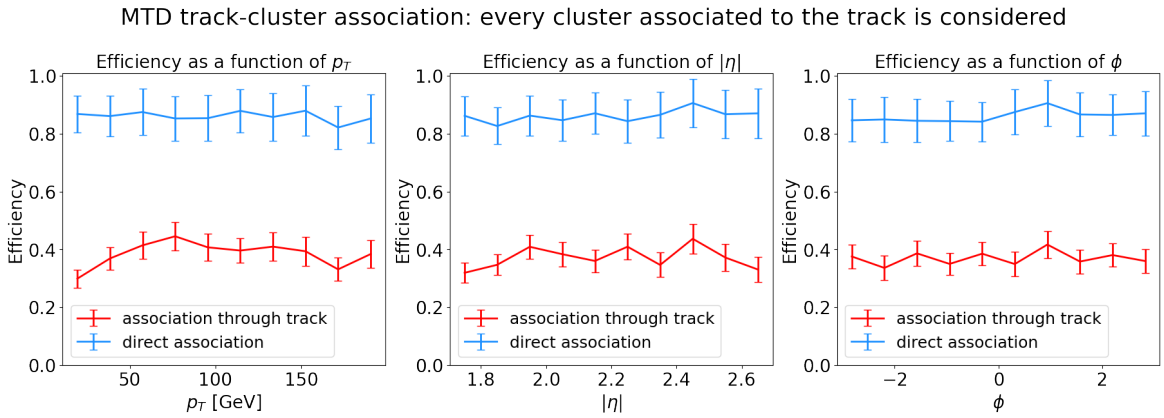


Figure 3.9: MTD clusters-track association efficiency for electrons: considering all the reconstructed tracks, the efficiency in red is the fraction of reconstructed MTD clusters that have been correctly associated with the tracks through the MTD reconstruction. The efficiency in light-blue instead is the fraction of reconstructed MTD clusters that have been associated with the tracks through the simulation. The efficiency is shown as a function of the electron p_{T}, η and ϕ and the one obtained using the MTD reconstruction is a lot worse than using the simulation

3.3 Monte Carlo truth information for HGCAL

The creation of the simulation level information for HGCAL is performed in the `CaloTruthAccumulator`. The processes that take place in the accumulator are described in Section 3.1.1 since, as already said, the `simHits` accumulation and the graph creation is similar between MTD and HGCAL. The output, as for the `MtdTruthAccumulator`, are two collections: the `CaloParticles` and the `simClusters`. These are the starting point for the creation of objects useful to check the reconstruction performance and optimize it.

3.3.1 HGCAL SimTICLCandidates

Starting from `CaloParticles` and `simClusters`, the so-called `simTracksters` and `simTICLCandidates` are created. They join information both from the simulation and the reconstruction to obtain objects that correspond to the best possible reconstruction and that can therefore act as a benchmark for the current one.

Two collections of `simTracksters` are created, one starting from the `CaloParticles` and the other starting from `simClusters`. Below there is a description of the `simTracksters` from `CaloParticle`, for those from `simClusters` the only difference is an additional mapping from the `simCluster` to its corresponding `CaloParticle`.

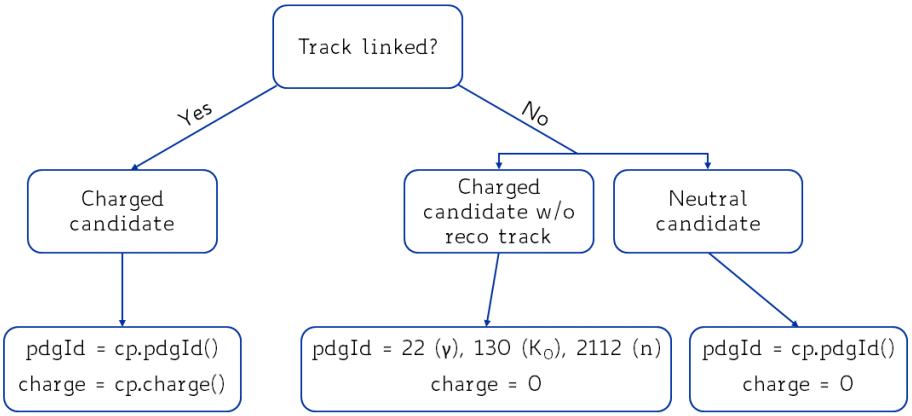


Figure 3.10: Procedure to assign `pdgId` and charge to each candidate: if there is a reconstructed track associated with it, it is a charged candidate and the `pdgId` and charge of its `CaloParticle` (cp) are used. If there is no track it can be either that the candidate is neutral or that is charged but the track was not reconstructed. In the former, the `CaloParticle` is used, in the latter the charge is set to zero and the `pdgId` is set to that of a photon, Kaon, or neutron depending on the original particle

For each `simTrackster` the associated `simTrack` is taken, and through the `simTrack` to `recoTrack` associator, the corresponding reconstructed track is obtained, if existing. The associator is based on hits matching, meaning that the `recoTrack` that shares the highest number of hits with the `simTrack` is retrieved, provided that the shared fraction is at least 75%. If no reconstructed track is found, it is changed to a neutral candidate, and its `pdgId` is adjusted accordingly, as illustrated in Figure 3.10. If the original particle was an electron, it becomes a photon, while light hadrons become Kaons and heavier hadrons become neutrons.

The corresponding `CaloParticle` is saved as well in the `simTrackster` and its energy is set as the trackster's energy. In addition to that, also the particle ID is stored in the trackster. Finally, associators between `CaloParticles` (or `simClusters`) and the layer clusters are used to assign to each simTrackster its reconstructed layer clusters.

The `simTracksters` are used to build the `simTICLCandidates`. Each `simTICLCandidate` contains all the `simTracksters` coming from the same `CaloParticle`, therefore there will be a one-to-one map with the `simTracksters` from `CaloParticle` and a one-to-many map with the `simTracksters` from `simCluster`. Each `simTICLCandidate` has all the information of the `simTracksters` it contains and, in addition to that, it is also possible to add information from other sub-detectors. For example, for those that have a track associated with them, it is possible to add the corresponding `MTDsimTrackster` to have the time associated with

the track. An example is shown in Figure 3.11, where a photon produced during a collision converts in a positron-electron pair through pair production, and the two charged leptons, after crossing MTD, enter HGCAL and leave there all their energy.

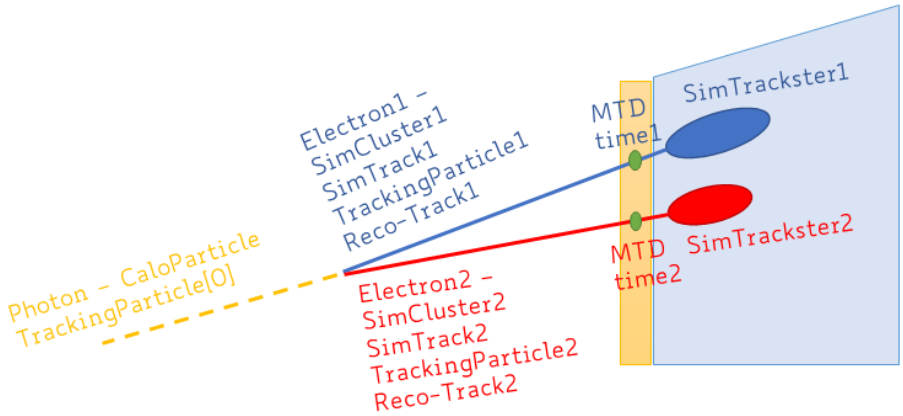


Figure 3.11: Example of a photon that does pair production and creates an electron-positron pair. Each of the leptons leaves hits in MTD that will be reconstructed as clusters and associated with the tracks and then creates an energy deposit in HGCAL, corresponding to a simTrackster from `simCluster`. Each energy deposit is linked with the MTD cluster and the reconstructed track and then they are associated with the original photon to form the final `simTICLCandidate`

Each electron track, together with its MTD cluster and HGCAL trackster becomes a `simTrackster` (from `simCluster`), while all together, also with the original photons, form the `simTICLCandidate` corresponding to that photon and associated with the photon's `CaloParticle`.

The `simTracksters` and `simTICLCandidates` are fundamental to check the correctness of the pattern recognition and linking algorithms and to validate the linking between tracks and tracksters.

Chapter 4

A time-aware reconstruction for the Phase 2 upgrade of CMS

The goal of this chapter is to use the time information provided by two CMS sub-detectors, HGCAL and ETL, to perform a time-aware linking between the tracks, that have a time from MTD associated with them, and the tracksters. The quality of the time information provided by MTD and the efficiency in associating it to a track has been discussed in Section 3.2. The next section will describe the way the time information is treated in the HGCAL reconstruction.

4.1 Studies on the time information in HGCAL

As discussed in Section 1.4.2, the High-Granularity Calorimeter (HGCAL) in addition to a fine spatial granularity provides precise time information to help unveil the details of particle interactions.

4.1.1 Reconstructed hits

When a particle enters HGCAL and initiates an electromagnetic or hadronic shower, it interacts with the material of the absorbers and generates energy deposits in the active sensors. These energy deposits translate into hits, each carrying a timestamp that denotes the moment of interaction. Notably, the accuracy of the recorded time information is inherently tied to the energy of the deposited particles. Higher-energy particles induce more substantial signals in the sensors, leading to more precise time measurements. The threshold for triggering the registration of time of arrival is set at a minimum charge deposit of 12 fC, which corresponds to approximately 3 Minimum Ionizing Particles (MIPs) traversing the 300 μm silicon sensors. As the

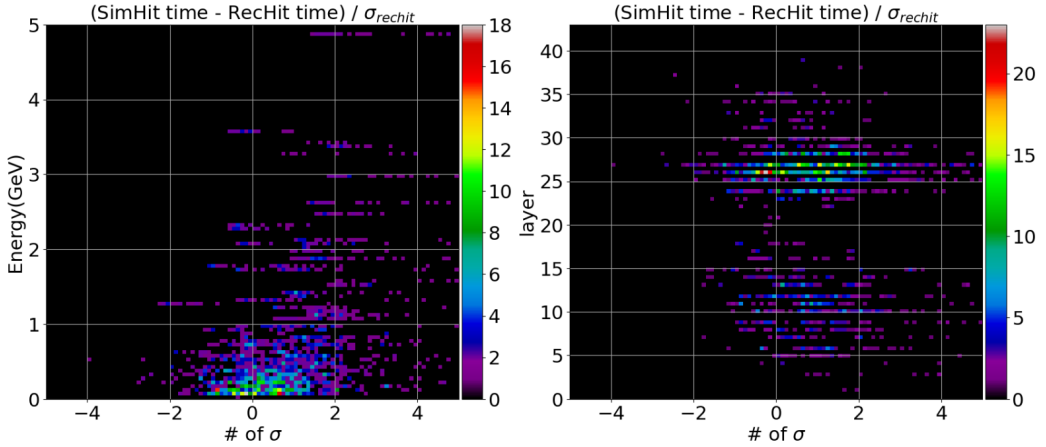
energy deposition surpasses this threshold, the time measurement becomes increasingly refined, allowing for more detailed insights into the shower development.

To check the accuracy of the time associated with the reconstructed hits, simulated pions, both charged and neutral, have been shot from the vertex. Subsequently, the residual between the simulated and reconstructed hits times have been computed, revealing deviations between the reconstructed and simulated time values. Consequently, to understand the reasons for these deviations, the residuals have been plotted against the rechits energy and their HGCAL layer number. These plots, in Figure 4.1, show a clear relation between the energy of the rechits and the magnitude of the temporal shift. As the energy of the rechits increases, so does the extent of the observed temporal discrepancy.

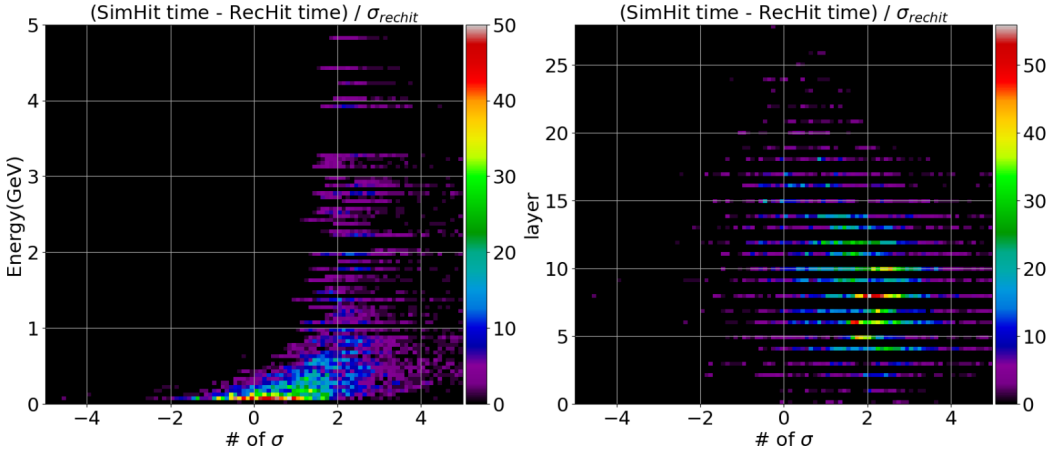
This could be related to an imperfectly addressed aspect known as time walk. Time walk correction represents a critical adjustment applied to the signals generated by particles within the detector. The time measurement of the particle is performed when a certain amount of charge is accumulated in the sensor, but the response time of the detection system can differ depending on the energy of the incoming particle. Particles with higher energy deposits might have a steeper rise of the signal and trigger the detection system slightly faster than particles with lower energy deposits.

Thus, the observed correlation between the energy of the rechits and the associated temporal shifts can be attributed to a not-perfect time walk correction. It is plausible that the time walk correction, despite its application, might not be finely calibrated or adequately applied, leading to residual timing shifts that co-vary with the energy of the rechits. Of particular note is the intensification of this discrepancy as the energy of particles escalates. As of now, no additional corrections have been introduced to the rechits' time values. Nonetheless, it is important to address this issue in future studies.

In the current CMSSW implementation, the time of the rechits is propagated back to the origin $(0, 0, 0)$ at the speed of light to obtain the time of the energy deposit at the vertex. However, many approximations in this approach lead to non-negligible errors. The propagation at the speed of light is correct only for photons, using a straight line is not correct for charged particles, especially at low energies, and the actual collisions vertices are spread in a window of about 10 cm along the beam axis. All these factors increase the possibility of committing errors that can result in too large uncertainties with respect to the desired time precision. The beam spot width in time units is ~ 200 ps, therefore additional uncertainties of the order of O(10 ps) can render the time information useless or wrong. An example of the entity of these errors is given in Figure 4.2, where the time at the vertex of a proton of 50 GeV at $\eta = 2$ would have been slightly different doing the propagation at the speed of light, but substantially different if the vertex were 10 cm apart. To avoid such problems that would require inflating the errors on the time of the rechits, it has been decided to remove the propagation to the origin and to keep the local time in the detector for



(a) Residual (# of σ) between simhits and rechits time as a function of the rechit energy and of the HGCAL layer for charged pions



(b) Residual (# of σ) between simhits and rechits time as a function of the rechit energy and of the HGCAL layer for neutral pions

Figure 4.1: Residuals between simhit and rechit time as a function of the rechit energy and of the HGCAL layer number for charged and neutral pions with an energy of 100 GeV. The shift is evident where the residuals are plotted as a function of the energy and increases with increasing energy. There is no evident correlation with the HGCAL layer

the current study.

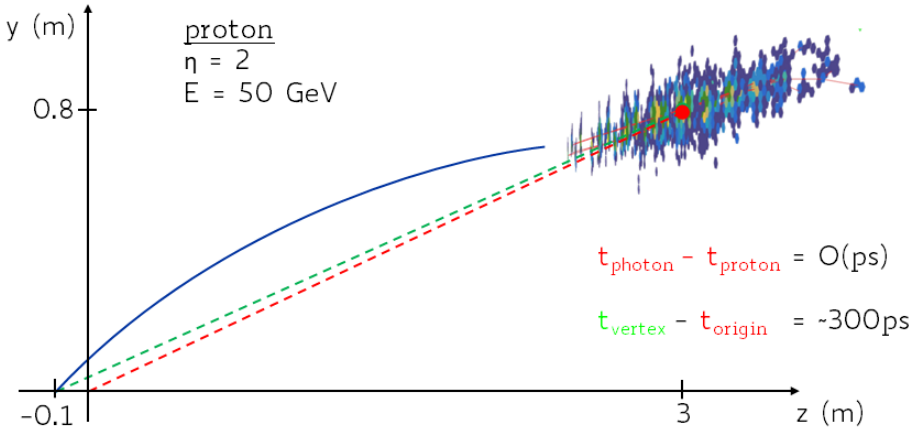


Figure 4.2: An example of the entity of the error done on assigning a time at the vertex to a proton of 50 GeV at $\eta = 2$. Using the speed of light induces a small error, while using the wrong vertex for the propagation would result in a non-negligible effect, even bigger than the beam spot width.

4.1.2 Layer clusters

The reconstructed hits serve as the primary input for CLUE 2D, the innovative clustering algorithm designed to create clusters on each HGCAL layer through a sophisticated density-based scan that takes into account also the energy associated with each hit. The clustering process developed in CLUE has been described in Section 2.3.1.1.

Once the layer clusters are established, additional information, i.e. time, comes into play. For a layer cluster to be endowed with a temporal attribute, it must be composed of a minimum of three reconstructed hits that possess time information.

The temporal value assigned to each layer cluster is derived through a calculation that emphasizes accuracy. The hits within a cluster contribute to this calculation with a weight corresponding to their timing precision, computed as the inverse square of their respective error values. This weighted approach ensures that hits with greater timing certainty hold more influence in determining the average time of the cluster.

To improve the accuracy, an additional constraint is applied to this process: all the hits used must be within a time span of 210 ps, corresponding to the expected time spread of the beam spot. To finalize the calculation, the hits are ordered chronologically and the set of hits that maximizes the number of hits within the specified time window is selected to compute the average time. This approach ensures that the most densely populated temporal region of the cluster contributes to the final calculated time.

Figure 4.3 shows a single simulated electron of 100 GeV shot from the vertex in the endcap. The electron crosses the Endcap Timing Layer (ETL) of MTD before showering in HGCAL. The green points are the clusters reconstructed by MTD that will later be associated with the track. The light-blue circles instead are the layer clusters with a time associated with them.

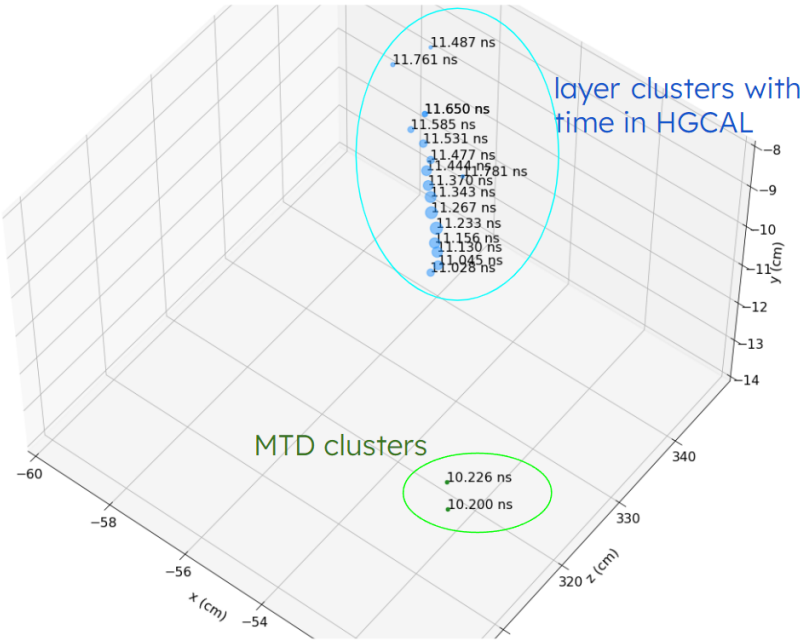


Figure 4.3: Single electron of 100GeV from the vertex: only the layer clusters with a time associated are plotted in light blue. The reconstructed MTD clusters associated with the electron are shown as well

Regarding the layer cluster times and errors, it is interesting to see the latter as a function of the energy, as shown in Figure 4.4. The plot shows a clear relation between the two, with more energetic clusters having more precise time information. The plateau for high energy clusters is around 5 ps, which is quite small and may be an underestimation of the errors.

4.1.3 Tracksters

Moving forward, the subsequent phase of TICL involves pattern recognition, which facilitates the linkage of layer clusters across various layers, resulting in the formation of tracksters. Each trackster necessitates a corresponding time assignment.

The original approach was to perform directly a weighted average of the times of the layer clusters in the trackster because their time was already extrapolated at the origin $(0, 0, 0)$. However, as explained in the previous section, such a time is very prone

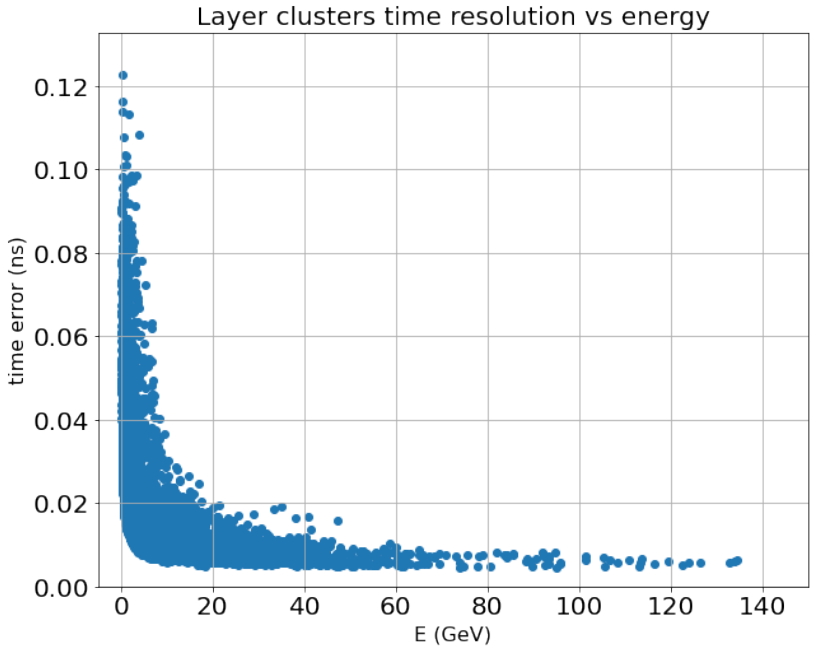
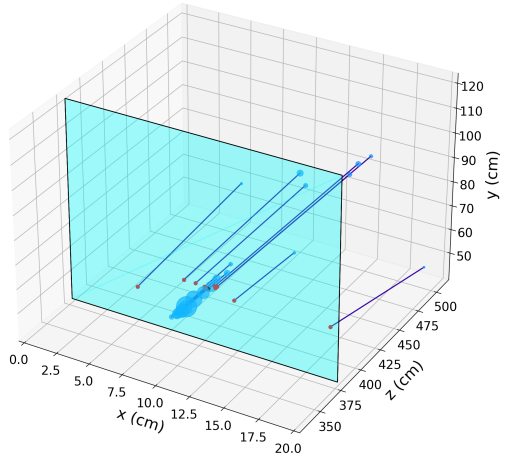
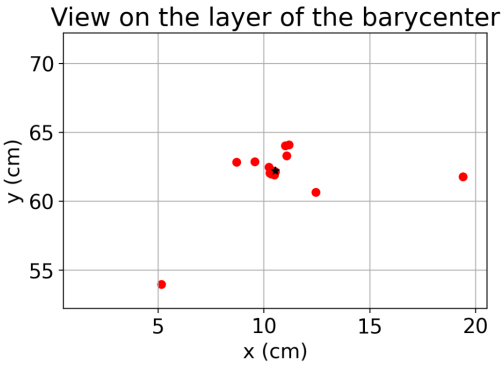


Figure 4.4: Layer clusters time errors as a function of their energy. There is a clear relationship between the two, with higher energy clusters having a more precise time. This is inherited from the rechits times and errors



(a) 3D visualization



(b) 2D projection on the HGCAL layer where the barycenter of the trackster lies

Figure 4.5: Trackster time computation: each trackster is composed of many layer clusters and those with enough energy have a time associated with them. All the layer clusters are connected in a straight line to the origin and the time is propagated to the interception of this line with the x-y plane where the barycenter lies. To obtain the trackster time, a weighted average of the times of the layer clusters is performed

to errors. The choice for the tracksters as well has been to compute their time locally, reducing the propagation of times to the minimum.

To compute the trackster time all the layer clusters are connected in a straight line to the origin and the time is propagated to the interception of this line with the x-y plane corresponding to the HGCAL layer where the barycenter is located. To obtain the trackster time, a weighted average of the propagated times of the layer clusters is performed. A visualization of this process is in Figure 4.5. For future studies, looking at Figure 4.5b an optimization can be done by selecting a small region around the barycenter and taking into account for the average only the clusters whose projection falls in this area.

4.2 Using the time information in track - trackster linking

4.2.1 Linking algorithm

The goal of the linking is to establish connections between tracks and tracksters [48]. To simplify the problem both tracks and tracksters are projected onto a common surface. Subsequently, the algorithm endeavors to identify geometric neighbors on this surface.

Once the graph of possible links is established, an iterative exploration is undertaken, guided by considerations of energy and time compatibility, to create the final TICL candidates. To visualize the challenge addressed here, the representation in Figure 4.6 can be used. It will be important also later in the linking efficiency definition. It shows a reconstructed track and its associated trackster, together with the `simTrack` and `simTrackster` from the MC simulation. The reconstructed and simulated tracks are associated with a map based on hits matching, while the map that associates simulated and reconstructed tracksters is based on the fraction of energy shared. The goal is to link the track and trackster as shown by the red arrow in the figure and the simulated information helps in determining the correctness of the link.

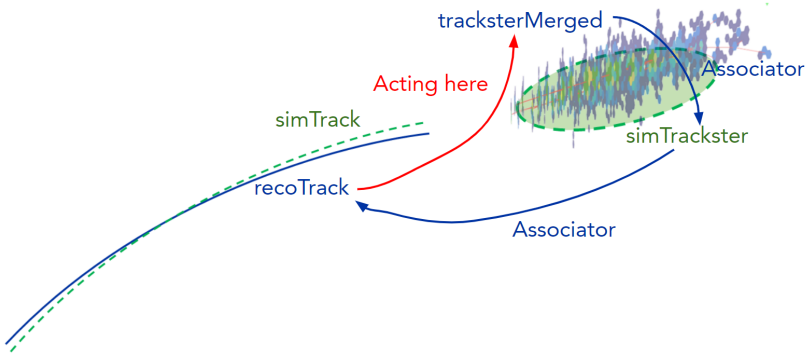


Figure 4.6: Schematic representation of a track and a trackster that should be linked together. The reconstructed track is associated with a simulated track that, in turn, is associated with a `simTrackster`. A specific map allows to associate the `simTrackster` to the reconstructed trackster and vice versa and can be used to check the correctness of the linking. The elements corresponding to the MC simulation are represented with dashed lines

Two surfaces have been selected for the linking: the first HGCAL layer and the interface between the CE-E and CE-H, namely the last CE-E layer. The propagation to the two surfaces is done differently for tracks and tracksters:

- the track propagation is done taking into consideration material-induced effects like multiple scattering and energy loss and accounting for bending due to the magnetic field. Only the tracks above 1 GeV, with high purity[1], less than five missing hits in the Outer Tracker and a valid propagation to HGCAL are considered as possible candidates for the linking;
- for the tracksters two approaches have been considered: a linear extrapolation of the barycenter (the energy-weighted average of layer-cluster positions) back to the origin $(0, 0, 0)$ or using the trackster's direction derived from the Principal

[1]requirements on the track's impact parameters and on the number of layers of the Tracker where the track left hits are used to categorize tracks. The selections used are listed in [49]

Component Analysis (PCA) of its layer clusters. The former is used because the PCA direction can be very unreliable, especially for small tracksters that do not have a well-defined direction.

A cut on the tracks energy is applied and tracks below 2 GeV are not considered for linking.

The linking phase starts with a surface and a seeding collection, either tracks or tracksters, projected to that surface. A $\eta - \phi$ window around each seed is opened and the tracksters which propagation falls in this window are sorted based on their distance to the seed and considered as possible links. After this stage, four distinct collections of possible links are generated:

- tracks with tracksters at the first HGCAL layer;
- tracks with tracksters at the last CE-E layer;
- electromagnetic tracksters, i.e. tracksters whose barycenter is in the CE-E, with hadronic tracksters, i.e. tracksters whose barycenter is in the CE-H, at the last CE-E layer;
- hadronic tracksters with other hadronic tracksters, propagated to the last CE-E layer.

The link exploration starts from charged candidates, i.e. those associated with tracks, and follows the procedure highlighted in Figure 2.4. For every track, the track-to-trackster collection at the first HGCAL layer is used to identify any linked tracksters. Each trackster discovered as a link here is subsequently employed to probe the electromagnetic-to-hadronic trackster linking and the hadronic-to-hadronic trackster linking at the last CE-E layer, using a depth-first approach. The same procedure is applied to the tracks to trackster linking directly at the last CE-E layer and continues in the same way as before.

Each track can be linked with more than one trackster, with an energy and time compatibility constraint. Tracksters can be added to the track until the total energy is less than the track's momentum plus a threshold that depends on the energy of the tracksters. Time compatibility checks that the difference between trackster and track times at the vertex is less than 3σ, with σ being the sum of the errors of trackster and track squared. However, as already said, the time compatibility at the vertex is not optimal and will be changed later. Tracks not linked with tracksters form candidates by themselves, the so-called empty candidates.

The creation of neutral candidates mirrors the aforementioned process, except for the fact that it does not involve tracks, as shown in Figure 2.5. The tracksters linked during the charged candidates' creation are not used at this step.

The procedure described is the starting point for the changes applied to the linking. The modifications that do not involve time are the following:

- the constraint on the energy on the track being at least 2 GeV has been removed because the linking algorithm takes as input tracks down to 1 GeV in p_T and there is no reason to remove them;
- tracks and tracksters are sorted by energy to link firstly the most energetic ones. Pileup tracksters are expected to be less energetic and likely will not be linked due to energy compatibility if huge tracksters are linked beforehand;
- a trackster is marked as hadronic not only if the probability of being an electron or a photon given by the convolutional neural network (CNN) used to do particle identification is less than 50%, but also if the energy deposited in the CE-E is less than 85% of the total energy;
- when a trackster marked as hadronic is linked to a track or another trackster, the candidate is forced to be hadronic. Without this, some pions with a relevant energy deposit in the CE-E were marked as electrons by the CNN.

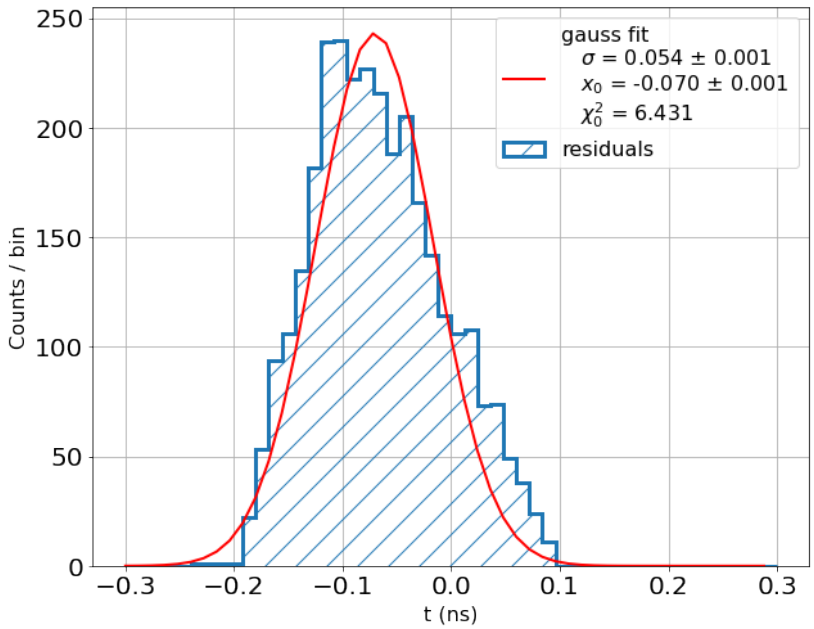


Figure 4.7: Plot of the residuals between the time of the tracks and of the tracksters linked together. There is a shift to the left of 70 ps that will be investigated and corrected by applying the same shift to the time of the trackster

The additional change, that affects only charged candidates and only the track-trackster linking, is the time compatibility one. The original version checks for compatibility at the vertex, with all the issues already described. The new approach is

to perform it with the local time in MTD and the trackster time computed with the procedure described in Section 4.1.3. To compare them, it is necessary to bring them to the same point in space. The speed used in the propagation is the one provided by MTD. When MTD associates a cluster to a track, it propagates the cluster time to the vertex following the trajectory and probes different mass hypotheses, choosing the one that gives the best time fit at the vertex. Knowing the mass and the energy of the particle, it is possible to obtain the speed of the particle $v = \beta c$. With this, the time compatibility is computed as follows:

$$\texttt{residual} = \frac{\texttt{distance}(\vec{x}_{track}, \vec{x}_{trackster})}{\beta c} - (t_{trackster} - t_{track}) \tag{4.2.1}$$

and the threshold has been computed as:

$$\texttt{threshold} = 3 \cdot \sqrt{\sigma^2_{track} + \sigma^2_{trackster}} \tag{4.2.2}$$

with σ being computed as the sum of the squares of the errors and asking for a 3σ compatibility. The error on the time of the trackster can be very small for high-energetic tracksters, up to O(1 ps), leading to an underestimation of the real error and a smaller threshold, resulting in a higher probability of rejecting good linking. For this reason, the threshold has been set as:

$$\texttt{threshold} = 3 \cdot \sqrt{2}\sigma_{track} \tag{4.2.3}$$

using the error on the time of the tracks also as the trackster time error.

Considering a sample of pions from the vertex, the residuals of Equation (4.2.1) are shown in Figure 4.7 and highlight a time shift of 70 ps with respect to zero. This shift has been observed, less evidently, also for electrons shot from the vertex. Several possible reasons have been considered for the observed shift and can partially explain it:

- reconstructed hits times differ from the simulated times and the difference becomes more evident with increasing energy. However, using the simulated times the shift does not disappear, so this cannot be the only reason;
- propagation of the layer clusters times using the speed of light. It is not guaranteed that shower development happens at the speed of light. A dedicated study on the speed of shower development should be done to be able to estimate the correct one to use for the propagation;
- for the time compatibility between tracks and tracksters the β computed during the MTD reconstruction is used. However, as for the previous point, it may overestimate the actual speed of the particle inside HGCAL. Using the simulated

times and imposing the time compatibility it is possible to obtain an estimation of the β of the particle. A preliminary study has been done, but to be more accurate it should be done as a function of many parameters such as the particle ID, its energy, η, and the HGCAL layer the barycenter lies on;

- calibration of the times reconstructed by the different detectors: it is not guaranteed that the "zero" of MTD corresponds to the "zero" of HGCAL and it may be that a global time calibration is needed.

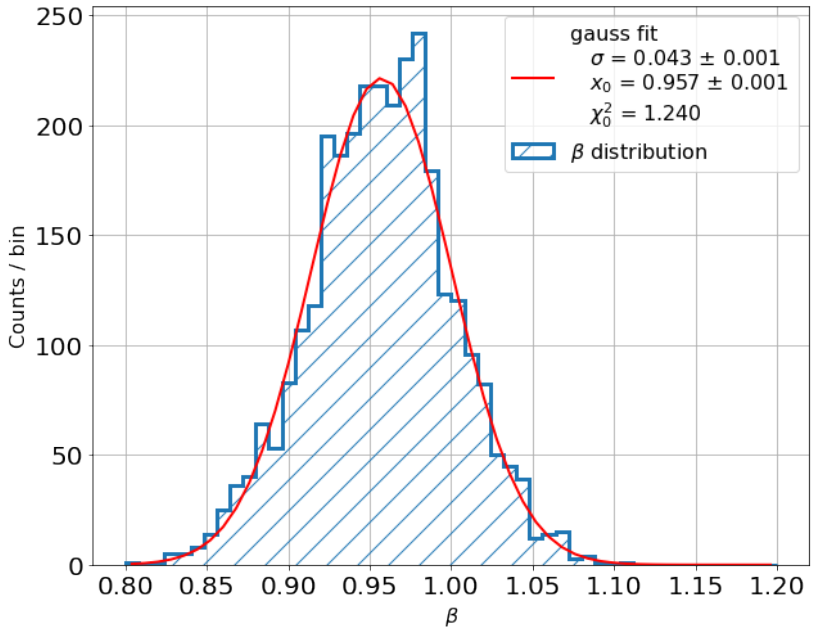
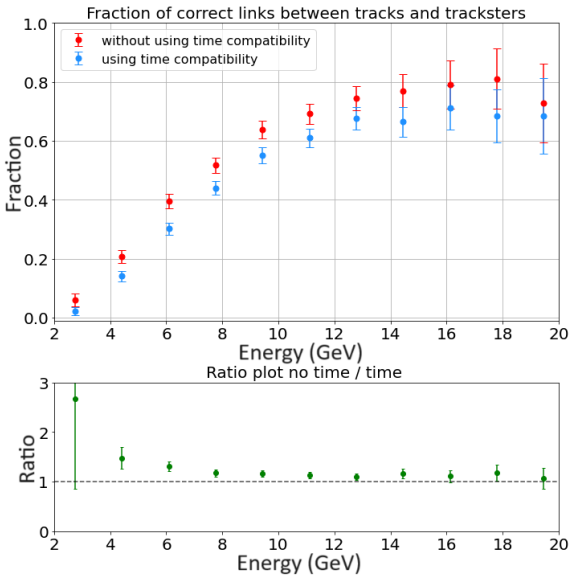


Figure 4.8: Distribution of the values of β obtained forcing the residuals of Equation (4.2.1) to be zero. The average value, 0.96, is the typical value of a MIP

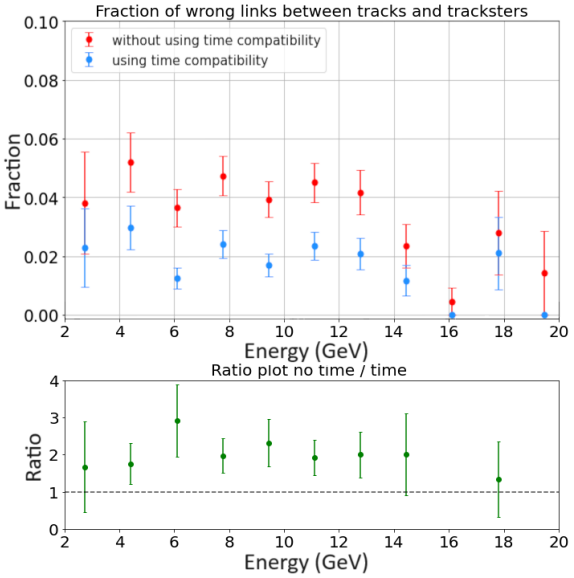
As a first approach, the decision has been to apply a fixed shift to the time of the trackster to match the average of the distribution with zero. Forcing the residuals to be zero allows us to obtain an estimate of β for the time propagation. This study has been done and the results, shown in Figure 4.8, highlight an average value for β of 0.96, the typical value of a MIP. However, further studies will be required to better understand all the causes of the shift.

4.2.2 Results for single pions

The developed time-aware linking has been tested on a sample of simulated pions shot from the vertex with energy from 5 to 20 GeV, within a cone with a radius of 30 cm on the first HGCAL layer and separated by a 100 ps interval.



(a) Fraction of correct links between tracks and tracksters for pions shot from the vertex with an interval of 100 ps in a radius up to 30 cm with energy from 5 to 20 GeV with (light-blue) and without (red) the time compatibility constraint. Using the time compatibility, the number of correct links is reduced, but that's acceptable considering the bigger improvement in the fake links reduction



(b) Fraction of wrong links between tracks and tracksters for pions shot from the vertex with an interval of 100 ps in a radius up to 30 cm with energy from 5 to 20 GeV with (light-blue) and without (red) the time compatibility constraint. Using the time compatibility, the number of wrong links is reduced

Figure 4.9: Correct and wrong links fractions for charged pions close in space (cone with a radius of 30 cm on the first HGCAL layer) and separated by a 100 ps interval

Since the pions are close in space the geometric linking may connect the track of a pion with the trackster of another one. The time-aware linking instead should be able to reject those links because they are incompatible in time.

To check the efficiency of the linking, the validation procedure starts from the simulated tracks and can be visualized in Figure 4.6. For each simulated track, the associated reconstructed track is retrieved and the candidate it belongs to is selected. If the candidate contains tracksters it can be associated with a `simTrackster`. The correct `simTrackster` is known from the simulated track, therefore it can be easily checked if the two `simTracksters` correspond or not.

Different categories are considered and summarized in Figure 4.10: for each track there can either be or not a trackster linked. If the trackster is present, a comparison with the simulation is done to check whether or not the link is correct. When there is no trackster, the inefficiency is split between the case where the trackster is there but the linking algorithm fails and the case where the trackster is not reconstructed.

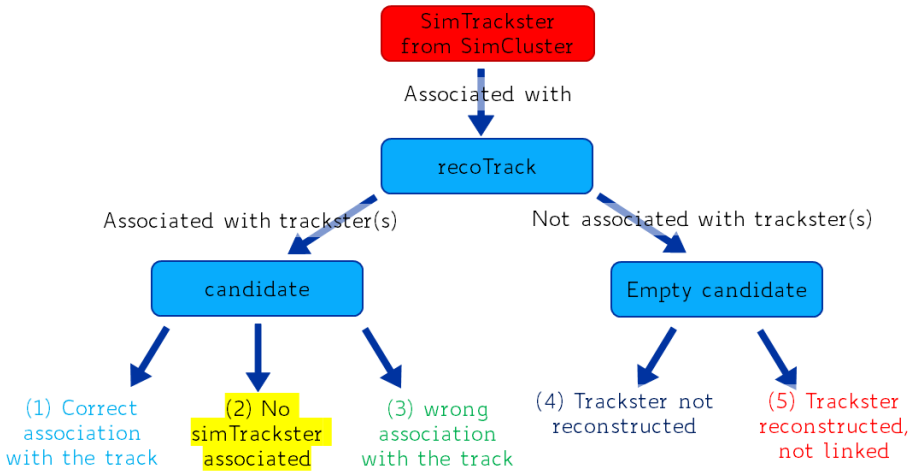
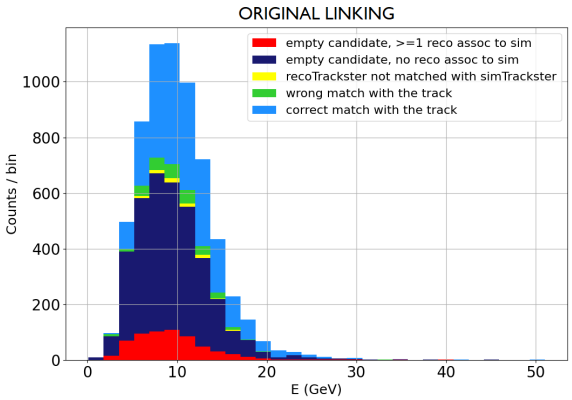


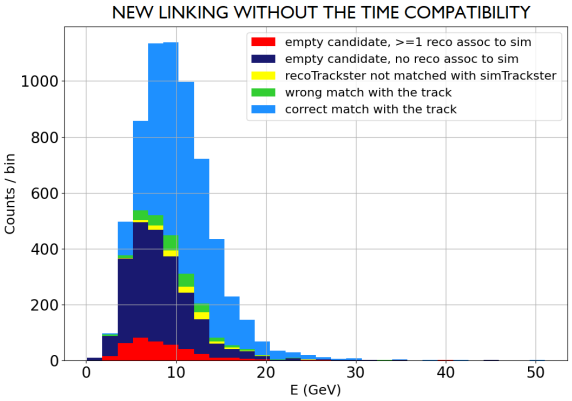
Figure 4.10: Different categories considered when computing the linking efficiency: for each track there can either be or not a trackster linked. If the trackster is present, a comparison with the simulation is done to check whether or not the link was correct. When there is no trackster, the inefficiency is split between the case where the trackster was there but the linking algorithm failed and the case where the trackster was not reconstructed.

The efficiency is then computed as the number of correct links between tracks and tracksters over the number of candidates coming from tracks that pass the same cuts used during the linking. i.e. $p_{\mathrm{T}} > 1$ GeV, high purity, and less than five missing outer hits. The results are shown in Figure 4.9a. The fake fraction has also been computed, with the same denominator as above and the number of wrong links at the numerator, and the results are shown in Figure 4.9b. The two, correct and wrong links, do not add up to one. The remaining part is the fraction of empty candidates, i.e. the candidates composed only of tracks.

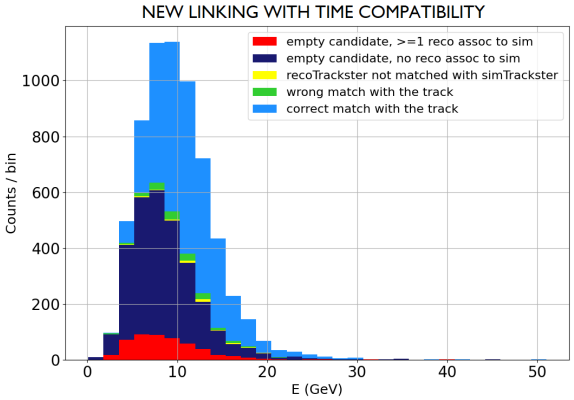
Looking at these plots, the constraint on time compatibility appears to bring an efficiency reduction of approximately 10%. On the other hand, this reduction in efficiency is counterbalanced by a substantial decrease in the fake rate, which is diminished by a factor of two or more. This trend offers a promising indication that time compatibility could be useful in the context of improving the linking. In the near future, the effort will be directed to recovering the efficiency fraction that has been lost, while keeping the wrong links low, to obtain an overall better linking.



(a) Stacked histogram of the linking efficiency for the original linking



(b) Stacked histogram of the linking efficiency for the new linking without time compatibility



(c) Stacked histogram of the linking efficiency for the new linking with time compatibility

Figure 4.11: Using the definitions of Figure 4.10 it is possible to break down all the inefficiencies of the linking. The light blue is the fraction of correct links; the green is the fraction of wrong links; the yellow corresponds to the fraction of reconstructed tracksters not matched with a `simTrackster`, probably due to a not-good reconstruction or a merge of deposits from different particles; the red is a missed link with a trackster was reconstructed and the dark blue is a missed linked with a trackster that was not reconstructed (or the reconstructed trackster is not good enough). These plots highlight how the changes done to the linking (b) improve its efficiency with respect to the original version (a) and what is the effect of the usage of linking (c). The histograms are stacked

To conclude, a comparison between the original linking and the new linking, with and without time compatibility, is presented in Figure 4.11 with a stacked histogram of the different outcomes of the linking. The colors used correspond to those of Figure 4.10: the light-blue is the fraction of correct links, the green is the fraction of wrong links, the yellow corresponds to the fraction of reconstructed tracksters not matched with a `simTrackster`, probably due to a not-good reconstruction or a merge of deposits from different particles; the red is a missed link with a trackster that is reconstructed and the dark blue is a missed linked with a trackster that is not reconstructed (or the reconstructed trackster is not good enough). These plots highlight the improvement in efficiency between the original linking and the new one while for the new linking, with and without time compatibility, underline the same trend observed before.

Chapter 5

Use of timing in electron isolation

After the studies on the linking between tracks and energy deposits, this chapter addresses another topic of event reconstruction, electrons isolation. The studies performed in Section 3.2 will be particularly useful to interpret the results and to identify areas where reconstruction can and should be improved.

5.1 Introduction to electron isolation

The intricate environment after a high-energy collision challenges physicists to disentangle the contributions of different particles and background effects to understand the underlying physics processes. Lepton isolation addresses this challenge by characterizing particles based on the activity around them inside the detector. It focuses on identifying the particles of interest, in this case electrons, by considering those not embedded in hadronic jets.

This technique plays a crucial role in distinguishing real electrons from various sources of background noise which can create signals in the detector that resemble those of electrons. An example of this is jets that are wrongly reconstructed as electrons. These cases fall into the fake leptons background, which can become very important in analyses involving leptons in the final state. An estimate for this background can be performed with a simulation, but being able to reduce it as much as possible is preferred. Isolation helps to suppress the contribution of such background by selecting electrons with a relatively low level of surrounding activity. It enhances the purity of the electron sample and improves the measurement of electron properties, such as energy and momentum. This requires accurate reconstruction of the electron trajectory and its energy deposition; isolation aids in minimizing the impact of energy deposits from other particles, leading to more precise measurements.

Track isolation is quantified by defining a cone around the electron's position in the detector and considering all the tracks within that region. If the total momentum of

the tracks is relatively low with respect to that of the electron, it is considered isolated. An example is shown in Figure 5.1.

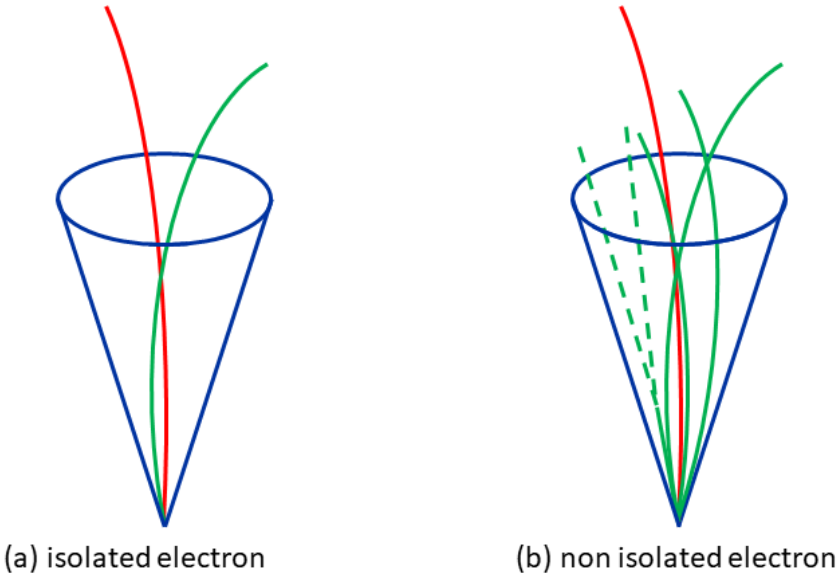


Figure 5.1: Example of an (a) isolated electron considered signal and of a (b) non-isolated electron, either a non-prompt electron or a fake electron

In general, the electrons that are expected to be isolated are those that come directly from the hard scattering, so electrons from the decay of bottom or charm quarks are not included. This study focused on the $Z \to e^+e^-$ process, where a Z boson decays in an electron-positron pair, for the signal and on $t\bar{t}$ for the background, because the presence of many jets increases the probability of having fake leptons. Both the processes have been considered in 200 pileup, corresponding to the ultimate scenario during HL-LHC.

5.2 How isolation is computed and the benefits of timing

For each reconstructed electron a series of criteria is applied, including checks on parameters such as the distance from the primary vertex in z (dz < 0.5 cm) and in the xy plane (dxy < 0.2 cm), p_{T} ($p_{\mathrm{T}} > 10$ GeV), and η ($|\eta| < 2.8$). For those that pass the cuts, the associated track is retrieved. To mark an electron as prompt, meaning that it is expected to be isolated, it is necessary to link it with the simulated objects. Therefore, an associator is employed to link the associated track to a `TrackingParticle`, i.e. the simulated particle coming from the vertex, and subsequently to a generator-level particle

called `genParticle`. If the `genParticle` is of an electron and its parent particle is one among W boson, Z boson, or τ lepton, then the electron is marked as prompt.

The electron location within the detector, whether barrel or endcap, is also taken into account because the study will be split into two regions due to the different conditions and different hardware used for MTD.

At this point, a cone is opened around the electron, and all the reconstructed tracks inside that cone are considered. The conditions that these tracks must meet are a $dR = \sqrt{\eta^2 + \phi^2}$ between the track and the electron from 0.01 to 0.3, a threshold on p_{T} ($p_{\mathrm{T}} > 0.7$ GeV for the barrel and $p_{\mathrm{T}} > 0.4$ GeV for the endcap) and a cut on the distance between the track and the electron along the z axis. The latter can be tuned to improve the isolation efficiency, the value used is dz < 0.2 cm. The p_{T} of the tracks passing all these cuts are summed and divided by the p_{T} of the electron to obtain the isolation value:

$$\texttt{iso} = \frac{\sum_{i}^{i \in cone} p_{\mathrm{T}}^{i}}{p_{\mathrm{T}}^{electron}} \tag{5.2.1}$$

This isolation value is subjected to a tunable threshold and an electron is classified as isolated if its computed isolation value falls below this threshold.

In a scenario where MTD is absent, the cuts on the tracks are based solely on proximity criteria. However, in the presence of MTD, an additional condition involving the time difference at the vertex between the tracks and the electron is applied. The request is the following:

$$\Delta t = |t_{track} - t_{electron}| < N \cdot \sqrt{\sigma_{track}^2 + \sigma_{electron}^2} \tag{5.2.2}$$

where N can be adjusted to find the value that gives the best efficiency. The time information used is the one propagated back to the vertex starting from the local time measured in MTD. It is done going along the particle trajectory at a speed computed using the best mass hypothesis after a fit with the other tracks and vertices. If no time information is available for the track or the electron, the geometric checks remain the only consideration. This happens also if there is time information available, but the quality of the association with the track obtained with the MVA is not good (<0.5).

Since collisions within a bunch crossing happen throughout about 180-200 ps, tracks coming from different vertices, i.e. from different collisions, may be not compatible in time. As shown in Figure 5.2, a track coming from a pileup vertex can finish in the electron cone and therefore can modify the electron properties. Instead, with the check on time compatibility this track will likely be removed.

The presence of a timing layer becomes very important in a high pileup environment, where the possibility of multiple tracks coming not from the primary vertex entering the isolation cone grows a lot. The time compatibility check has the effect of reducing the pileup effects and recovering, in the best-case scenario, the 0 pileup performance.

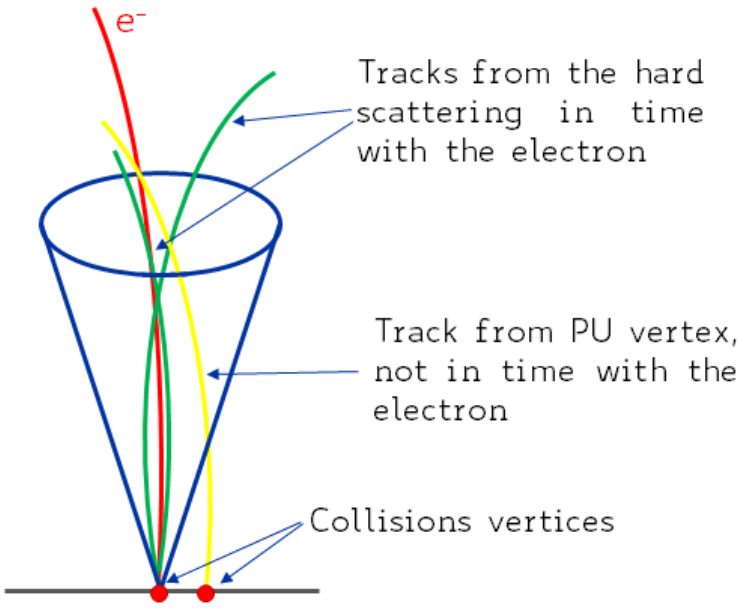


Figure 5.2: Example of the effect of the additional time cut. In this case, based only on the ΔR cut, the yellow track would be included in the isolation cone and would increment the probability of considering that electron non-isolated. Instead, assuming that the two collisions (red points) in the image happened at different moments, the yellow track would not pass the time cut not being compatible with the electron's time, and would be removed from the isolation cone

An alternative possibility to compute the isolation using the time information is to consider the time of the vertex the electron is associated with instead of the electron's track as a reference. Equation (5.2.2) becomes:

$$\Delta t = |t_{track} - t_{vertex}| < N \cdot \sqrt{\sigma^2_{track} + \sigma^2_{vertex}} \simeq N \cdot \sigma_{track} \tag{5.2.3}$$

The upside of this approach is that a vertex with many tracks always has a time associated and the uncertainty is smaller with respect to that of a single track since the time is computed as the average of the times of the tracks that belong to that vertex. However, there is also a downside: if the primary vertex, i.e. the vertex corresponding to the hard scattering, is not identified correctly, the time used to compute the isolation will be wrong. Isolation efficiency studies using the vertex time have been performed as well and the results are shown in the next section.

5.3 Results of isolation efficiency studies

To set a reference on how good the current isolation is and how much it can be improved, the Monte Carlo truth information has been used. As described in the previous section, for each reconstructed track the corresponding `TrackingParticle` and `genParticle` are obtained. The `TrackingParticle` knows its associated simulated vertex and can be used to obtain the true simulated time of that track. In this way is possible to perform the time compatibility check also with the true time. The `genParticles` are the initial particles created by the Monte Carlo event generator and therefore exist only if the particle comes from the hard scattering (pileup particles are added to the simulated event in the following step). The check on the existence of the `genParticle` can be used to select only the particles that belong to the primary vertex and to compute the true isolation. The isolation computed with the true time from the `TrackingParticle` instead represents the best possible isolation that can be obtained with a perfect MTD local reconstruction and the correct assignment of the time information to the track.

Before looking at the isolation efficiency, a check has been done on the distribution of the time difference between the electron and the tracks in the cone. The plots for the $\Delta t = |t_{track} - t_{ele}|$ distribution for signal (ZEE) and background ($t\bar{t}$) are shown in Figure 5.3. Each plot has been done with the simulated (light-blue) and with the reconstructed (red) time information. The shape is the result of the overlap of two distributions: a delta in 0 and a Gaussian centered in zero with the width of the beam spot. The first one corresponds to the tracks in time with the electron, i.e. coming from the same vertex, and instead of a delta is a Gaussian with the resolution of MTD (30-35 ps). The second Gaussian represents the contribution from pileup tracks that have fallen into the cone. The distribution for the background has a bigger peak around zero because there are many more tracks in the cone with respect to the signal.

A cut on the Δt in this case would not remove a lot of tracks since the electrons are not isolated. Both for the simulation and the reconstruction there are additional contributions to the peak in zero. For the simulation (*sim*) this is due to the fake tracks, i.e. reconstructed tracks that are not associated with a simulated track, that cannot be linked to a simulated vertex. For the reconstruction (*reco*) those are all the tracks without a time assigned to them. In these cases, the check on time compatibility is not possible.

In Figure 5.4 the isolation distribution for signal and background in 200PU is shown. The red line represents the isolation values computed without using the time information. The light-blue region represents the isolation computed using the time information reconstructed by MTD. The green region is very similar to the light-blue, except for the fact that it uses the true time of the vertex, while the grey region uses the existence of a `genParticle` to select only the tracks that come from the same vertex as the electron and resembles the 0PU case. There is a clear difference between the signal and background distributions. For the signal, the values are smaller and there is a peak around zero for the isolated electrons with a tail probably due to pileup contamination in the cone not removed by the time cut. On the other side, for the background, the isolation values are bigger since the electrons are not isolated.

The definition of the isolation efficiency is quite simple. The isolation is defined in Equation (5.2.1) and, given a threshold, the electron is isolated if its isolation is under the threshold. The efficiency then is the number of isolated electrons over the total number of electrons. The first thing that can be checked is the efficiency trend as a function of the threshold. This is shown in Figure 5.5 for signal and background in 200PU. Instead of defining a single threshold to decide whether or not an electron is isolated, it is possible to check how the efficiency changes when varying the threshold, in this case from 0.03 to 0.3. As for the previous plot, the red line represents the isolation without using the time information, the light-blue line represents the isolation computed using the time information reconstructed by MTD, the green line uses the true time of the vertex, while the grey line corresponds to the 0PU case. For the signal the efficiency approaches one very fast increasing the threshold, while for the background, even though it increases a bit, the efficiency stays pretty low, as expected. This shows that lower values for the threshold are better because the efficiency on background isolation is very low, meaning that we are pretty good at rejecting fake leptons. At the same time, MTD helps keep up the efficiency of the signal even with a small threshold.

The isolation efficiency has been studied also as a function of the simulated p_{T} of the electron setting the threshold to 0.05, split into barrel and endcap. Each bin in p_{T} is the ratio between the number of isolated electrons and the total number of electrons. Figure 5.6 shows this efficiency for signal and background. The color code is the same as the previous plots. The grey is the truth, corresponding to the 0PU case, and the

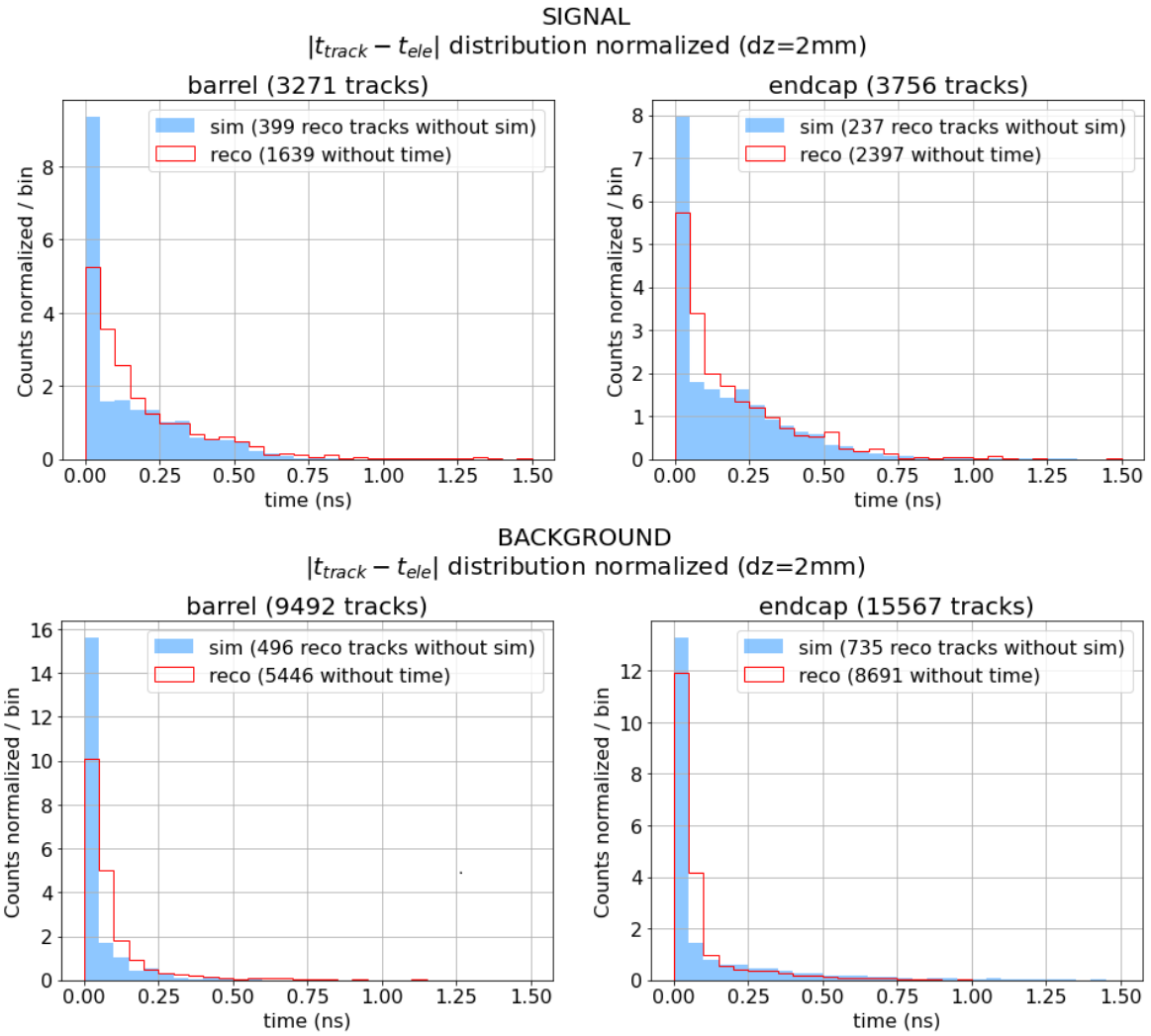


Figure 5.3: Distribution of the $\Delta t = |t_{track} - t_{ele}|$ for signal (ZEE) and background ($t\bar{t}$) in 200PU. The Δt distribution in red has been computed using the time information reconstructed by MTD. The light-blue instead uses the true time of the vertex. The biggest difference between the signal and background distributions lies in the fact that non-isolated electrons are more in time with the tracks in the cone with respect to the isolated ones. The distributions are characterized by a Gaussian centered in 0 with the MTD resolution as σ and another one with the width of the beam spot. The number of tracks in each title is the total number of tracks. In the legend the "sim" is the simulation and the number of fake tracks, i.e. reconstructed tracks that are not associated with a simulated track, is indicated, while the "reco" is the reconstruction, and the number of tracks without a time assigned is shown.

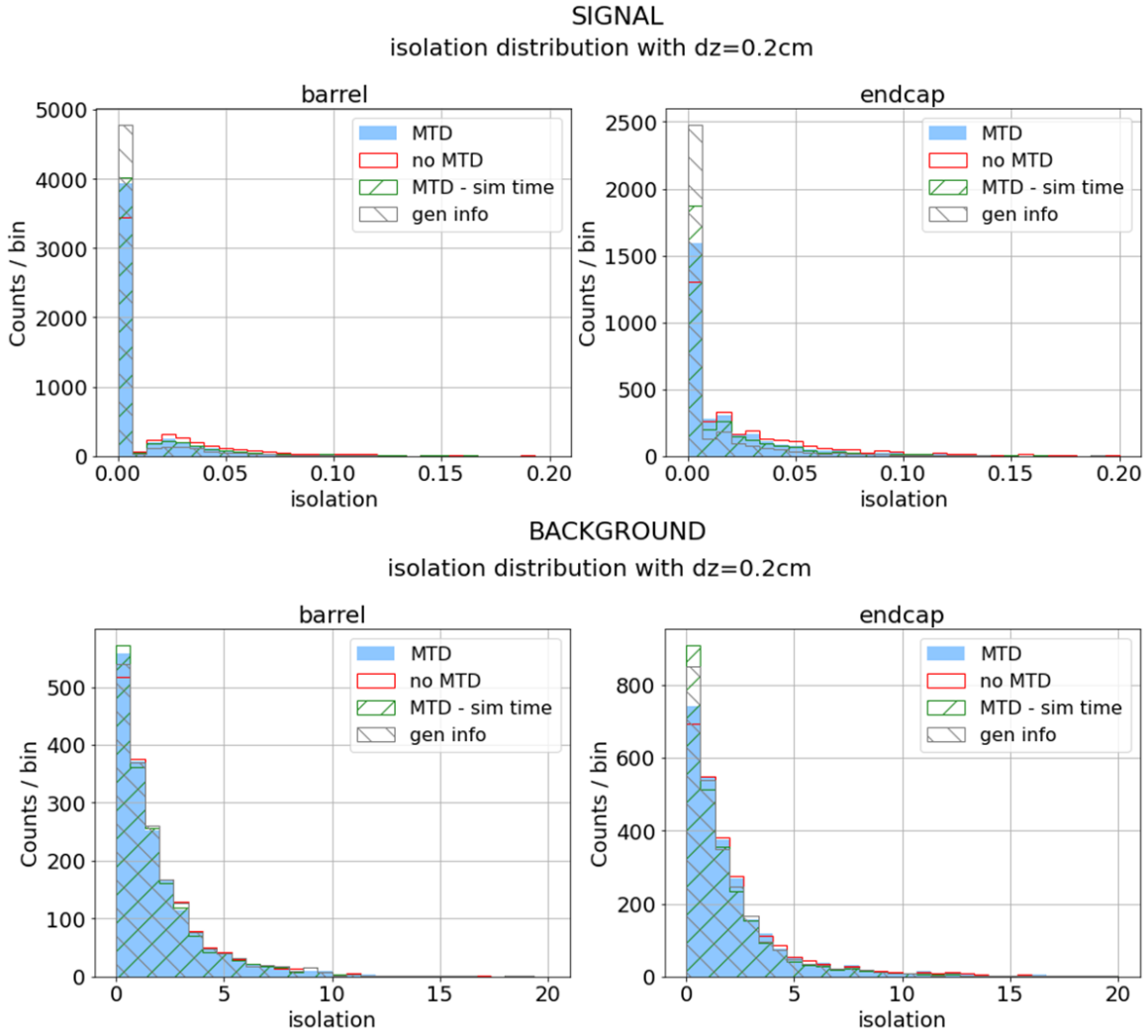


Figure 5.4: Distribution of the isolation value of the electrons for signal (ZEE) and background ($t\bar{t}$) in 200PU. The red line represents the baseline, i.e. the isolation values without using the time information. The light-blue region represents the isolation computed using the time information reconstructed by MTD. The green region is very similar to the light-blue, except for the fact that it uses the true time of the vertex, while the grey region uses the existence of a `genParticle` to select only the tracks that come from the same vertex as the electron and resembles the 0PU case. The difference between the signal and background distributions is very clear. For the signal, the values are smaller and there is a peak around zero for the isolated electrons with a tail probably due to pileup contamination in the cone. On the other side, for the background the isolation values are bigger since the electrons are not isolated

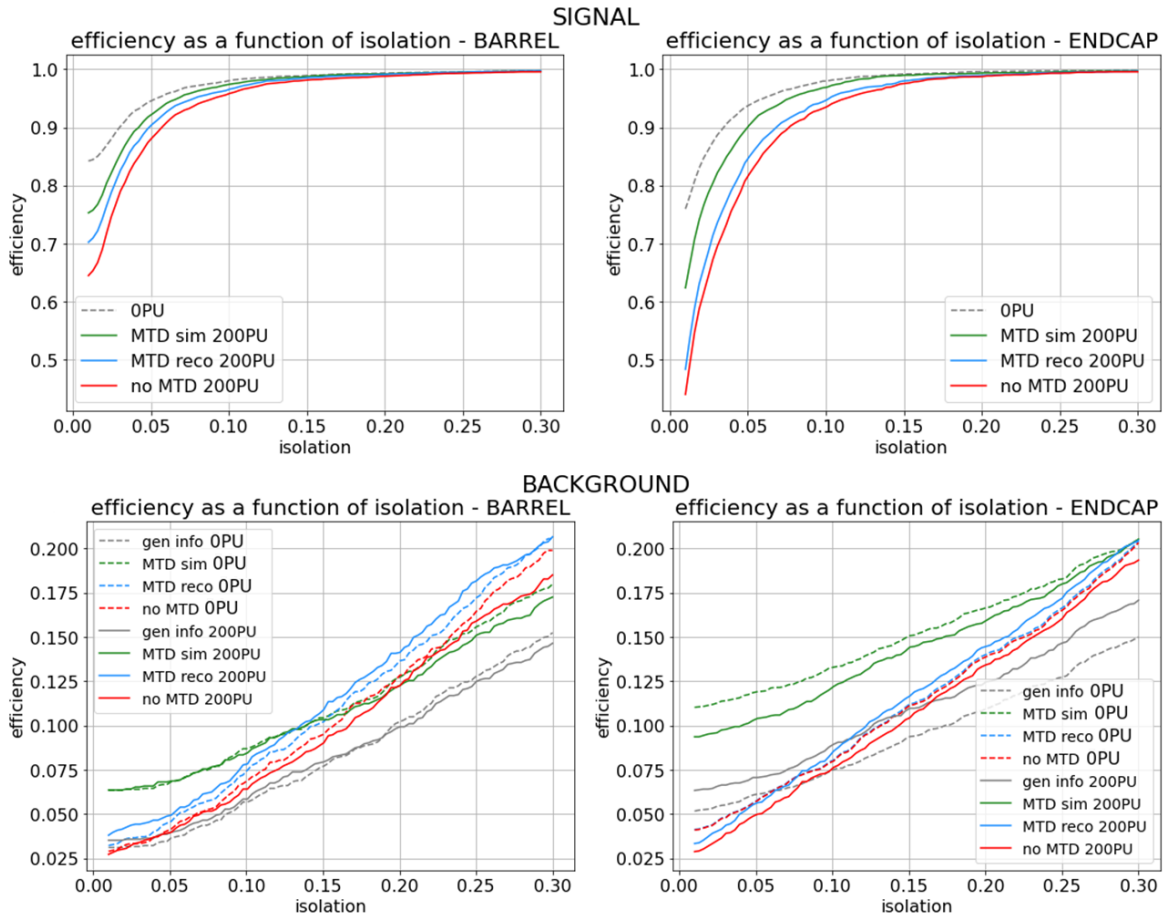


Figure 5.5: The isolation efficiency as a function of the isolation cut for signal (ZEE) and background ($t\bar{t}$) in 200PU. Instead of defining a single threshold to decide whether or not an electron is isolated, it is possible to check how the efficiency changes when varying the threshold, in this case from 0.03 to 0.3. The red line represents the isolation without using the time information. The light-blue line represents the isolation computed using the time information reconstructed by MTD. The green line is very similar to the light-blue, except for the fact that it uses the true time of the vertex, while the grey line uses the existence of a `genParticle` to select only the tracks that come from the same vertex as the electron and resembles the 0PU case

green is what can be achieved with a perfect reconstruction from MTD. Focusing on the signal, there is an improvement in the isolation efficiency using MTD with respect to the baseline where no time information is used. However, there is a noticeable gap between the current isolation performance using the reconstructed time by MTD and the best performance achievable with perfect reconstruction. The possible reasons for this gap are analyzed in Section 5.4. For the background the efficiency with MTD is higher, probably due to cone overcleaning, i.e. more tracks are removed from the cone with respect to the no MTD case, and some electrons are marked as isolated. In general, this is expected because, as shown in the bottom part of Figure 5.3, there is a tail not in time with the electron that will be removed when using the MTD information.

In Figure 5.7 the same study has been done using the time of the vertex as a reference. The color scheme is the same as for the previous plot, but this time the gap between the current reconstruction with MTD and the simulation is smaller because the vertex has always a time associated with it, while that is not true for the electron.

5.4 Inefficiencies breakdown

As noticed in Figure 5.6, there is a gap between the efficiency computed using the current reconstruction and the one computed with the true time from the Monte Carlo simulation and the generator-level information. There are many factors that can affect performance, some of them are:

- geometric acceptance: MTD covers a region up to $|\eta| < 3$, therefore if a track gets to higher η it will not have a time information. In addition, there is a drop in efficiency at around $|\eta| \simeq 1.5$ where there is the transition between BTL (barrel) and ETL (endcap);
- detector efficiency: not all the particles that cross MTD leave hits. If they don't, there will be no time information for their track;
- time assignment to the track: the algorithm that performs the time assignment to the track may not always find the link with the correct cluster or may not find a link at all even if the cluster is there;
- time resolution: MTD has a resolution of 30-40 ps, that will drop to 60-70 ps at the end of HL-LHC. This can spread the track times.

Not much can be done regarding the geometric acceptance, while the detector efficiency has been already discussed in Section 3.2. These are the main reasons for the gap between the green (sim) and the grey (gen) points, while the third and fourth items in the list can be part of the reason for the gap between the light-blue (MTD) and green (sim) ones.

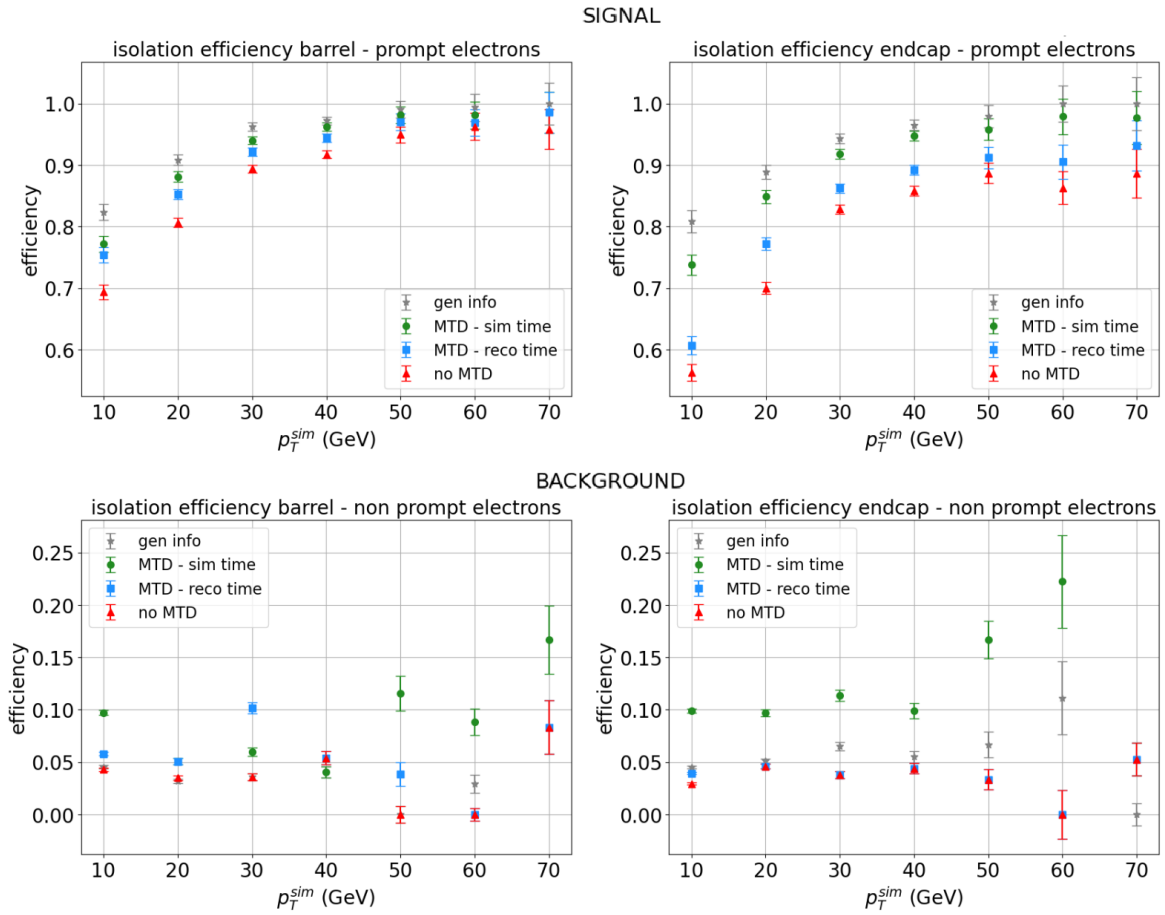


Figure 5.6: The isolation efficiency as a function of the p_{T}^{sim} of the electron for signal (ZEE) and background ($t\bar{t}$) in 200PU using the electron time as a reference. The red points represent the baseline, i.e. the isolation efficiency without making use of the time information. The light-blue points represent the performance using the time information reconstructed by MTD. Those two use only reconstructed information. The green and grey points instead use true information from the Monte Carlo simulation. The green points are very similar to the light-blue, except for the fact that they use the true time of the vertex, while the grey points use the existence of a `genParticle` to select only the tracks that come from the same vertex as the electron. The grey is the truth, corresponding to the 0PU case, and the green is what can be achieved with a perfect reconstruction from MTD

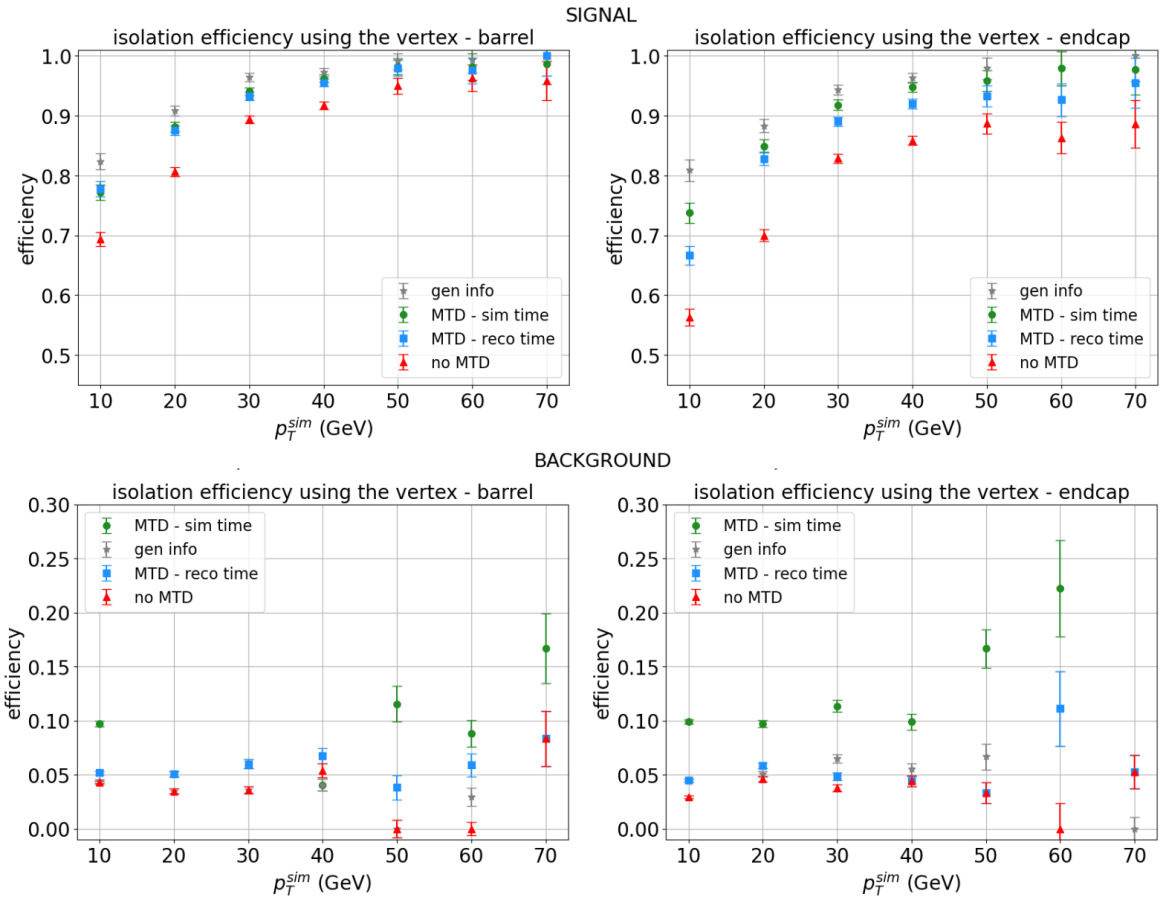


Figure 5.7: The isolation efficiency as a function of the p_{T}^{sim} of the electron for signal (ZEE) and background ($t\bar{t}$) in 200PU. The color scheme is the same as for Figure 5.6, but this time the reference is the time of the vertex and not the time of the electron. The efficiency in the MTD case is higher with respect to before because the vertex has always a time associated with it, while that is not true for the electron

In Figure 5.8 the efficiency of the MTD reconstruction in assigning a time to the track is shown, splitting the tracks associated with the electrons (right) and the other tracks in the cone (left). The efficiency is around 0.7, a bit less for the electrons in the endcap. The reason could be they go through more material before reaching the endcap and therefore have more possibilities to interact, change direction, and create other particles. For the electrons in particular, also Bremsstrahlung has to be taken into account.

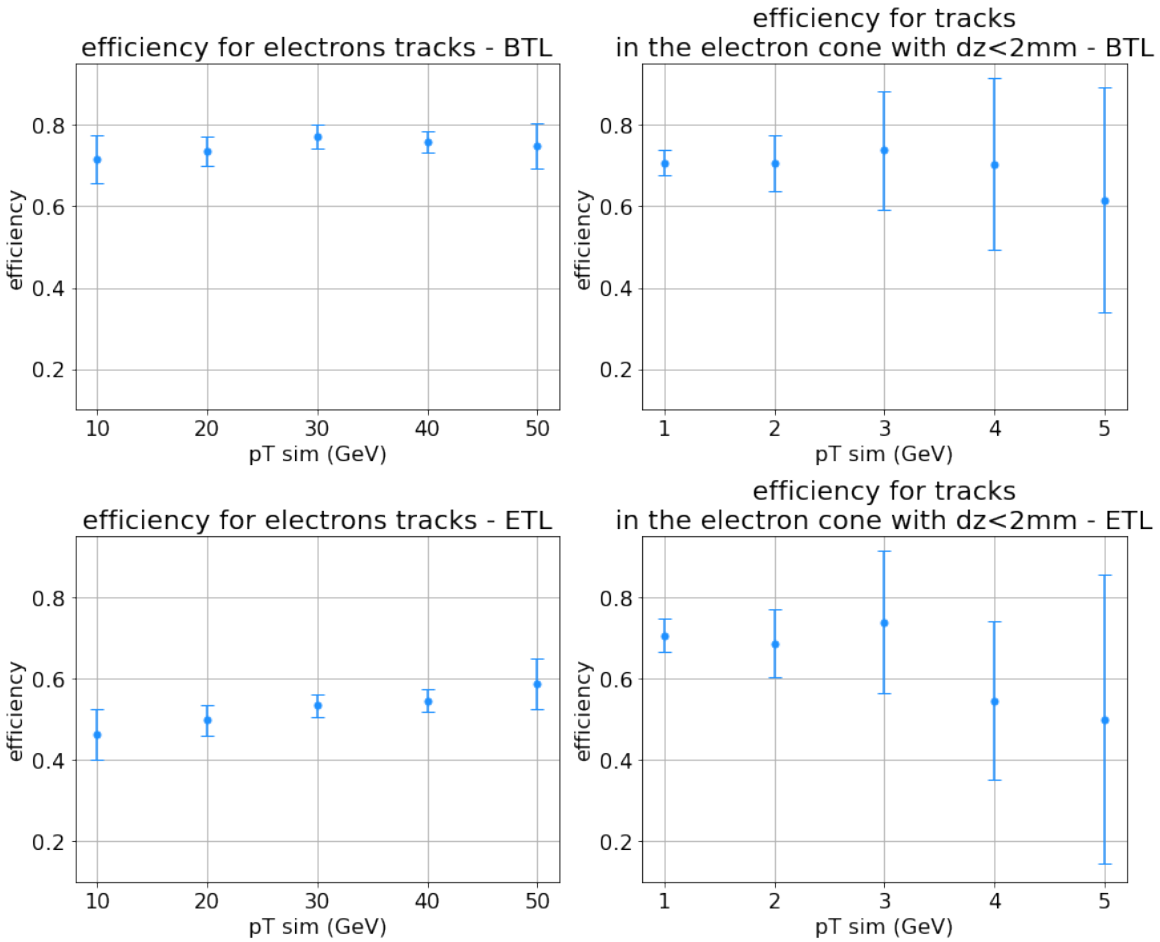


Figure 5.8: Efficiency of the MTD reconstruction in assigning a time to the track divided between barrel (BTL) and endcap (ETL) and between the tracks associated with the electrons (right) and the other tracks in the cone (left). The efficiency is around 0.7, a bit less for the electrons in the endcap

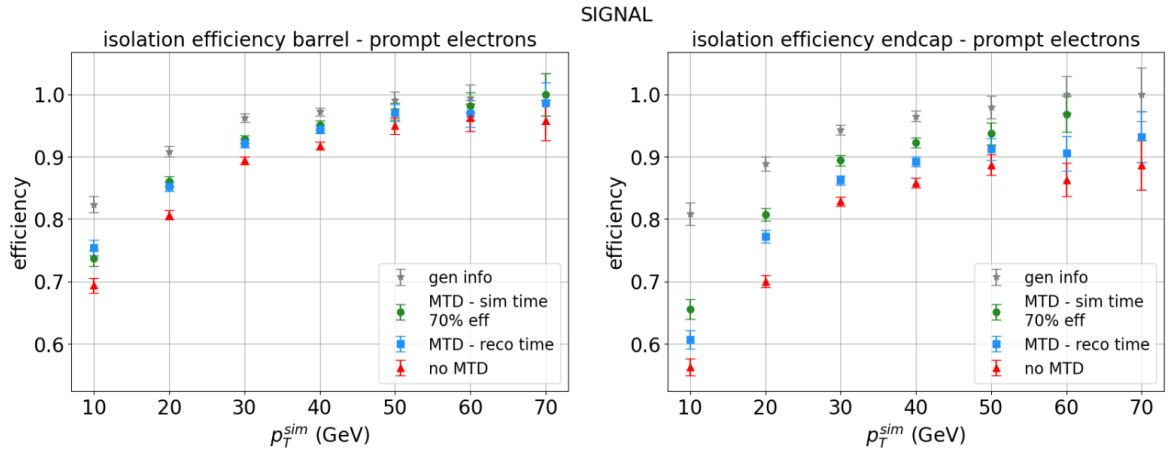


Figure 5.9: The isolation efficiency as a function of the p_{T}^{sim} of the electron for signal (ZEE) in 200PU. The color scheme is the same as for Figure 5.6, but in these plots for the simulation (green) 30% of the times associated with the tracks have been dropped, to emulate the efficiency of MTD in the time assignment to the track. This has a great impact on the efficiency and may be the first reason for the gap between the simulation and the reconstruction.

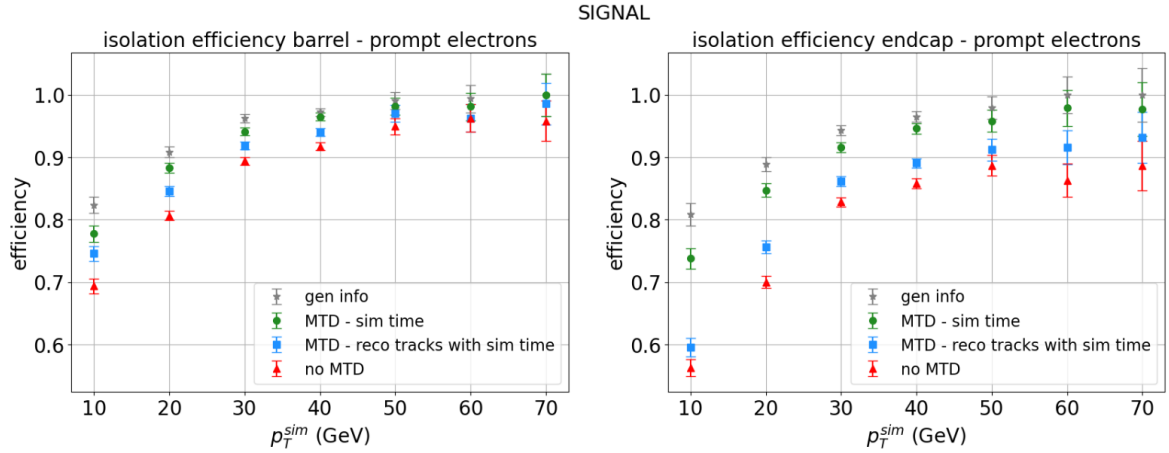


Figure 5.10: The isolation efficiency as a function of the p_{T}^{sim} of the electron for signal (ZEE) in 200PU. The color scheme is the same as for Figure 5.6, but in these plots, the time used for the electrons in the MTD case (light-blue) is the true time of the vertex. The idea is to check the influence of the time resolution on the efficiency computed using the time information and, from what can be seen from the plot, the impact is negligible.

Since the assignment efficiency is more or less flat in p_{T}, a check on the impact that this can have on the isolation can be obtained by randomly dropping 30% of the times from the simulation. However, a more consistent check should be done by removing them as a function of p_{T}. The result of this study is shown in Figure 5.9 and it highlights that the gap is considerably smaller now, indicating that this inefficiency is probably one of the most important reasons for the gap in performance.

Another study that has been done consists of using the simulated times instead of the reconstructed ones for the tracks. This is reported in Figure 5.10 and shows that there is no substantial improvement when doing this, meaning that the impact of the MTD time reconstruction is not relevant.

To conclude, the main source of inefficiency in the current performance lies in the assignment of time to the tracks. Considering that the lepton used for the isolation studies is the electron that has a particularly difficult track to reconstruct due to the non-negligible bremsstrahlung that requires a dedicated track reconstruction, the efficiency will improve when MTD has a dedicated reconstruction for the electrons.

Chapter 6

Heterogeneous computing in CMS

The higher instantaneous luminosity and pileup during the High Luminosity Phase of the LHC and the more granular sub-detectors will cause a considerable increase in the complexity of event processing at the Level-1 trigger, at the High Level Trigger (HLT) and also during the offline reconstruction. Looking at the online reconstruction, the CMS experiment will increase the Level-1 output rate from 100 kHz in Run 3, up to 750 kHz. The HLT will have to reduce the event rate with respect to the Level-1 trigger by a factor of 100 as in Phase 1 while maintaining comparable efficiencies for the main physics objects while being constrained by the online computing resources available. For this reason, substantial upgrades of the Data Acquisition (DAQ) system are planned [28] to fully exploit the physics reach of the HL-LHC and to cope with the requirements posed by the detector upgrades and the higher luminosity.

One path that has been explored is that of heterogeneous computing, which consists of offloading part of the computing work to devices called accelerators that are designed to perform specific computational tasks much faster and more efficiently with respect to CPUs. Some examples of accelerators include Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), and Application-Specific Integrated Circuits (ASICs).

This work is focused on the High Level Trigger (HLT) reconstruction, where the choice has been to combine CPUs and GPUs. The CPU and GPU hardware exhibit distinct architectures, each offering specialized capabilities. The CPU manages computational tasks more serially, often involving a smaller number of cores operating at higher frequencies (typically around 3-4 GHz). It excels at optimizing single-threaded tasks, executing them in a pipelined manner to maximize efficiency. In contrast to CPUs, GPUs are optimized for SIMD (Single Instruction, Multiple Data) operations. The GPU architecture is focused on parallelism and is characterized by the presence of an array of Streaming Multiprocessors (SMs) each composed of many Streaming Processors(SPs) or cores. Typically, a GPU has thousands of cores running at a lower

frequency (1-2 GHz). An SM is the central part of a GPU where the device functions are executed. This architecture is tailored to parallel execution, breaking down tasks into separate threads and processing them concurrently. More details on the GPU programming model will be given in the next section.

Although it doesn't lower the absolute computing power requirements, heterogeneous computing offers new possibilities for CMS. The high level of parallelism offered by these specialized accelerators and the fact that they are optimized to execute specific tasks very efficiently can reduce the total execution time and energy consumption. A reduced processing time allows us to analyze more events and perform a more complex reconstruction, improving the quality of the selection at the trigger level. To fully exploit the hierarchical structure of GPUs and the massive parallelism they offer, developers need to rethink and redesign the algorithms and optimize them for parallel execution. As described in Section 1.4.4, the HLT selection algorithms are executed within the same reconstruction framework used for offline processing. As a consequence, rewriting the online reconstruction to run on GPUs not only reduces the computational cost but makes it available also offline, enabling the possibility to use all the resources available at the Worldwide LHC Computing Grid (WLCG) and High-Performance Computing (HPC) facilities, where accelerators are becoming more and more common.

CMS demonstrated the advantages of the adoption of GPUs with the R&D work done by the Patatrack group, with the Pixel Tracks and Vertex reconstruction written in CUDA [50] to run on NVIDIA GPUs. The deployment of a HLT farm in Run 3 equipped with two NVIDIA Tesla T4 GPUs for each of its 200 nodes increased the event processing throughput by 70%, with a 50% better performance per kW and a 20% better performance per initial cost. The algorithms' physics performance has been improved because they were rewritten making a better use of detector data. Additionally, the reduced execution time provided the opportunity to perform a more sophisticated reconstruction.

6.1 Introduction to heterogeneous computing

Heterogeneous computing is commonly used in high-performance computing (HPC) systems, where large amounts of data need to be processed quickly and efficiently. However, developing software for heterogeneous computing can be more challenging than for homogeneous computing, as it requires the use of specialized programming models and tools.

In heterogeneous computing applications, it is common to use the terms "host" and "device" to differentiate between different types of processors in a system. The "host" refers to the general-purpose CPU that manages and controls the overall system, and is responsible for managing memory and scheduling and executing tasks. In particular, it schedules memory operations such as the transfer of data between the host and device

and schedules the so-called kernels, sub-programs executed on the device. Memory operations (copy, set) and kernels can be submitted to a queue, i.e. a sequence of commands associated with a device that are executed in order. These operations are usually non-blocking and asynchronous with respect to the host, meaning that if the results are needed by some other computation on the host this must synchronize explicitly with the queue. When the execution on the host reaches that point it stops and waits for the device to complete all the tasks that have been queued.

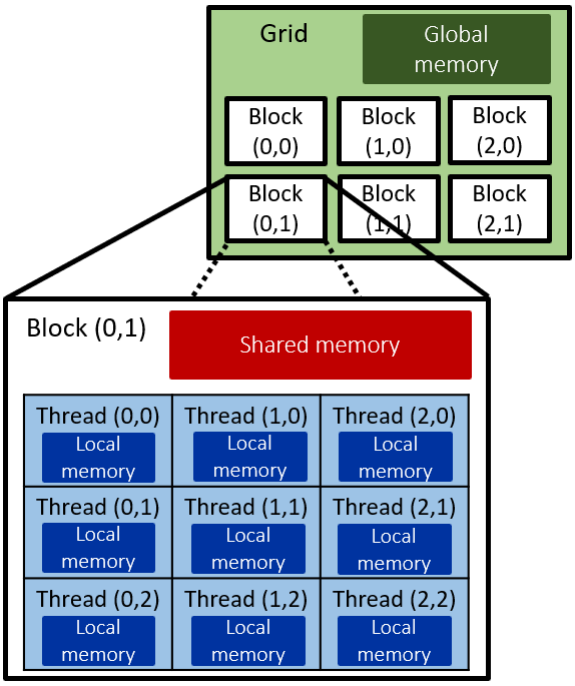


Figure 6.1: The GPU programming model is organized in a logical hierarchy of grids, blocks, and threads. Different blocks in a grid run independently of each other and can access the same global memory. Within a block, different threads run concurrently, with each thread having its local memory. Additionally, there is another type of memory shared between all the threads in a block, the shared memory.

In contrast, the "device" is a specialized processor that is optimized for specific tasks, e.g. a GPU. GPUs have a logical hierarchy of grids, blocks, and threads that allow for efficient and parallel execution of complex computations (see Figure 6.1). In this hierarchy, different blocks run independently of each other and can access the same global memory. Within a block, different threads can run concurrently, with each thread having its own memory. In addition, there is another type of memory within GPU architecture that is shared among all the threads residing within a particular block. It allows data sharing and exchange among threads, thereby fostering more efficient and coordinated parallel execution. However, given that different threads can access the

same memory, race conditions can arise. Race conditions happen when multiple threads attempt to write to the same variable simultaneously, leading to unpredictable and erroneous outcomes. To avoid them and ensure data integrity, specialized approaches like the use of atomic operations come into play. Atomic operations are designed to execute uninterruptedly, ensuring that no other thread can intervene during their execution. As a result, when threads perform atomic operations on shared memory, one thread at a time is allowed to modify the shared variable preventing concurrent writes and data inconsistencies.

An additional feature of the programming model is sub-groups of threads, called warps in CUDA. All the threads in a sub-group execute the same instruction sequences together as frequently as possible to maximize performance. Sub-groups also allow having multiple threads to issue common instructions to data instead of only a single thread.

The CMSSW framework has been extended with the concept of *external worker*, a mechanism to allow modules to offload asynchronous work outside of the framework scheduler. The `produce()` method of the traditional `EDPProducer` is split into two: `acquire()` and `produce()`. The `acquire()` method reads the input data and launches one or more asynchronous operations. When these operations are complete, the CMSSW framework runs the `produce()` method. In this way, the tasks are asynchronous and can be scheduled in parallel with respect to each other and other tasks running on the host, without blocking the whole execution [51]. GPUs may not always be available, therefore it is desirable to have the possibility to fall back on CPUs. For this purpose, a new producer has been introduced, the `SwitchProducer`. It lists the alternative implementations of an algorithm in the same configuration and the decision on the most adequate device to use is performed at runtime depending on the available resources. As we will see in the next section, this can become hard to maintain because it requires a plain C++ version of the code for CPUs in addition to the CUDA version for NVIDIA GPUs.

6.2 Performance portability libraries

One main drawback of heterogeneous computing is the need to write different code for each architecture since specialized programming frameworks are needed. The difference is not only between different kinds of devices, e.g. between CPUs and GPUs but also between devices from different vendors. On the CPU side, there are AMD and Intel (x86), ARM, and IBM Power. For what concerns GPUs the main vendors are NVIDIA, AMD, and Intel and each of them has its programming language, namely CUDA [52], HIP [53] and SYCL [14].

As already said, the Pixel Tracks and Vertex reconstruction used to demonstrate the benefits of using GPUs has been written in CUDA to target NVIDIA GPUs. However,

to run on AMD GPUs the code should be rewritten using HIP and to run on CPUs a plain C++ version is needed. In addition, although NVIDIA GPUs are the more widespread in HPC centers [54], the adoption of GPUs from other vendors is increasing. To give some examples, the supercomputer Frontier at Oak Ridge is equipped with AMD GPUs [55], while Aurora at the Argonne National Laboratory employs Intel GPUs [56]. Writing multiple versions of the same code introduces code duplication and makes the project more difficult to maintain, and therefore should be avoided as much as possible.

A solution to this is provided by performance portability libraries, tools that allow the user to write a single code base that can be compiled and executed on different target architectures. These libraries provide an abstraction for parallelism and data management and take care of offloading the code to the targeted device. The concept of abstraction is the process of concealing system complexities, providing a refined interface to the developer that filters out the non-essential details. As already described, CPUs and GPUs are substantially different, both in the architecture and in the way tasks are scheduled to be executed on them. Portability layers hide these differences providing a common interface for all the architectures. This leads to a more maintainable code and avoids the need to maintain different implementations of the same algorithm.

Having code that can run on different devices extends the number of HPC centers that can be targeted and the range of architectures and vendors among which to choose the best ones for the HLT in terms of performance, power consumption, and cost. It will also allow the automatic fallback to CPUs when GPUs are not available.

Among the existing performance portability libraries, those evaluated by CMS are [57]:

- alpaka [58]: short for Abstraction Library for PArallel Kernel Acceleration, it's a header-only C++17 abstraction library for accelerator development. It's open source and supports serial and parallel execution on CPUs, NVIDIA GPUs, and AMD GPUs. The support for Intel GPUs and FPGAs has been addressed during this thesis work and is now officially part of the alpaka library. It allows to build one single executable that runs on different back-ends at the same time.

- kokkos [59]: is a programming model and a C++ library for performance portable applications. It supports many back-ends: CUDA, HIP, SYCL, High-Performance ParalleX (HPX), OpenMP, and C++ threads, but currently, it can be compiled and executed for only one back-end at a time, making it not suitable for CMSSW [60].

- `std::par` [61]: NVIDIA developed an NVC++ compiler that can compile Standard C++17 algorithms with parallel execution policies for execution on NVIDIA GPUs. At the moment the compiler presents bugs and some features like shared memory are not supported.

- OpenMP [62]: is an industry-standard API that supports parallel programming in C++ and other languages. It provides a set of compiler directives and runtime libraries to enable developers to write parallel code. Preliminary studies showed slower kernels and much more data movement with respect to the native version.
- SYCL/oneAPI [14, 15]: SYCL is a cross-platform abstraction layer maintained by Khronos Group and has been included by Intel in its oneAPI programming model. It's the main subject of the next sections.

CMS has investigated these performance portability solutions looking at programming experience, ease of development, framework integration feasibility, performance with respect to the native implementation, performance of fallback to CPU, user support, supported back-ends, and long-term prospects.

The choice for Run 3 has been alpaka since it meets all the requirements needed to be integrated into the CMS software and has shown close to native performance both on CPUs and GPUs [63]. This prompted an evolution of the CMSSW framework. The `SwitchProducer` used to choose between the CPU and CUDA versions is no longer needed and the usual method to schedule the module can be used with a small addiction (`@alpaka` after the module name) that specifies that the module uses alpaka. It means that many versions of the module exist and the framework decides which version to run on which device based on the configuration and available hardware.

The studies on portability libraries are still ongoing and in this work, the performance of SYCL/oneAPI has been investigated.

6.2.1 SYCL / oneAPI

SYCL is a cross-platform programming model developed by Khronos Group that allows code to be written using standard ISO C++ both for the host and the device in the same source file. The first version, 1.2, was released in March 2014, followed by SYCL 1.2.1 three years later. To address the many limitations of this version, the Khronos Group launched SYCL 2020 in February 2021.

Starting from the SYCL 2020 standard, Intel has developed the oneAPI toolkit for parallel programming on heterogeneous computing systems. It includes the Data Parallel C++ (DPC++, Figure 6.2), a language built upon SYCL and the ISO C++ standard that extends them to provide explicit support to run parallel algorithms on different devices. DPC++ supports many back-ends: CPUs, Intel FPGAs, Intel, NVIDIA, and AMD GPUs. To target NVIDIA and AMD GPUs it is necessary to use the Intel oneAPI 2023 release (released December 2022) or newer. Since part of the studies shown here have been carried out before that date, the open source Intel's LLVM Compiler [64] has been used to target non-Intel devices.

The single-source code written in SYCL can be compiled for different back-ends at the same time allowing the user to choose the preferred device at runtime. The SYCL

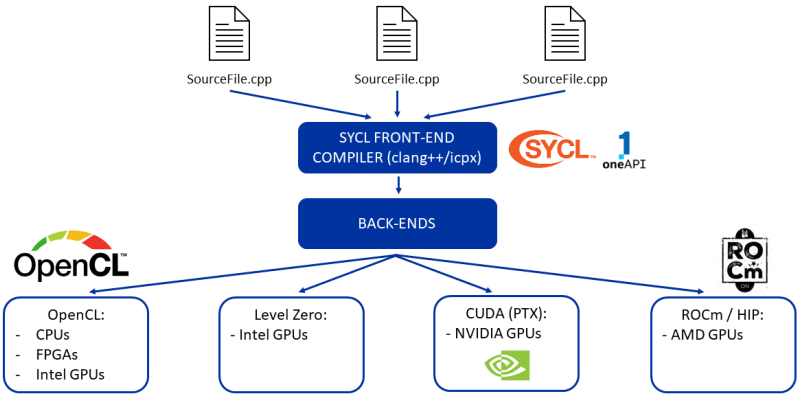


Figure 6.2: The DPC++ programming model

programming model starts from the definition of a queue, associated with a specific device, that is used to submit memory operations and kernels. Those operations run asynchronously on the device and the execution order is sequential only if explicitly specified, therefore synchronization is required before accessing results from the host.

There are different reasons to explore this library starting from the fact that when a new compatibility layer is developed, it is always relevant to explore its performance in comparison to native code and other compatibility layers. Being an open standard, SYCL can count on a variety of implementations. This allows for more flexible adaptations of the standard to specific needs. Furthermore, it is supported by Intel, a huge company in the tech industry, which bodes well for the future support of the standard.

6.3 Patatrack pixel tracks and vertices reconstruction with SYCL

The performance of SYCL has been evaluated on the Patatrack pixel tracks reconstruction, which includes the modules for pixel-only tracks and primary vertices reconstruction for the CMS "Phase 1" Pixel detector. The signals generated by charged particles traversing the pixel detectors are digitized by the read-out electronics and packed to minimize the data rate. To reduce the copy of data from/to the device, the raw data are transferred from host to device and then the full chain runs on the device.

Starting from those data, the first step is the reconstruction of the individual hits in the detector. The digitized information is unpacked and interpreted to create *digis*: each digi represents a single pixel with a charge above the signal-over-noise threshold and contains information about the collected charge and the local row and column position in the module grid. This process is parallelized on two levels: information

coming from different modules is processed in parallel by independent blocks of threads, while each digi within a module is assigned a unique index and is processed by a different thread.

Neighboring digis are grouped to form clusters using an iterative process. If two or more adjacent digis are found, the one with the smaller index becomes the seed for the others. This procedure is repeated until all the digis have been assigned to a seed. Digis and clusters work in local coordinates. Then, they are converted to global coordinates and corrected for the beam spot position.

Clusters are then linked together to form n-tuplets that are later fitted to extract the final track parameters. The n-tuplets production starts from the creation of doublets by connecting hits belonging to adjacent pairs of pixel detector layers (and some non-adjacent pairs to account for geometrical and detector inefficiencies). The doublets that share a common hit are tested for compatibility to form a triplet. Root doublets are identified by having the inner hit on the inner layer of the detector and starting from each of them a depth-first-search (DFS) is performed to obtain n-tuplets. The n-tuplets are used as input for the track fit done with the "Broken Line" fit [65]. It takes into account energy loss and multiple scattering of the particle passing through the detector and is performed in three steps: a first fast fit in the transverse plane, a line fit in the R-z plane, and a circle fit in the transverse plane. Each n-tuplet is assigned to a different thread. In the last module, the fitted pixel tracks are used to form pixel vertices. Vertices are searched by clustering the z coordinate of the point of closest approach of each track with the beamline. Finally, the vertices are sorted using the sum of the p_T^2 of the contributing tracks. The vertex with the largest $\sum p_T^2$ is labeled as the "primary" vertex. A scheme of the reconstruction is represented in Figure 6.3.

In total, there are almost 40 kernels, with an execution between few μs to ~ 1 ms per event. These kernels employ parallel patterns like warp-level primitives and cooperative group methods (functions that operate among all the threads in a block). Shared memory and atomic operations are used as well.

In this work, the full Patatrack pixel tracks reconstruction has been rewritten in SYCL.

6.4 Computing performance of the SYCL application

The performance studies have been performed using 1000 $t\bar{t}$ simulated events from CMS open data, with an average of 50 superimposed pileup collisions with a center-of-mass energy of 13 TeV. The data are read at the beginning of the job and repeated as many times as needed. Before looking at the computing performance, all the physics results have been validated with the dedicated plugin. It compares the results obtained with the expected ones for the number of modules, digis, clusters, tracks, and vertices.

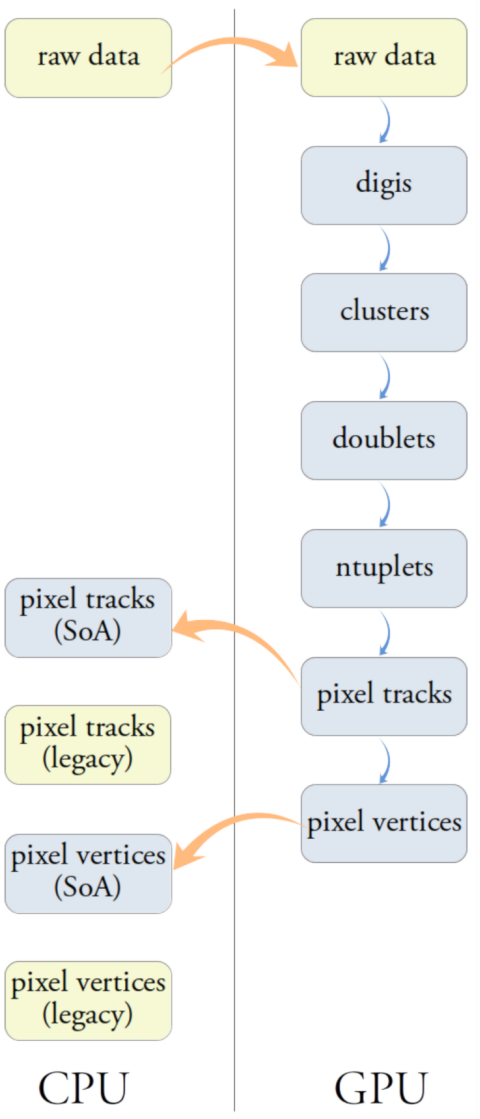


Figure 6.3: Steps of the pixel tracks and vertices reconstruction: after the transfer of raw data to the GPU, the whole chain from digis unpacking to vertices creation runs on the GPU. The results can optionally be copied back to the CPU to be saved.

Computing performances have been evaluated on CPU and GPU comparing the SYCL version with the native (serial, CUDA) and alpaka versions. The devices used for the benchmark are:

- CPU: AMD Ryzen 9 5900X 12-Core Processor [66]
- GPU: NVIDIA GeForce GTX 1080Ti [67]

The performance plot for the CPU is shown in Figure 6.4, while for the NVIDIA GPU, the results are shown in Figure 6.5. In both cases on the y-axis there is the throughput, i.e. the number of events processed per second, while on the x-axis there is the number of concurrent events, i.e. the number of events processed simultaneously. Each point is the average of 10 repetitions with 10k (1k) events for the GPU (CPU).

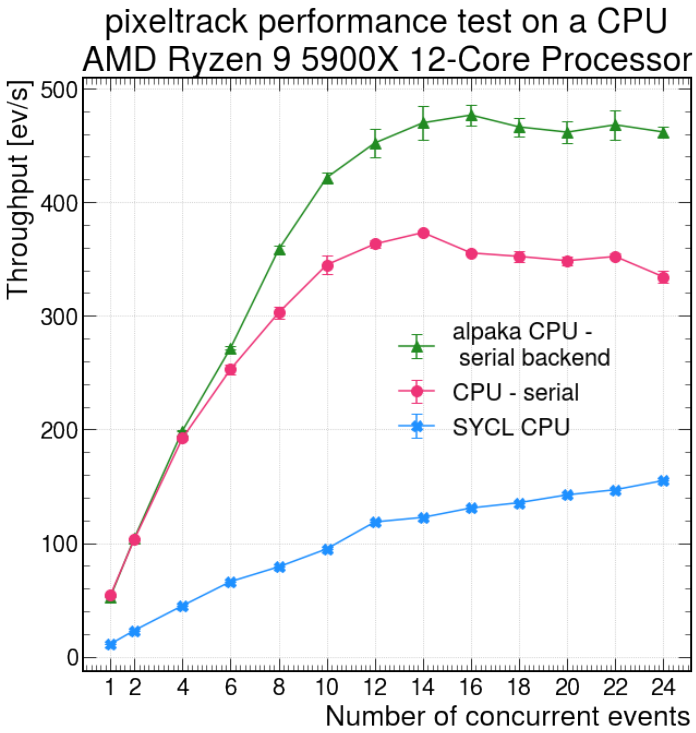


Figure 6.4: Performance comparison between native serial code, alpaka code running through the serial back-end, and SYCL code running through the CPU back-end on a CPU AMD Ryzen 9 5900X 12-Core Processor

In both cases what can be observed is that the SYCL implementation is not able to reach either the native implementation or the alpaka version. Some profiling and analysis have been done to try to understand the reasons for this gap in performance. This is a list of the main slowdowns observed:

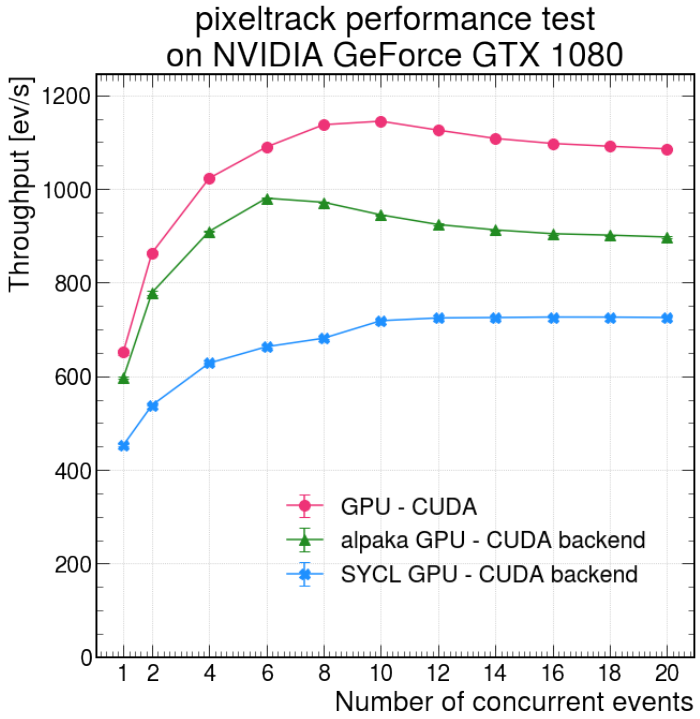


Figure 6.5: Performance comparison between native CUDA code, alpaka code running through the CUDA back-end, and SYCL code running through the CUDA back-end on a GPU NVIDIA GeForce GTX 1080Ti

- Collective operations (`sycl::all_of_group()`, `sycl::any_of_group()`, `sycl::shift_group_right()`, `sycl::reduce_over_group()`...) are slower than their CUDA counterpart
- Every kernel launch (`submit()`) and memory operation (`memcpy()`, `memset()`) creates a SYCL event and this is costly in terms of performance.
- SYCL events cannot be reused, so new SYCL events are created for each framework event. In CUDA events can be reused across different events.
- To avoid data races more synchronizations have been introduced slowing down the code.

In addition to this, it has been highlighted that SYCL kernels have a larger overhead than the CUDA ones and this becomes noticeable with the small kernels present in this application.

6.5 SYCL as a back-end of alpaka

As already said, alpaka has been chosen as the portability layer in CMSSW for Run 3. Once the migration of the Patatrack pixel tracks reconstruction code to SYCL was completed, the application ran successfully and the functioning of SYCL/oneAPI was understood, the next step has been the integration of SYCL as a back-end of alpaka, to be able to target Intel GPUs also in CMSSW.

Alpaka included an experimental SYCL back-end, but it was in an experimental state and was based on SYCL 1.2.1. Therefore it did not use the latest standard with the Unified Shared Memory (USM), but a different approach based on buffers and accessors. USM has a pointer-based approach, where memory is allocated on the host and the value of the pointer returned is guaranteed to be valid also on the device. However, this does not mean that device memory can be accessed by the host; the pointer value is valid and can be used, but dereferencing it is generally still an error. On the other hand, buffers are just views of the allocated memory and to access it accessors must be used. The limitation of this approach comes from the fact that complex data structures that rely on pointers (e.g. lists, trees, n-dimensional arrays...) cannot easily be passed from host to device or back.

A lot of work was involved in rewriting the SYCL back-end to use USM and to align with the latest standard and compiler version, but now the SYCL back-end is an official back-end of alpaka and will be included in the upcoming alpaka version 1.0. It has been tested and used to run the alpaka version of the pixel tracks application on an Intel Data Center GPU Max 1100 (Ponte Vecchio). The performance of the native SYCL version of the code and the alpaka version are shown in Figure 6.6. oneAPI can target

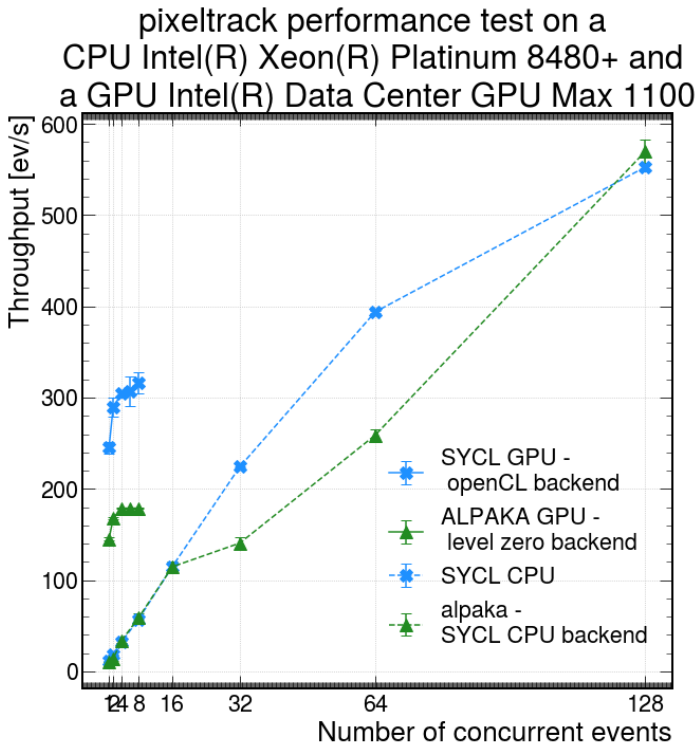


Figure 6.6: Performance of the native SYCL implementation and of the alpaka implementation with the SYCL back-end running on a CPU Intel(R) Xeon(R) Platinum 8480+ and on an Intel Data Center GPU Max 1100 (Ponte Vecchio)

Intel GPUs through two back-ends: OpenCL and Level Zero. Both have been tested because, having different implementations, can lead to different performances, and the best one is shown in the plots. The results for the alpaka version are more or less consistent with the native version on the CPU, while on the Intel GPU, they are worse. This was expected since profiling of the back-end showed some additional overhead in the alpaka version when launching kernels that will require further optimizations.

Some features, like device global variables, have not been implemented yet because the way these variables have been implemented in oneAPI is substantially different from what CUDA, HIP, and therefore alpaka itself, do. Integrating this feature would imply a breaking change in the alpaka library and has to be addressed carefully.

Chapter 7

Conclusions and future perspective

The High-Luminosity phase of the LHC, starting in 2029, will pose challenges to the CMS data acquisition and event reconstruction and will force the detectors to operate in much harsher conditions. To tackle these challenges CMS has decided to devote part of its sub-detectors upgrades to provide time-related information about particles in collision events.

This work has demonstrated the importance of incorporating timing information into the global event interpretation, particularly in the face of high pileup scenarios. A comprehensive study of the time measurements provided by MTD and HGCAL has been carried out, assessing the efficiency of current MTD reconstruction and refining the time attribution for energy deposits in HGCAL to minimize the introduction of additional errors and uncertainties. Throughout this process, the development of Monte Carlo Truth information has played a pivotal role in verifying the accuracy of these studies.

Consequently, two aspects of the event reconstruction have been examined. The first one is the linking between tracks and energy deposits in the calorimeters. A better version of the linking has been developed and the time compatibility check has been corrected and improved, demonstrating a substantial increase in performance with respect to the original version of the algorithm. While the time compatibility constraint may reduce efficiency by approximately 10% on a sample of single pions, it substantially reduces incorrect connections, by a factor of two or more depending on energy. Future efforts will aim to regain the efficiency lost while preserving the benefits of time compatibility.

The other aspect considered is the reconstruction of physics objects with a specific focus on electron isolation. Employing a time compatibility constraint between tracks in the vicinity of electrons and the electrons themselves has revealed enhanced isolation efficiency for a sample of Z bosons decaying in an electron-positron pair in events with high pileup, moving closer to the performance for events without pileup than the

efficiency computed without time constraints. Using the time of the vertex instead of the time of the electron has shown even greater improvement, driven by the consistent association of time with the vertex. However, using the simulated information realized previously the efficiency improves even more, pointing out the necessity to improve the current reconstruction to obtain an even better result.

In summary, the integration of timing information into event reconstruction has demonstrated its efficacy in mitigating the challenges posed by high pileup scenarios, lowering the effective pileup to a level below the actual one. Nevertheless, there remain unresolved issues that demand attention to refine the quality of time information, thereby enhancing the overall reconstruction process. Future studies will address these concerns to improve the time-related quantities. Looking ahead, the prospect of expanding the use of timing information to other stages of the reconstruction, such as Particle Flow algorithms and PUPPI, holds great promise and will be covered in future developments.

Heterogeneous computing has been the choice of CMS to fulfill the processing power requirements of the HL-LHC era. To avoid code duplication and maintainability issues, heterogeneous computing goes hand in hand with performance portability libraries. Many libraries have been tested and alpaka has been the final choice for Run 3, being the one that meets all the requirements of the CMS software. The final part of this work focused on the portability library SYCL/oneAPI. It provides a unified programming model that reduces the cost and complexity of developing software for heterogeneous computing systems and has demonstrated its ability to run a single source code on different architectures, such as CPUs, NVIDIA, and Intel GPUs. In principle, it can target FPGAs as well, but they have not been tested within this work. However, the results showed an important gap in performance with respect to the native implementation that, in addition to the compiler bugs and the not yet-supported features, makes SYCL unsuitable for use in production in the CMS software today. The whole work has been carried out in contact with Intel developers who have provided constant support and have been notified about the bugs found. In addition, profiling of the application has been conducted and the possible reasons for the performance gap that came to light have been communicated to them. Considering that the oneAPI library is under continuous development and some improvements have already been noticed with respect to when this work started one year ago, it is expected to improve over time, making it worthwhile to keep an eye on it for the future.

At last, oneAPI has been integrated into alpaka, serving as a novel back-end to target many architectures including CPUs, Intel GPUs, and Intel FPGAs. This integration represents a way to expand the compatibility and utilization of these hardware resources within the CMS software. While this addition marks a noteworthy achievement, the effort will continue, focusing on implementing the remaining features in the back-end and on optimizing its performance to match those of the native implementation.

Acknowledgement

I would like to acknowledge Wahid for all the help and support I received during this year of thesis, both in terms of physics and personal advice. I also want to express my gratitude to Felice, who believed in me since the very beginning and supported me throughout my thesis journey. A special mention goes to Alice for constantly supervising her dad during the thesis corrections. I'd like to extend my thanks to my supervisor, Pietro, who gave me the possibility of spending several months at CERN and allowed me to pursue my own direction. I want to express my gratitude to Andrea for guiding me through bugs and difficulties and for patiently answering all my questions. I'm also grateful to Marco for the precious advice, and to Fabio and Martina for the insightful discussions regarding MTD. Thanks also to Igor and Klaus-Dieter from Intel and to all the people of the alpaka team.

I also want to thank all my amazing friends and in particular Chiara and Giulia; I couldn't have asked for better friends than you. Thank you for always being there to belay me, whether it is on the cliffs or during difficult times. I know you'll never let me fall.

Thanks to all my classmates for sharing lunch breaks, moments of despair, birthday candles, and packets of croccantelle. I'm grateful to all my adventure and climbing friends spread across different countries. A special thanks goes to Michele and Elena for their kindness and for making me feel at home in their beautiful hut.

To my housemates during these recent months, thank you for adding a healthy dose of "Law & Order" to the thesis-writing process and for all the memorable moments we have shared.

Lastly, the biggest thanks goes to my family, whom I thank for allowing me to chase my dreams even if it meant spending a lot of time away from them. Thanks for everything.

Bibliography

[1] Lyndon Evans and Philip Bryant. "LHC Machine". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08001. DOI: 10.1088/1748-0221/3/08/S08001. URL: https://dx.doi.org/10.1088/1748-0221/3/08/S08001.

[2] "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716.1 (2012), pp. 30–61. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2012.08.021. URL: https://www.sciencedirect.com/science/article/pii/S0370269312008581.

[3] ATLAS Collaboration. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (Sept. 2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020. URL: https://doi.org/10.1016%2Fj.physletb.2012.08.020.

[4] ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08003. DOI: 10.1088/1748-0221/3/08/S08003. URL: https://dx.doi.org/10.1088/1748-0221/3/08/S08003.

[5] CMS Collaboration. *The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment*. Tech. rep. Also published by CERN Geneva in 2010. 2008, S08004. DOI: 10.1088/1748-0221/3/08/S08004. URL: https://cds.cern.ch/record/1129810.

[6] ALICE Collaboration. "The ALICE experiment at the CERN LHC". In: *JINST* 3 (2008), S08002. DOI: 10.1088/1748-0221/3/08/S08002.

[7] LHCb Collaboration. "The LHCb Detector at the LHC". In: *JINST* 3 (2008). Also published by CERN Geneva in 2010, S08005. DOI: 10.1088/1748-0221/3/08/S08005. URL: https://cds.cern.ch/record/1129809.

[8] O. Aberle et al. *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*. Tech. rep. Geneva: CERN, 2020. DOI: 10.23731/CYRM-2020-0010. URL: https://cds.cern.ch/record/2749422.

[9] CMS Collaboration. *CMS Physics: Technical Design Report Volume 1: Detector Performance and Software*. Tech. rep. 2006.

[10] CMS Collaboration. *CMS technical design report, volume II: Physics performance*. Tech. rep. 6. 2007, pp. 995–1579. DOI: 10.1088/0954-3899/34/6/S01.

[11] *The Phase-2 Upgrade of the CMS Barrel Calorimeters*. Tech. rep. Geneva: CERN, 2017. URL: https://cds.cern.ch/record/2283187.

[12] CMS Collaboration. *The Phase-2 Upgrade of the CMS Endcap Calorimeter*. Tech. rep. Geneva: CERN, 2017. DOI: 10.17181/CERN.IV8M.1JY2. URL: https://cds.cern.ch/record/2293646.

[13] CMS Collaboration. *Technical proposal for a MIP timing detector in the CMS experiment Phase 2 upgrade*. Tech. rep. Geneva: CERN, 2017. DOI: 10.17181/CERN.2RSJ.UE8W. URL: https://cds.cern.ch/record/2296612.

[14] *The Khoronos SYCL Working Group, SYCL 2020 Specification (revision 7) (2023)*. URL: https://registry.khronos.org/SYCL/specs/sycl-2020/html/sycl-2020.html.

[15] *Intel oneAPI: A Unified X-Architecture Programming Model*. URL: https://www.oneapi.com/.

[16] *LHC / HL-LHC plan*. Jan. 2022. URL: https://hilumilhc.web.cern.ch/sites/default/files/HL-LHC_Janvier2022.pdf.

[17] *HL-LHC main parameters and assumptions*. URL: https://espace.cern.ch/HiLumi/WP2/Wiki/HL-LHC%20Parameters.aspx.

[18] Tai Sakuma. "Cutaway diagrams of CMS detector". In: (2019). URL: https://cds.cern.ch/record/2665537.

[19] "Longer term LHC schedule". In: (Jan. 2022). URL: https://lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm.

[20] CMS Collaboration. "Updates on Performance of Physics Objects with the Upgraded CMS detector for High Luminosity LHC." In: (2016). URL: https://cds.cern.ch/record/2222084.

[21] CMS Collaboration. "Updates on Projections of Physics Reach with the Upgraded CMS Detector for High Luminosity LHC". In: (2016). URL: https://cds.cern.ch/record/2221747.

[22] Burkhard Schmidt. "The High-Luminosity upgrade of the LHC: Physics and Technology Challenges for the Accelerator and the Experiments". In: *J. Phys.: Conf. Ser.* 706.2 (2016), p. 022002. DOI: 10.1088/1742-6596/706/2/022002. URL: https://cds.cern.ch/record/2263093.

[23] D Contardo et al. *Technical Proposal for the Phase-II Upgrade of the CMS Detector*. Tech. rep. Geneva: CERN, 2015. DOI: 10.17181/CERN.VU8I.D59J. URL: https://cds.cern.ch/record/2020886.

[24] CMS Collaboration. *The Phase-2 Upgrade of the CMS Tracker*. Tech. rep. Geneva: CERN, 2017. DOI: 10.17181/CERN.QZ28.FLHW. URL: https://cds.cern.ch/record/2272264.

[25] *The Phase-2 Upgrade of the CMS Level-1 Trigger*. Tech. rep. Final version. Geneva: CERN, 2020. URL: https://cds.cern.ch/record/2714892.

[26] Thomas Reis and for the CMS Collaboration. "CMS ECAL upgrade for precision timing and energy measurements at the High Luminosity LHC". In: *Journal of Physics: Conference Series* 2374.1 (Nov. 2022), p. 012072. DOI: 10.1088/1742-6596/2374/1/012072. URL: https://dx.doi.org/10.1088/1742-6596/2374/1/012072.

[27] *The Phase-2 Upgrade of the CMS Muon Detectors*. Tech. rep. Geneva: CERN, 2017. URL: https://cds.cern.ch/record/2283189.

[28] CMS Collaboration. *The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger*. Tech. rep. Geneva: CERN, 2021. URL: https://cds.cern.ch/record/2759072.

[29] Jia Liu, Zhen Liu, and Lian-Tao Wang. "Enhancing Long-Lived Particles Searches at the LHC with Precision Timing Information". In: *Phys. Rev. Lett.* 122 (13 Apr. 2019), p. 131801. DOI: 10.1103/PhysRevLett.122.131801. URL: https://link.aps.org/doi/10.1103/PhysRevLett.122.131801.

[30] Geonhee Oh. "New opportunities in heavy ion physics at HL-LHC with a MIP Timing Detector at the CMS experiment". In: *PoS* HardProbes2020 (2021), p. 178. DOI: 10.22323/1.387.0178. URL: https://cds.cern.ch/record/2799559.

[31] Marco Rovere et al. "CLUE: A Fast Parallel Clustering Algorithm for High Granularity Calorimeters in High-Energy Physics". In: *Frontiers in Big Data* 3 (2020). DOI: 10.3389/fdata.2020.591315. URL: https://www.frontiersin.org/article/10.3389/fdata.2020.591315.

[32] CMS Offline Software and Computing. *CMS Phase-2 Computing Model: Update Document*. Tech. rep. Geneva: CERN, 2022. URL: https://cds.cern.ch/record/2815292.

[33] *CMSSW Application Framework*. URL: https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCMSSWFramework.

[34] *Intel® oneAPI: Threading Building Blocks (TBB)*. https://github.com/oneapi-src/oneTBB.

[35] CMS Collaboration. "Particle-flow reconstruction and global event description with the CMS detector". In: *Journal of Instrumentation* 12.10 (Oct. 2017), P10003. DOI: 10.1088/1748-0221/12/10/P10003. URL: https://dx.doi.org/10.1088/1748-0221/12/10/P10003.

[36] Milos Dordevic. "The CMS Particle Flow Algorithm". In: *EPJ Web Conf.* 191 (2018), p. 02016. DOI: 10.1051/epjconf/201819102016. URL: https://cds.cern.ch/record/2678077.

[37] Daniele Bertolini et al. "Pileup per particle identification". In: *Journal of High Energy Physics* 2014.10 (Oct. 2014). DOI: 10.1007/jhep10(2014)059. URL: https://doi.org/10.1007%2Fjhep10%282014%29059.

[38] Luigi Calligaris and on behalf of the CMS collaboration. "Status of the Phase-2 Tracker Upgrade of the CMS experiment at the HL-LHC". In: *Journal of Physics: Conference Series* 1690.1 (Dec. 2020), p. 012039. DOI: 10.1088/1742-6596/1690/1/012039. URL: https://dx.doi.org/10.1088/1742-6596/1690/1/012039.

[39] Wolfgang Adam et al. *Track Reconstruction in the CMS tracker*. Tech. rep. Geneva: CERN, 2006. URL: https://cds.cern.ch/record/934067.

[40] Susanna Cucciarelli et al. *Track reconstruction, primary vertex finding and seed generation with the Pixel Detector*. Tech. rep. Geneva: CERN, 2006. URL: https://cds.cern.ch/record/927384.

[41] R. Frühwirth. "Application of Kalman filtering to track and vertex fitting". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 262.2 (1987), pp. 444–450. ISSN: 0168-9002. DOI: https://doi.org/10.1016/0168-9002(87)90887-4. URL: https://www.sciencedirect.com/science/article/pii/0168900287908874.

[42] Felice Pantaleo. "New Track Seeding Techniques for the CMS Experiment". 2017. URL: https://cds.cern.ch/record/2293435.

[43] Felice Pantaleo and Marco Rovere. *The Iterative Clustering framework for the CMS HGCAL Reconstruction*. Tech. rep. Geneva: CERN, 2022. URL: https://cds.cern.ch/record/2806234.

[44] J. Alwall et al. "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations". In: *Journal of High Energy Physics* 2014.7 (July 2014). DOI: 10.1007/jhep07(2014)079. URL: https://doi.org/10.1007%2Fjhep07%282014%29079.

[45] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. "PYTHIA 6.4 physics and manual". In: *Journal of High Energy Physics* 2006.05 (May 2006), p. 026. DOI: 10.1088/1126-6708/2006/05/026. URL: https://dx.doi.org/10.1088/1126-6708/2006/05/026.

[46] S. Agostinelli et al. "Geant4—a simulation toolkit". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)01368-8. URL: https://www.sciencedirect.com/science/article/pii/S0168900203013688.

[47] W Adam et al. "Reconstruction of electrons with the Gaussian-sum filter in the CMS tracker at the LHC". In: *Journal of Physics G: Nuclear and Particle Physics* 31.9 (July 2005), N9–N20. DOI: 10.1088/0954-3899/31/9/n01. URL: https://doi.org/10.1088%2F0954-3899%2F31%2F9%2Fn01.

[48] Abhirikshma Nandi. "New Techniques for Reconstruction in the CMS High Granularity Calorimeter". Presented 16 Jan 2023. 2022. URL: http://cds.cern.ch/record/2854616.

[49] CMS Collaboration. "Description and performance of track and primary-vertex reconstruction with the CMS tracker". In: *Journal of Instrumentation* 9.10 (Oct. 2014), P10009–P10009. DOI: 10.1088/1748-0221/9/10/p10009. URL: https://doi.org/10.1088%2F1748-0221%2F9%2F10%2Fp10009.

[50] Andrea Bocci et al. "Heterogeneous Reconstruction of Tracks and Primary Vertices With the CMS Pixel Tracker". In: *Frontiers in Big Data* (2020). DOI: 10.3389/fdata.2020.601728. URL: https://www.frontiersin.org/articles/10.3389/fdata.2020.601728/full.

[51] Andrea Bocci et al. "Bringing heterogeneity to the CMS software framework". In: *EPJ Web of Conferences* 245 (2020), p. 05009. DOI: 10.1051/epjconf/202024505009. URL: https://doi.org/10.1051%2Fepjconf%2F202024505009.

[52] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. *CUDA*. URL: https://docs.nvidia.com/cuda/.

[53] *AMD ROCm™*. URL: https://rocm.docs.amd.com/en/latest/.

[54] *Highlights - June 2023*. URL: https://top500.org/lists/top500/2023/06/highs/.

[55] *Frontier Supercomputer at Oak Ridge National Laboratory*. URL: https://www.amd.com/en/products/frontier.

[56] *Aurora Supercomputer at Argonne National Laboratory*. URL: https://www.intel.com/content/www/us/en/high-performance-computing/supercomputing/exascale-computing.html.

[57] Matti J. Kortelainen et al. *Evaluating Performance Portability with the CMS Heterogeneous Pixel Reconstruction code*. URL: https://indico.jlab.org/event/459/contributions/11824/attachments/9281/14171/20230511-CHEP23_CMSPortability.pdf.

[58] *alpaka - Abstraction Library for Parallel Kernel Acceleration*. URL: https://github.com/alpaka-group/alpaka.

[59] *Kokkos core libraries*. URL: https://github.com/kokkos/kokkos#readme.

[60] Taylor Childers et al. *Porting CMS Heterogeneous Pixel Reconstruction to Kokkos*. 2021. arXiv: 2104.06573 [physics.comp-ph].

[61] *NVC++ compiler - NVIDIA HPC SDK*. URL: https://docs.nvidia.com/hpc-sdk//compilers/c++-parallel-algorithms/index.html.

[62] *The OpenMP API specification for parallel programming*. URL: https://www.openmp.org/.

[63] Andrea Bocci et al. "Performance portability for the CMS Reconstruction with Alpaka". In: *Journal of Physics: Conference Series* 2438.1 (Feb. 2023), p. 012058. DOI: 10.1088/1742-6596/2438/1/012058. URL: https://dx.doi.org/10.1088/1742-6596/2438/1/012058.

[64] Intel corporation. *Intel llvm*. URL: https://github.com/intel/llvm.

[65] V. Blobel. "A new fast track-fit algorithm based on broken lines". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 566.1 (2006). TIME 2005, pp. 14–17. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2006.05.156. URL: https://www.sciencedirect.com/science/article/pii/S0168900206007996.

[66] AMD Corp. *AMD Ryzen™ 9 5900X processor*. URL: https://www.amd.com/en/product/10461.

[67] NVIDIA Corp. *NVIDIA GeForce GTX 1080 Ti GPU*. URL: https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1080-ti/specifications/.