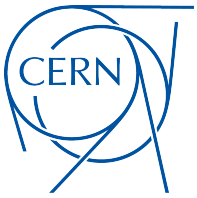
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



New Track Seeding Techniques for the CMS Experiment

Dissertation

zur Erlangung des Doktorgrades
des Department Physik
der Universität Hamburg

vorgelegt von

Felice Pantaleo

aus Bari, Italien

Hamburg
2017

Gutachter/innen der Dissertation:	Prof. Dr. Erika Garutti Dr. Alexander Schmidt
Zusammensetzung der Prüfungskommission:	Prof. Dr. Erika Garutti Dr. Vincenzo Innocente Prof. Dr. Robin Santra Prof. Dr. Peter Schleper Dr. Alexander Schmidt
Vorsitzender der Prüfungskommission:	Prof. Dr. Robin Santra
Datum der Disputation:	27/11/2017
Vorsitzender des Fach-Promotionsausschusses Physik :	Prof. Dr. Wolfgang Hansen
Leiter des Fachbereichs Physik :	Prof. Dr. Michael Potthoff
Dekan der Fakultät MIN :	Prof. Dr. Heinrich Graener

A mia madre.

Contents

Introduction

The CMS experiment [1], together with the ATLAS experiment [2], represents the current program on the Energy Frontier at the Large Hadron Collider [3] at CERN in Geneva, Switzerland. The two general purpose experiments study the production of particles in proton-proton interactions, aiming to better understand the fundamental forces in nature. In 2012 they succeeded in finding the Higgs particle [4] [5], the last fundamental particle predicted by the Standard Model of particle interactions. Both experiments employ a complex setup of tracking, electromagnetic and hadronic calorimeter, and muon detectors inside a strong magnetic field to record the produced particles and their individual momentum and/or energy. Starting from 2017, during CMS Phase-1 [6] and 2 [7], the increased accelerator luminosity with the consequently increased number of simultaneous proton-proton collisions (pile-up) will pose significant new challenges for the CMS experiment.

The reconstruction of the trajectories of charged particles recorded in the silicon pixel and silicon strip detectors is one of the most important components in the interpretation of the detector information of a proton-proton collision [8]. It provides a precise measurement of the momentum of charged particles (muons, electrons and charged hadrons), the identification of interaction points of the proton-proton collision (primary vertex) and decay points of particles with significant lifetimes (secondary vertices).
The increasing complexity of events, due to the growing number of simultaneous proton-proton collisions, will make track reconstruction especially challenging. In fact, algorithms have to explore many combinations before being able to connect traces left by the same particle in the detector.

The quest of significantly reducing the 40 MHz data rate delivered by proton-proton collisions to the detectors, together with the retention of those events which are potentially interesting for searches of new physics phenomena, led to the evaluation of modern

multi-cores and many-cores computer architectures for the enhancement of the existing computing infrastructure used for the event selection, i.e. the High-Level Trigger (HLT) [9]. The objective of the HLT is to apply a specific set of physics selection algorithms and to accept the events with the most interesting physics content. To cope with the incoming event rate, the online reconstruction of a single event for the HLT has to be done within 220 ms on average.

In Chapter 2 of this thesis, the CMS experiment and its experimental apparatus, in the context of which this work was carried out, are presented.

The track reconstruction problem and its application in the CMS online and offline environments is described in Chapter 3. In particular, the implications of the upgrade of the Pixel Detector on track reconstruction are described in this Chapter.

Heterogeneous parallel computing is described in Chapter 4, focusing on the concepts used to develop the parallel algorithms running on Graphics Processing Units (GPUs).

GPUs are massively parallel computer architectures that can be programmed using extensions to the standard C and C++ languages [10]. Their parallelism can be exploited to explore many possible tracks at the same time, hence reducing the time spent in track reconstruction in each event.

Chapter 5 focuses on the development of a parallel algorithm, the Hit-Chain Maker based on the concepts of Cellular Automata and a fast nearest neighbor data structure, namely FKDtree, for track seeding targeted for GPUs.

The results of the integration of the Hit-Chain Maker running on GPUs in the CMS track seeding software framework are discussed in Chapter 6, both from the physics and from the computing performance point of view. The integration of a traditional C++ version of the Hit-Chain Maker is discussed in this chapter.

The traditional version of this innovative algorithm is on average three to four times faster than the existing implementation. It is also more robust with respect to growing pile-up conditions. These innovative algorithms for track reconstruction use for the first time concepts of parallel computing in the CMS experiment. As these algorithms result in a significant improvement both, in physics and computing performance, they have replaced the previous ones for event selection at the HLT as well as in offline track reconstruction.

Zusammenfassung/Einleitung

Das CMS-Experiment [1] repräsentiert, zusammen mit dem ATLAS-Experiment [2], das aktuelle Forschungsprogramm am weltweit leistungsfähigsten Teilchenbeschleuniger, dem Large Hadron Collider (LHC) [3] am CERN in Genf, Schweiz. Die beiden Allzweck-Experimente untersuchen die Produktion von Teilchen in Proton-Proton-Kollisionen, um die grundlegenden Kräfte in der Natur besser zu verstehen. Im Jahr 2012 gelang es ihnen, das Higgs Boson, das letzte fundamentale, vom Standardmodell vorhergesagte Teilchen nachzuweisen [4] [5]. Beide Experimente verwenden eine komplexe Zusammenstellung von Siliziumspurdetektor, elektromagnetischen und hadronischen Kalorimetern und Muon-Detektoren in starken Magnetfeldern, um die produzierten Teilchen und ihre individuellen Impulse und / oder Energien aufzuzeichnen. Seit 2017, während der Umbauten der CMS Phase-1 [6] und Phase-2 [7], stellt die erhöhte Intensität des Beschleunigers, mit der damit verbundenen erhöhten Anzahl gleichzeitiger Proton-Proton-Kollisionen ("Pile-up"), eine für das CMS Experiment bedeutende neue Herausforderung dar.

Die Rekonstruktion der Spuren der geladenen Teilchen, die in den Siliziumpixel- und Siliziumstreifendetektoren aufgezeichnet werden, ist eine der wichtigsten Komponenten der Interpretation der Detektorinformation einer Proton-Proton-Kollision [8]. Sie liefert eine präzise Messung der Impulse der geladenen Teilchen (Myonen, Elektronen und geladene Hadronen), die Identifizierung von Wechselwirkungspunkten der Proton-Proton-Kollision (primärer Wechselwirkungspunkt) und Zerfallspunkte von Teilchen mit signifikanten Lebensdauern (sekundäre Wechselwirkungspunkte).
Die auf Grund der wachsenden Zahl von gleichzeitigen Proton-Proton-Kollisionen zunehmende Komplexität der Ereignisse macht die Spurrekonstruktion besonders anspruchsvoll. Die Algorithmen müssen viele Kombinationen untersuchen, um die Signale, die von demselben Teilchen im Detektor zurückgelassen werden, korrekt zu Spuren zu verbinden.

Die Notwendigkeit der signifikanten Verringerung der 40 MHz-Datenrate, die durch

Proton-Proton-Kollisionen an die Detektoren geliefert wurde, unter gleichzeitiger Beibehaltung jener Ereignisse, die für die Suche nach neuen physikalischen Phänomenen potentiell interessant sind, führte zur Untersuchung von modernen Mehr-Kern Rechnerarchitekturen für die Erweiterung der vorhandenen Recheninfrastruktur, die für die Ereignisauswahl verwendet wird, der so genannte "High-Level Trigger" (HLT) [9]. Das Ziel des HLT ist es, einen bestimmten Satz von Auswahlalgorithmen anzuwenden und idealerweise nur die Ereignisse mit dem interessantesten Physikinhalt zu akzeptieren. Um die eingehende Ereignisrate zu bewältigen, muss die Rekonstruktion eines einzelnen Ereignisses im HLT durchschnittlich in 220 ms durchgeführt werden.

In Kapitel 2 dieser Arbeit wird das CMS-Experiment, in dessen Rahmen diese Arbeit durchgeführt wurde, vorgestellt.

Das Spurrekonstruktionsproblem und seine Anwendung in den CMS Online- und Offline-Umgebungen wird in Kapitel 3 beschrieben. Insbesondere werden die Auswirkung des Umbaus des Pixeldetektors auf die Spurrekonstruktion in diesem Kapitel beschrieben.

Heterogenes Parallelrechnen wird in Kapitel 4 beschrieben. Es konzentriert sich auf die Konzepte, die zur Entwicklung der parallelen Algorithmen verwendet werden, die auf den Grafikeinheiten ("Graphics Processing Units", GPUs) laufen.

GPUs sind massiv parallele Rechnerarchitekturen, die mit Erweiterungen der Standard-C und -C++ Sprachen [10] programmiert werden können. Ihre Parallelität kann genutzt werden, um viele mögliche Spuren gleichzeitig zu erforschen und damit die Zeit zu reduzieren, die mit der Spurrekonstruktion in jedem Ereignis verbracht wird.

Kapitel 5 konzentriert sich auf die Entwicklung eines neuen, parallelen Algorithmus ("Hit-Chain Maker") zum Finden der Startpunkte der Teilchenspuren, der auf den Konzepten von Zellulären Automaten basiert, sowie eine schnelle, "nächster-Nachbar" Datenstruktur ("FKDtree").

Die Ergebnisse der Einbindung des Hit-Chain Maker, der auf GPUs innerhalb des Spurfindungsprogramms des CMS-Software-Frameworks läuft, werden in Kapitel 6 diskutiert, sowohl aus der Sicht der Rechenleistung als auch der des Einflusses auf die Physik-Resultate. Die Integration einer traditionellen C++ -Version des Hit-Chain Makers wird ebenfalls in diesem Kapitel besprochen.
Die traditionelle Version dieses innovativen Algorithmus ist durchschnittlich drei- bis

viermal schneller als die bestehende Implementierung. Sie ist auch robuster in Bezug auf die erhöhte Anzahl gleichzeitiger Proton-Proton-Kollisionen. Die in dieser Arbeit entwickelten Algorithmen zur Spurrekonstruktion führen zum ersten Mal Konzepte zur Nutzung paralleler Rechnerarchitekturen im CMS-Experiment ein. Diese neuen Algorithmen haben die bisherigen Algorithmen zur Spurrekonstruktion sowohl im HLT als auch in der Offline-Spurrekonstruktion ersetzt, da sie in allen Belangen deutlich leistungsfähiger sind.

Chapter 1

The Standard Model of Elementary Particles

1.1 Brief History of Particle Physics

The *Standard Model* of particle physics (SM) is a theory that describes the elementary particles and their interactions, except for the gravitational one, which has negligible effects at microscopic scales.

The journey that has led mankind to develop the concepts of particles and eventually to the formulation of the Standard Model started in the fifth century B.C. in Greece by Leucippus and his pupil Democritus.
Their ontology was based on the concepts of atoms[1] and vacuum, the fundamental elements of which the universe is made.
Aristotle in the *Metaphysics*, writes about atoms and vacuum referring to them as *being* and *non-being* [11]:

> "Leucippus and his associate Democritus say that the full and the empty are the elements, calling the one 'being' and the other 'non-being' — the full and solid being 'being', the empty non-being (whence they say 'being' no more is than 'non-being', because the solid no more 'is' than the empty); and they make these the material causes of things."

[1] ἄτομος, àtomos, the Greek word for indivisible.

The theory of atoms was rejected by Aristotle and for this reason it was ignored for two millennia.

The hypothesis of the discrete composition of things started to come back at the beginning of the XIX-century, with the studies by R. Brown on the motion of particles contained in the pollen of plants [12]. In 1799, the chemist Joseph-Louis Proust found out that the elements which constitute a chemical compound are always present in fixed ratio of mass [13]. This is nowadays known as the *law of definite proportions*.
In 1811, Italian physicist and mathematician A. Avogadro made a hypothesis which stated that equal volumes of all gases, at the same temperature and pressure, have the same number of molecules:

$$V = kN \tag{1.1}$$

where V is the volume, N is the number of molecules and k a constant.
Starting from 1865, the hypothesis made by Avogadro was proved by Loschmidt, van der Waals, Roentgen, Rayleigh, Plank, Wilson, Thomson, and Einstein, who, with different experiments, were able to measure the Avogadro constant, as the number of Carbon-12 atoms in 0.012 kg of Carbon-12.

It is after the discovery of electricity that particle physics history started to move with strong pace. During his studies on cathodic rays, J. J. Thomson noticed that these rays were able to discharge electroscopes. He demonstrated that particles that make up cathodic rays are deviated by electric and magnetic fields. Thomson was then able to measure the ratio e/m between the charge e and the mass m of these particles [14]. The first elementary particle had been discovered and it was called *electron*, nowadays indicated with e^-. The experiment aimed to the absolute measurement of the charge of the electron was carried out in 1910 by Millikan [15].
To justify the neutral charge and much higher mass of atoms with respect to electrons, Thomson proposed a model in which electrons are enclosed in a positively charged paste. However, this model was rejected with Rutherford's experiments on scattering [16]. In fact, the outcome of these experiments was that most of the mass is concentrated in the center of atoms.
These experiments inspired Bohr's model of the hydrogen atom [17], in which an electron orbits around a positively charged particle called proton. This model showed an unprecedented agreement with experimental data. However, discrepancies on the

value of the mass of heavier nuclei appeared. These discrepancies are explained in 1932 by Chadwick's hypothesis on the existence of a heavy neutral particle in the nucleus of atoms, called neutron [18]. In 1922, the Stern and Gerlach experiment demonstrated the fact that the electron has an intrinsic angular momentum, i.e. *spin* [19].

Max Planck' studies on black-body radiation [20] and Albert Einstein's work on the photoelectric effect [21] demonstrated that light is absorbed in *quanta* (discrete quantities) of energy, called *photons*.

The theory of *Special Relativity*, created in 1905 by Albert Einstein postulates that [22], [23]:

> "...light is always propagated in empty space with a definite velocity [speed] c which is independent of the state of motion of the emitting body."
> "If a system of coordinates K is chosen so that, in relation to it, physical laws hold good in their simplest form, the same laws hold good in relation to any other system of coordinates K' moving in uniform translation relatively to K."

It took more than two decades to combine Special Relativity and Quantum Mechanics together. A fundamental step towards a unified theory was done in 1928 by Paul Dirac [24]. In order to describe the structure of the atoms with point-charge electrons, Dirac blends concepts of the special relativity and quantum mechanics together. However, Dirac's equation admits negative energy solutions.
While studying the production of secondary particles produced by the interaction of cosmic rays in a cloud chamber placed in the electromagnet of Caltech's Guggenheim Aeronautical Laboratory, Carl D. Anderson noticed something unexpected in the results. Many pictures showed simultaneous ejection of an electron and a positive particle much lighter than a proton. Further investigations led Anderson to the discovery of the *positron* (anti-electron) [25] — what seemed to be a deficiency in Dirac's equation became its greatest success.

Quantum Electrodynamics (QED) was developed in the end of the 1940s thanks to the contributions of Sin-Itiro Tomonaga [26], Julian Schwinger [27] [28] and Richard Feynman [29] to describe, using quantum mechanics and special relativity, all those phenomena that involve charged particles interacting by means of electromagnetic force.

It is one of the most accurate theories existing today, predicting quantities like the anomalous magnetic moment of the electron and the Lamb shift of the energy levels of hydrogen with great precision.

The first half of the XX-century gave birth to the research on the weak interaction and neutrino physics. The existence of neutrinos ν was postulated by Wolfgang Pauli in 1930 in his famous "Open letter to radioactive persons" [30]. The existence of neutrinos was required to ensure conservation of energy, momentum and angular momentum in the continuous spectrum observed in beta decays.
In 1934, Enrico Fermi developed a comprehensive theory of radioactive decays [31]. This theory includes Pauli's hypothetical particle that was renamed *neutrino*, meaning in Italian: "little neutral one". With the inclusion of the neutrino, Fermi's theory accurately explains many experimentally observed results.
In 1946, Bruno Pontecorvo suggested a radiochemical method of detecting a free neutrino after it had been emitted in a beta decay [32]. Pontecorvo's method was based on the observation of decay of daughter nucleus produced in the reaction:

$$\nu + (A, Z) \rightarrow e^- + (A, Z + 1). \tag{1.2}$$

In 1956, Clyde Cowan and Fred Reines discovered a particle fitting the expected characteristics of the antineutrino produced in a nuclear reactor. The antineutrino was detected using the inverse beta decay:

$$\bar{\nu} + p \rightarrow e^+ + n. \tag{1.3}$$

The interaction keeping protons and neutrons together in nuclei was still mysterious at that time. The Japanese physicist Hideki Yukawa made the hypothesis that a quantum of the nuclear field had a similar nature to the quantum of the electromagnetic field, the photon. His calculations predicted that the mass of this quantum had to be about 200 times larger than the mass of the electron.
In 1937, Anderson and Neddermeyer, while studying the interaction of cosmic rays with matter in a magnetic field, found particles with a value of mass between the mass of the electron and the mass of the proton. These particles were named mesotrons, and later mesons.
However, the results of an experiment carried out by Marcello Conversi, Ettore Pancini and Oreste Piccioni [33] were in disagreement with the hypothesis of a strongly interacting

Yukawa particle. In 1947, Enrico Fermi, Edward Teller and Victor Weisskopf [34] showed that there were about 12 orders of magnitude difference between the predicted capture time for a negative Yukawa particle and the results of the experiment for negative particles.
It was found that the mesotron, renamed π meson (pion), is not the strongly interacting Yukawa particle. Furthermore, Cesare Lattes, Beppo Occhialini and Cecil Powell [35], discovered that the pions decay into muons, which decay to produce electrons.

In 1947, Bruno Pontecorvo suggested an analogy between beta processes and those of emission and absorption of a muon [36] indicating a universality of the weak interaction. In 1959, at the Kiev Conference, he expressed the idea that experiments with high-energy accelerators could prove whether a neutrino produced in association with a muon ν_μ is different from a neutrino produced in association with an electron ν_e.
Experiments at Brookhaven National Laboratory, carried out by Leon M. Lederman, Melvin Schwartz and Jack Steinberger [37], demonstrated the doublet structure of the through the discovery of the muon neutrino ν_μ.
The τ particle, a heavier version of the electron and muon, has been discovered at SLAC almost fifteen years later [38]. Its corresponding neutrino ν_τ was discovered at Fermilab, by the DONUT experiment in the year 2000 [39].

Following the World War II, research on cosmic rays and with accelerators started to develop quickly. Tens of new particles had been discovered, making up the so called "zoo of particles". Physicists started to look for relations between particles, searching for a "Table of Elements" for particle physics. This research revealed many approximate symmetries and similarities.

Gell-Mann and Nishijima introduced a new quantum number, the strangeness s. Strangeness is conserved in strong interactions but not in weak interactions. In 1961, Murray Gell-Mann found out that many discovered particles could be arranged in a two-dimensional schema called the *Eightfold Way* (Figure 1.1).

In 1964, Gell-Mann [40] and George Zweig proposed independently that the reason for this symmetry is that strongly-interacting particles are made of three fundamental entities. Gell-Mann called these fundamental particles *quarks*[2]. To distinguish the

[2]Gell-Mann took inspiration for the term *quark* from James Joyce's Finnegans Wake phrase: "Three quarks for Muster Mark".

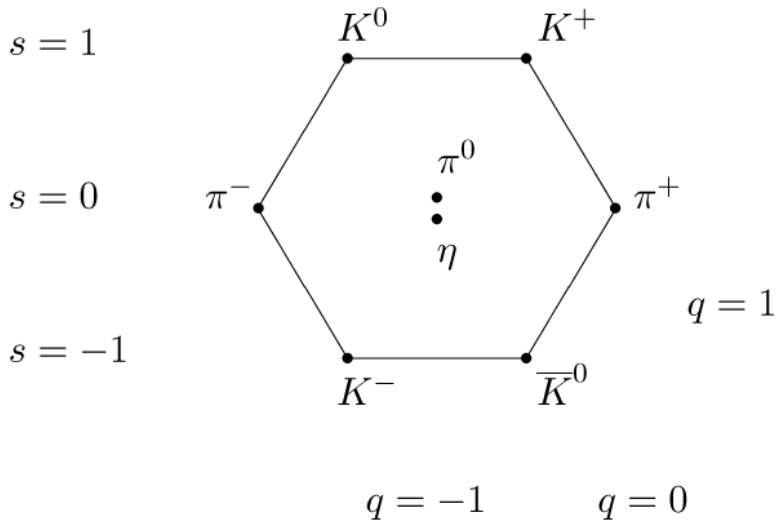


Figure 1.1: Gell-Mann's Eightfold Way which arranged the particles based on their charge and strangeness. Particles on the same horizontal line have the same strangeness (s). Particles on the same diagonal line have the same charge (q).

properties of these different elementary particles, they were called *up* (u), *down* (d), *strange* (s). Protons and neutrons are combinations respectively of uud and udd. The charge of quarks is a fraction of the charge e: $\frac{2}{3}e$ for the u-quark, $-\frac{1}{3}e$ for the d- and s-quarks.
The violation of Pauli's exclusion principle in the case of the $\Delta^{++}(uuu)$ particle, was solved in 1965 by Moo-Young Han and Yoichiro Nambu [41]. They introduced three different color charges that quarks could have. Conventionally these colors are called *red*, *green*, *blue*. The evidence of the existence of an internal structure in nucleons, the so called *parton model*, is given by deep inelastic scattering experiments.

In 1970, Sheldon Lee Glashow, John Iliopoulos and Luciano Maiani, made the hypothesis of generations of quarks and leptons (ν_e, e), (ν_μ, μ), (u, d), (c, s). This hypothesis required the existence of the quark charm. A bound state $J/\psi(c\bar{c})$ was discovered independently in 1974 by Samuel Ting at the Brookhaven National Laboratory [42] and by Burton Richter at SLAC [43], in what has become the *November Revolution*. In 1977, Lederman discovered a meson Υ, bound state $(b\bar{b})$ of *beauty*-quarks [44].
The third quark generation was completed when the CDF [45] and D0 [46] collaborations at the Tevatron collider at Fermilab, discovered the *top*-quark, in 1995.

Figure 1.2: A three-jets event at the TASSO experiment, DESY.

In the 1970s the theory of quarks and colors, the *Quantum Chromodynamics* (QCD), was the best candidate for describing the nuclear interactions. However, the proof of the existence of its mediating boson, the *gluon*, was long in coming. Gluons carry color charge, because they couple colored quarks. They exist in eight color configurations. In a process like:

$$e^+e^- \rightarrow q\bar{q}, \tag{1.4}$$

gluons can be radiated by a quark. As a result, the hadron shower (*jet*) produced by the quark radiating the gluon would broaden, producing a third shower.
A proof of the existence of three-jets events was found by the TASSO collaboration [47] at the PETRA accelerator at DESY in 1979 (Figure 1.2).

In 1967, to solve the discrepancy at high-energy between Fermi's theory of weak interaction and experimental results, Sheldon Glashow, Steven Weinberg and Abdus Salam independently propose a theory to unify weak and electromagnetic interaction. Their theory required the existence of two electrically charged ($W^{\pm}$) and one neutral massive bosons (Z^0).
In 1974, the Gargamelle experiment at CERN observed the weak neutral current [48].
The discovery of the $W^{\pm}$ and Z^0 bosons came almost ten years later by Carlo Rubbia who was leading the UA1 collaboration and Simon van der Meer who was working at the CERN SPS accelerator [49].

Later experiments at CERN's Large Electron-Positron (LEP) accelerator determined that no additional light neutrinos beyond the three already known can exist, by measuring the width of the Z^0-boson [50].

The existence of massive bosons, communicators of the weak interaction and massive fermions, could only be explained with the Spontaneous Symmetry Breaking mechanism proposed by François B. Englert, Robert Brout [51], Gerald Guralnik, Carl Hagen, Thomas Kibble [52] and Peter Higgs [53]. This mechanism required the existence of a neutral particle, the Higgs boson H, discovered at CERN in 2012 by the CMS and ATLAS collaborations [4] [5].

1.2 Particles and Fields in the Standard Model

The SM is a Quantum Field Theory (QFT) [54] in which matter is described in terms of spin-1/2 fermions while their interactions are mediated by bosons with spin 1. These bosons are photons, gluons, W and Z. They mediate respectively the electromagnetic, strong, and weak interactions.

Fermions are distributed in three generations, each made of:

- two quarks with electric charge respectively $\frac{2}{3}e$ and $-\frac{1}{3}e$ (Table 1.1),
- two leptons, one having electric charge $-1e$, the other with no electric charge (Table 1.2).

Table 1.1: Quark properties [55].

Quark	Mass	Charge [e]
u	$2.2^{+0.6}_{-0.4}$ MeV/c^2	$\frac{2}{3}$
d	$4.7^{+0.5}_{-0.4}$ MeV/c^2	$-\frac{1}{3}$
c	1.27 ± 0.03 GeV/c^2	$\frac{2}{3}$
s	96^{+8}_{-4} MeV/c^2	$-\frac{1}{3}$
t	$173.21 \pm 0.51 \pm 0.71$ GeV/c^2	$\frac{2}{3}$
b	$4.18^{+0.04}_{-0.03}$ GeV/c^2	$-\frac{1}{3}$

Table 1.2: Lepton properties [55].

Lepton	Mass	Charge	L_e	L_μ	L_τ
e^-	$0.5109989461 \pm 0.0000000031$ MeV	$-1e$	1	0	0
μ^-	$105.6583745 \pm 0.0000024$ MeV	$-1e$	0	1	0
τ^-	1776.86 ± 0.12 MeV	$-1e$	0	0	1
ν_e	< 2 eV	0	1	0	0
ν_μ	< 0.19 MeV	0	0	1	0
ν_τ	< 18.2 MeV	0	0	0	1

In the SM, the quark flavor is always conserved in electromagnetic and strong interactions, but this is not always true in weak interactions. For leptons, flavor is conserved within the same generation. However, only the total lepton number is conserved in neutrino oscillation processes [56].

A gauge theory is a QFT in which particles have a local symmetry that operates on the particles' internal degrees of freedom.
The gauge group of the SM is given by the direct product:

$$G_{SM} = \mathrm{SU(3)_C} \otimes \mathrm{SU(2)_L} \otimes \mathrm{U(1)_Y}. \tag{1.5}$$

1.2.1 Strong interaction

The $\mathrm{SU(3)_C}$ group is used to describe the *Quantum Chromo-Dynamics* (QCD).
In QCD, strongly interacting particles have a color charge.
The $\mathrm{SU(3)_C}$ group is non-abelian. Corresponding to the 8 generators, gluons carry color and are self-interacting. A possible representation of these eight gluons is the following:

$$r\bar{g}, r\bar{b}, g\bar{b}, g\bar{r}, b\bar{r}, b\bar{g}, \frac{1}{\sqrt{2}}(r\bar{r} - g\bar{g}), \frac{1}{\sqrt{6}}(r\bar{r} + g\bar{g} - 2b\bar{b}). \tag{1.6}$$

In the ninth representation, $\frac{1}{\sqrt{3}}(r\bar{r} + g\bar{g} + b\bar{b})$, all the colors are neutralized by their respective anti-colors, such that it does not exist.

Quarks and anti-quarks are represented by Dirac fields in the simplest representations of this group 3 and $\bar{3}$. These can be multiplied to produce a gauge-invariant bilinear singlet $q\bar{q}$ (corresponding to mesons), and the octet $\bar{q}\gamma^\mu\lambda^a q$ (corresponding to baryons) [54].

Given a massless quark spinor field ψ and the gluon field tensor $G^a_{\mu\nu}$, one can write the Lagrangian for QCD, a function of the fields in the system and their derivatives as follows:

$$\mathcal{L}_{QCD} = \bar{\psi} i\gamma^\mu D_\mu \psi - \frac{1}{4} G^a_{\mu\nu} G^{\mu\nu}_a. \tag{1.7}$$

The covariant derivative is given by:

$$D_\mu = \delta_\mu + \frac{i}{2} g_s \lambda_a G^a_\mu \tag{1.8}$$

where λ_a are the Gell-Mann matrices, and g_s is the strong coupling constant. The coupling constant $\alpha_s = \frac{g_s^2}{4\pi}$ is called *running constant* because its value depends on the value of the transferred momentum.
In the high-energy regime (at short distances), interactions between quarks and gluons can be successfully described by a perturbative theory: color interaction becomes weak. This weakness causes the so called *asymptotic freedom*.
The increased value of the coupling constant is the reason of the so-called *Confinement of quarks* [57].
The confinement effect appears at the energy scale of $\Lambda_{QCD} \sim 200$ MeV, which is the light hadrons mass scale. At low energies, the strong interaction becomes important and perturbative methods become less useful. Non-perturbative approaches like *Lattice QCD* [58], a gauge theory formulated on a grid of points in space and time, are used to solve QCD approximately.

All the free particles observed are colorless and this is allowed by the *hadronization* process. Hadronization is the process of formation of hadrons: when a quark or gluon is extracted from the original hadron, it combines with quarks or anti-quarks produced in pairs from the vacuum.

1.3 Electroweak interaction

As described in Section1.1, the electromagnetic and weak interactions can be treated together. The gauge theory which describes the electroweak interaction is based on the invariance for $\mathrm{SU(2)_L} \otimes \mathrm{U(1)_Y}$ gauge transformations.
The $\mathrm{SU(2)_L}$ group is employed to describe the *weak isospin*. Three gauge bosons W^j

correspond to the $\mathrm{SU(2)_L}$ group's three generators. Fermions are represented in the constituents of the electroweak interaction:

- fermions of left-handed chirality form the isospin doublets ($T = 1/2$)

$$\begin{pmatrix} \nu \\ e_L \end{pmatrix}, \begin{pmatrix} u_L \\ d_L \end{pmatrix} \tag{1.9}$$

  for each of the three fermionic generations;

- fermions of right-handed chirality form the isospin singlets ($T = 0$) and do not interact with W^j bosons:

$$e_R, u_R, d_R. \tag{1.10}$$

  Charged leptons and quarks interact electroweakly with both the left-handed and right-handed components. In the SM, only the left-handed chirality component is present for neutrinos.

The *weak hypercharge* is described by the $\mathrm{U(1)_Y}$ abelian group. The mediator of the interaction is the B neutral boson.
The electric charge Q of a particle is hence given by the sum $Q = T_3 + Y$.
The Lagrangian for the electroweak interaction can be finally written as:

$$\mathcal{L}_{EW} = \bar{\psi} i\gamma^\mu D_\mu \psi - \frac{1}{4} W^a_{\mu\nu} W_a^{\mu\nu} - B_{\mu\nu} B^{\mu\nu}. \tag{1.11}$$

The covariant derivative for the left-handed doublets is given by:

$$D^L_\mu = \delta_\mu + \frac{i}{2} g \tau_a W^a_\mu + \frac{i}{g'} Y B_\mu, \tag{1.12}$$

instead for the right-handed singlets:

$$D^R_\mu = \delta_\mu + \frac{i}{g'} Y B_\mu, \tag{1.13}$$

where g and g' are the coupling constants of the W^j and B bosons respectively. Physical fields A_μ, $W^\pm_\mu$ and Z, arise as linear combinations of W^j and B fields:

$$A_\mu = \sin\theta_W W^3_\mu + \cos\theta_W B_\mu, \tag{1.14}$$

Table 1.3: Properties of the gauge bosons [55].

Gauge Boson	Mass $[GeV/c^2]$	Charge $[e]$
γ	0	0
g	0	0
Z^0	91.1876 ± 0.0021	0
$W^{\pm}$	80.385 ± 0.015	± 1

$$Z_\mu = -\sin\theta_W B_\mu + \cos\theta_W W_\mu^3, \tag{1.15}$$

$$W_\mu^{\pm} = \frac{1}{\sqrt{2}}(W_\mu^1 \pm W_\mu^2). \tag{1.16}$$

The g and g' coupling constants are connected through the Weinberg mixing angle θ_W as follows:

$$e = g\sin\theta_W = g'\cos\theta_W, \tag{1.17}$$

where e is the electron electric charge.

The properties of the gauge bosons are summarized in Table 1.3.

1.3.1 Spontaneous symmetry breaking and the Higgs mechanism

The weak interaction acts at short distance. This requires the gauge bosons to be massive, which makes them incompatible with the electroweak gauge symmetry demanding fermions and gauge bosons to be massless.
Massless gauge bosons are evidently in contrast with the experimental observation of $W^{\pm}$ and Z^0 bosons [48] [49].
Their mass will be explained in the following by means of the Spontaneous Symmetry Breaking mechanism.

$U(1)$ spontaneous symmetry breaking

Consider a charged scalar particle described by the field ϕ in a theory with $U(1)$ local symmetry. The Lagrangian can be written as follows:

$$\mathcal{L} = (D_\mu \phi)^\dagger (D^\mu \phi) + \mu^2 \phi^\dagger \phi - \lambda (\phi^\dagger \phi)^2 - \frac{1}{4} F_{\mu\nu} F^{\mu\nu}. \tag{1.18}$$

with

$$D_\mu \phi = (\partial_\mu - igA_\mu)\phi, \tag{1.19}$$

and

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu \tag{1.20}$$

A local transformation in $U(1)$ has the following form:

$$\phi(x) \to \phi'(x) = e^{-i\alpha(x)} \phi(x). \tag{1.21}$$

The derivative instead does not transform as a phase $U(1)$ transformation. In fact:

$$\partial_\mu \phi(x) \to \partial_\mu \phi'(x) = e^{-i\alpha(x)} (\partial_\mu \phi(x) - i\partial_\mu \alpha(x)). \tag{1.22}$$

Introducing a gauge field $A_\mu(x)$ that forms a covariant derivative

$$D_\mu \phi = (\partial_\mu - igA_\mu)\phi, \tag{1.23}$$

and by requiring that

$$A_\mu(x) \to A'_\mu(x) = A_\mu(x) - \partial_\mu \alpha(x)), \tag{1.24}$$

we now obtain that the covariant derivative transforms as a phase $U(1)$ transformation in Maxwell's equations:

$$(D_\mu \phi)' = e^{-i\alpha} (D_\mu \phi). \tag{1.25}$$

In this case the gauge field is massless because the mass term $A_\mu A^\mu$ is not gauge invariant.

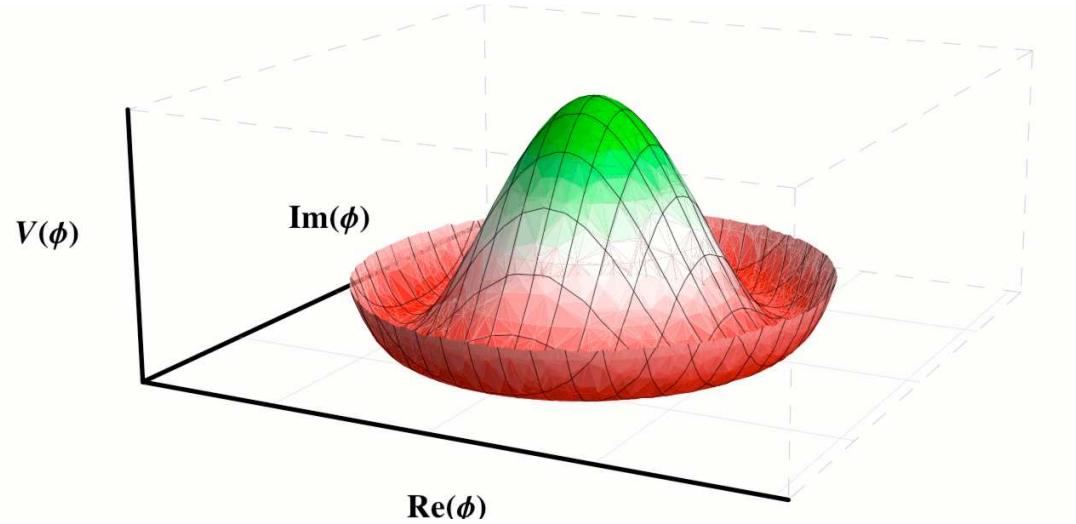


Figure 1.3: The Mexican hat potential for $\mu^2 > 0$.

The potential term V of the Lagrangian depends only on $\rho^2 = \phi^\dagger\phi$,

$$V(\rho) = -\mu^2\rho^2 + \lambda\rho^4. \tag{1.26}$$

If $\mu^2 > 0$, the minimum of the potential is found on the circumference $\rho^2 = \frac{\mu^2}{2\lambda}$ (Figure 1.3).

By dividing the ϕ complex field into its real and imaginary components,

$$\phi = \frac{1}{2}(\phi_1 + i\phi_2), \tag{1.27}$$

the vacuum expectation values for the fields are $\langle 0|\phi_1|0\rangle = v$ and $\langle 0|\phi_2|0\rangle = 0$, with $v^2 = \frac{\mu^2}{\lambda}$. ϕ_2 is a massless Goldstone boson [59].

The mass term for the gauge boson can be extracted from the covariant derivative

$$|D_\mu\phi|^2 = |(\partial_\mu - igA_\mu)\phi|^2 = ... + \frac{g^2v^2}{2}A^\mu A_\mu + ... \tag{1.28}$$

hence giving a value of the mass $m = gv$.

The complex field ϕ can be written in terms of two scalar fields in polar coordinates:

$$\phi = \frac{1}{\sqrt{2}}(v + \eta(x))e^{i\xi/v} \tag{1.29}$$

One can introduce a gauge transformation, to remove the dependence over ξ.
In this case one obtains:

$$B_\mu = A_\mu - \frac{1}{gv}\partial_\mu \xi, \tag{1.30}$$

ξ has become the longitudinal component of B_μ, which has acquired mass by "eating" the Goldstone boson.

1.3.2 Spontaneous symmetry breaking in $\mathrm{SU(2)_L} \otimes \mathrm{U(1)_Y}$

The same conclusions can be extrapolated to the electroweak gauge group $\mathrm{SU(2)_L} \otimes \mathrm{U(1)_Y}$.
The Higgs field Φ before symmetry breaking is a $\mathrm{SU(2)_L}$ doublet with $Y = \frac{1}{2}$

$$\Phi = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} \tag{1.31}$$

The part of the Lagrangian containing Φ can be written as:

$$\mathcal{L}_\oplus = (D_\mu \Phi)^\dagger (D^\mu \Phi) + \mu^2 \Phi^\dagger \Phi - \lambda(\Phi^\dagger \Phi)^2. \tag{1.32}$$

The covariant derivative of Φ can be written as follows:

$$D_\mu \Phi = (\partial_\mu - \frac{i}{2} g \tau_a W_\mu^a + \frac{i}{2} g' B_\mu)\Phi \tag{1.33}$$

After spontaneous symmetry breaking, the evaluation of the minimum of the potential gives:

$$-\mu^2 + 2(\Phi^\dagger \Phi) = 0. \tag{1.34}$$

By choosing the unitary gauge for which:

$$\langle 0|\Phi|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v \end{pmatrix} \tag{1.35}$$

where $v = \sqrt{\frac{\mu^2}{\lambda}}$, the expansion of the fields around the minimum can be obtained:

$$\Phi = \Phi' + \langle 0|\Phi|0\rangle = \begin{pmatrix} \phi'^{+} \\ \phi'^{0} + \frac{v}{\sqrt{2}} \end{pmatrix}. \tag{1.36}$$

The terms that have quadratic dependence from v in the Lagrangian will give mass to the gauge bosons:

$$\begin{aligned} \mathcal{L}_{\oplus} &= \frac{v^2}{8}\{g^2((W_\mu^1)^2 + (W_\mu^2)^2) + (gA_\mu^3 - g'B_\mu)^2\} + ... \\ &= M_W^2 W^{+\mu} W_\mu^- + \frac{1}{2} M_Z^2 Z^\mu Z_\mu + ... \end{aligned} \tag{1.37}$$

where:

$$\begin{aligned} W_\mu^\pm &= \frac{1}{\sqrt{2}}(W_\mu^1 \mp i W_\mu^2), \text{ with } M_W = \frac{gv}{2}, \\ Z_\mu^0 &= \frac{1}{\sqrt{g^2 + g'^2}}(gW_\mu^3 - g'B_\mu) \text{ with } M_Z = \frac{v}{2}\sqrt{g^2 + g'^2}, \\ A_\mu &= \frac{1}{\sqrt{g^2 + g'^2}}(gW_\mu^3 + g'B_\mu) \text{ with } M_A = 0. \end{aligned} \tag{1.38}$$

The $W_\mu^\pm$ and Z_μ^0 fields describe the W and Z bosons. The A_μ field describes the photon.

Just like it was done in the case of the abelian group $U(1)$, one can write the scalar field Φ in the form:

$$\Phi = e^{i\vec{\tau}\cdot\vec{\xi}(x)/v} \begin{pmatrix} 0 \\ v + \eta(x) \end{pmatrix} \tag{1.39}$$

where $\vec{\xi}(x)$ are Goldstone bosons, and apply a gauge transformation such that:

$$\begin{aligned} \Phi' = U(\vec{\xi})\Phi &= \begin{pmatrix} 0 \\ v + \eta(x) \end{pmatrix}, \\ \frac{\vec{\tau}\cdot\vec{W'_\mu}}{2} = U(\vec{\xi})\frac{\vec{\tau}\cdot\vec{W}_\mu}{2}U^{-1}(\vec{\xi}) &- \frac{i}{g}(\partial_\mu U(\vec{\xi}))U^{-1}(\vec{\xi}), \\ B'_\mu &= B_\mu \end{aligned} \tag{1.40}$$

where $U(\vec{\xi}) = e^{i\vec{\tau}\cdot\vec{\xi}(x)/v}$.
Only one scalar component of the Φ field is left and it is referred to as the *Higgs Boson*.

The Higgs mechanism can be applied to provide quarks and leptons with mass as well. The coupling of the Higgs field with the fermion states leads to the mass term:

$$m_f = \frac{\sqrt{2} g_f M_W \sin\theta_W}{e}, \tag{1.41}$$

where g_f is the coupling constant between the Higgs boson and $f\bar{f}$, which is proportional to m_f.

1.4 Production and decay channels of the SM Higgs boson

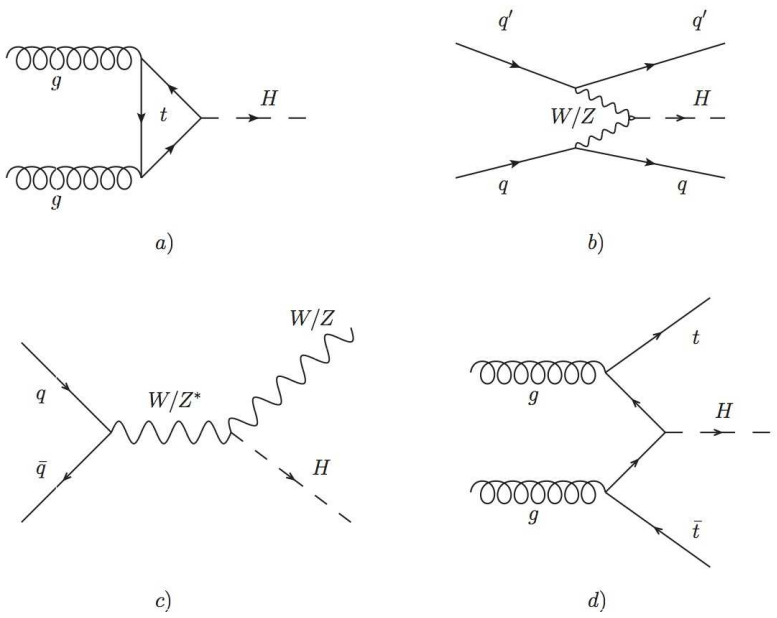
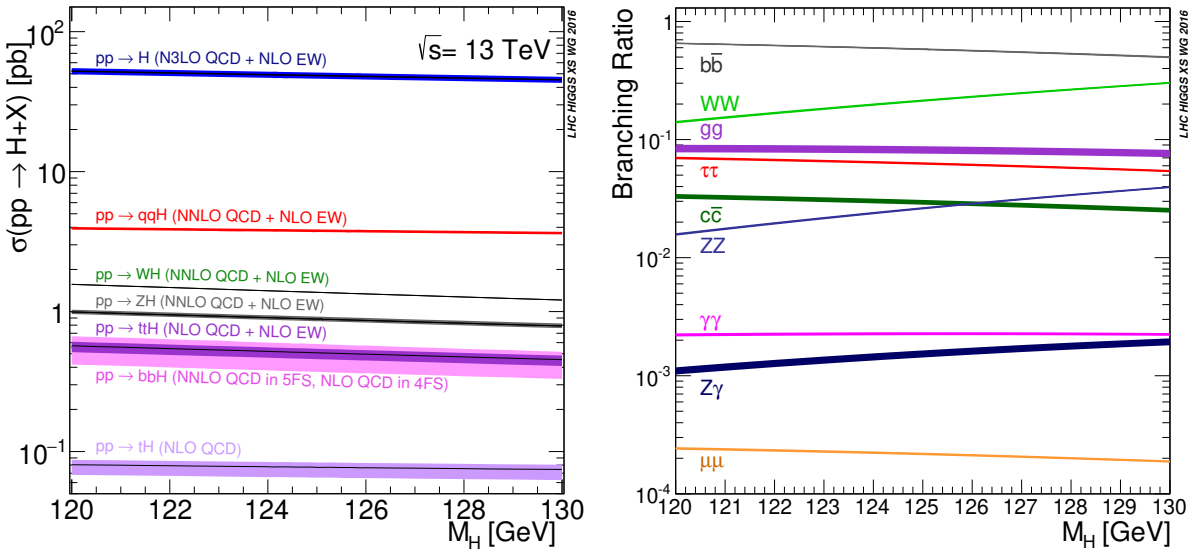


Figure 1.4: Feynman diagrams for the different Higgs production mechanisms: (a) gluon-gluon fusion, (b) vector boson fusion, (c) associated production with vector boson, (d) associated production with $t\bar{t}$ pair.

In the SM, the production of Higgs bosons at hadron colliders happens with four mechanisms (Figure 1.4):

- gluon fusion $gg \to H$, mediated by a virtual quark loop in which the main contribution comes from the top quark due to its large Yukawa coupling;

- vector-boson fusion $qq \to H + 2$ jets;
- associated production with a W or Z boson, which allows to measure the Higgs boson couplings to weak gauge bosons;
- associated production with a $t\bar{t}$ pair.

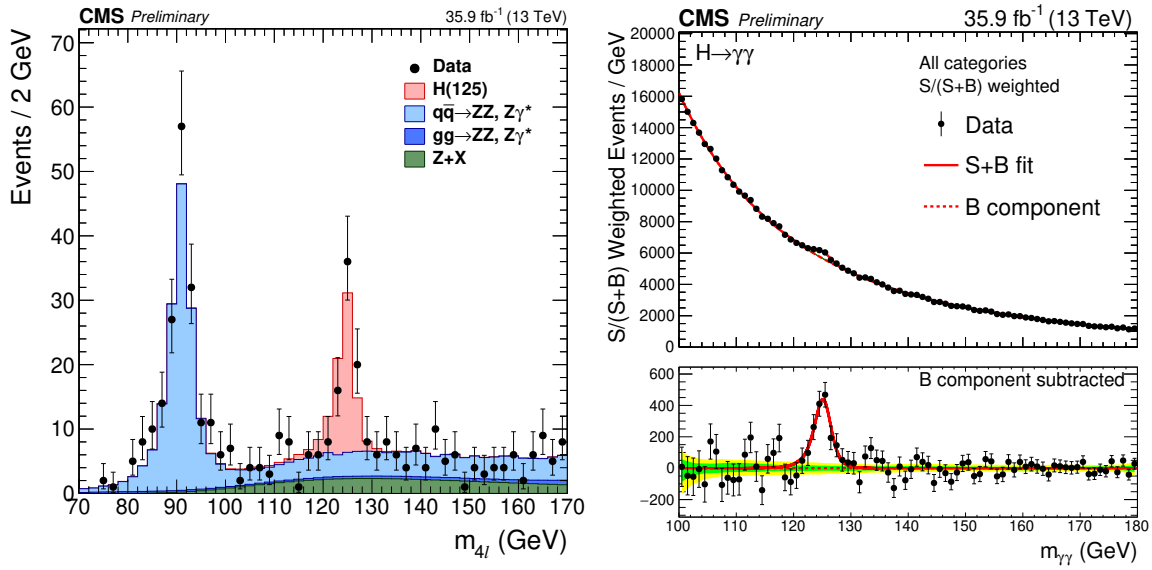


(a) (b)

Figure 1.5: a) The hierarchy of the production cross sections and their dependence on the Higgs boson mass at a proton-proton collider with a center-of-mass energy $\sqrt{s} = 13$ TeV. b) SM Higgs boson's branching ratios as a function of the Higgs boson mass. For both plots, the theoretical uncertainties are indicated as bands [60].

The hierarchy of the cross sections and their dependence on the Higgs boson mass at a proton-proton collider with a center-of-mass energy $\sqrt{s} = 13$ TeV are shown on the left in Figure 1.5.

The possible decay channels of the Higgs boson are shown on the right in Figure 1.5. The Higgs boson can decay producing a pair of fermions, leptons or pairs of W and Z bosons. A W or fermion loop cause the Higgs boson to decay into a pair of gluons or photons.



(a) (b)

Figure 1.6: a) Distribution of the four-lepton reconstructed invariant mass m_{4l}. Points with error bars represent the data and stacked histograms represent expected distributions. The SM Higgs boson signal with $m_H = 125$ GeV, denoted as $H(125)$, and the ZZ backgrounds are normalized to the SM expectation, the $Z + X$ background to the estimation from data [61].
b) Diphoton mass spectrum weighted by the ratio $S/(S + B)$ in each event class, together with the background subtracted weighted mass spectrum [62].

The Higgs boson was recently discovered at the Large Hadron Collider at CERN by the ATLAS and CMS experiments [4] [5], with a mass of [61]:

$$m_H = 125.26 \pm 0.20(stat.) \pm 0.08(syst.)\text{GeV} \tag{1.42}$$

where the width is constrained to be $\Gamma_H < 1.10$ GeV at 95% confidence level. The $\gamma\gamma$ and $4l$ decay channels provide a clear signature in the reconstructed mass spectra (Figure 1.6).
The study of the angular distributions of the $H \to ZZ \to 4l$ decay channel ($l = e, \mu$) made possible the determination of the Higgs boson spin-parity. This is consistent with a particle with spin 0 and even parity [63] [64].

Physics Beyond the Standard Model

The hypothesis that the Standard Model is valid also for energies several orders of magnitudes higher than the TeV scale leads to the naturalness problem [65], i.e. parameters should be fine-tuned making the theory unnatural.
The Standard Model fails also at describing some characteristics of our Universe. Not all the four fundamental forces are explained by the Standard Model. A Quantum Field Theory for gravity is still missing. Moreover, the SM postulates massless neutrinos, but their oscillation requires them to have a mass. Furthermore, astronomical observations, including rotation of galaxies and gravitational lensing effect [66], have shown that only around 5% of the total energy density in the Universe is in the form of baryonic matter, while 26.8% is made of particles that do not interact via electromagnetic nor strong interactions, the *dark matter* [67]. The Standard Model does not provide a stable *Weakly Interacting Massive Particle* to explain dark matter.
By postulating the existence of new particles, theories *Beyond the Standard Model* (BSM) can overcome some of the weak points of the Standard Model. BSM theories include:

- New quarks having a vector-like coupling to the W-boson, in contrast with SM quarks coupling, which is vector-minus-axial (V-A). This implies that their interaction via the weak force is identical for their left and right handed components [68].
- Unification of the fundamental forces through the additions of spatial extra dimensions [69].

The discovery of the Higgs boson raises further questions on the origin of elementary particle masses and on its role in the more fundamental theory underlying the Standard Model, which may involve additional particles to be discovered around the TeV scale.

Chapter 2

The Compact Muon Solenoid Experiment at the Large Hadron Collider

2.1 The Large Hadron Collider at CERN

The *European Organization for Nuclear Research* (CERN) has a very challenging program at the Energy Frontier. This represents the priority program of the world's most powerful accelerator, the *Large Hadron Collider* (LHC) [3] and its detectors to work on the following main experimental lines:

- Precision measurements of the observed Higgs boson.
- Verification of the compatibility of the observed Higgs boson with the Standard Model.
- Improve our knowledge about the electroweak symmetry breaking mechanism and provide evidence of physics beyond the Standard Model [70].
- Study of the the de-confined state of quarks and gluons called Quark-Gluon Plasma.

At the Intensity Frontier, precisions tests of the Standard Model are also being carried out.

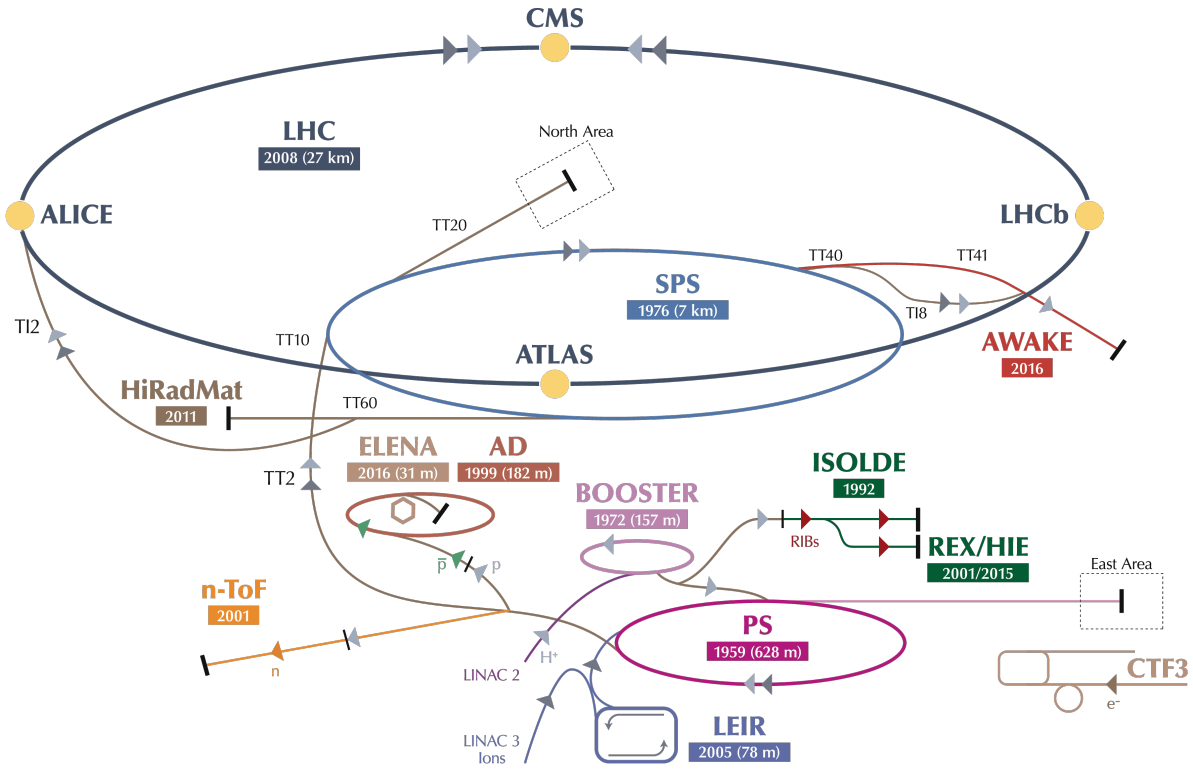


Figure 2.1: A schematic view of the CERN's accelerator complex [71].

In the following, the CERN accelerator complex and the CMS detector, one of the detectors pushing the Energy Frontier, are described.
The LHC is a superconducting circular accelerator hosted by the *European Organization for Nuclear Research* (CERN), in Geneva, Switzerland, in a 26.7 km long tunnel located at about 100 m below ground, which once accommodated the *Large Electron Positron Collider* (LEP), a e^+e^- accelerator.

2.1.1 CERN's accelerator complex

The LHC is the last element of an accelerator complex used to accelerate protons and ions to increasingly higher energies, as shown in Figure 2.1.
Protons stripped from hydrogen gas enter the LINAC 2, where they get accelerated to the energy of 50 MeV. The beam is then injected into the Proton Synchrotron Booster (PSB), which accelerates the protons to 1.4 GeV. The next in chain is the Proton Synchrotron

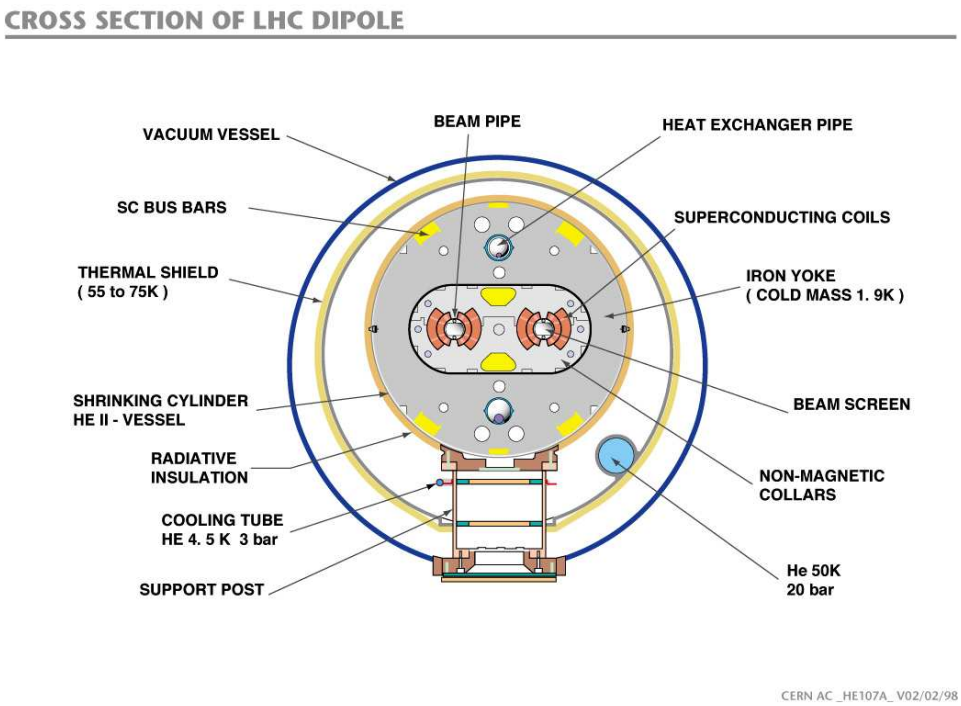


Figure 2.2: Cross section of LHC dipole [72].

(PS), where the beam particles reach the energy of 25 GeV. Prior to the injection to the LHC, the protons are accelerated in the Super Proton Synchrotron (SPS) to 450 GeV.

Protons are then injected into the LHC, where the two counter-rotating proton beams reach an energy of 6.5 TeV per beam.
The LHC is divided into eight octants. Each octant comprises an *arc* and a *straight sections*. In the arcs, dipolar, quadrupolar and corrective magnets steer and focus the beams. The average radius of curvature in the arc is about 2.6 km.
Straight sections contain detectors, radio-frequency cavities (RF), beam cleaning systems, injectors and the beam dump.
Overall, 1248 dipoles and 400 quadrupoles comprise the LHC. Dipoles are 15 meters long, instead quadrupoles are much shorter with 3.3 m in length. Dipoles are designed to operate up to a maximum flat top field of 8.33 T, which determines the maximum energy per beam of 7 TeV. Dipoles form a optic lattice and the weakest one determines the final energy of the beam. For this reason the bending strength of each dipole should be the same within 0.01%.
In order to reach the nominal electric current of 11850 A, the LHC employs copper

stabilized NbTi Rutherford cables operating in superfluid helium at 1.9 K [73] [74]. A cross section of a LHC dipole is shown in Figure 2.2.

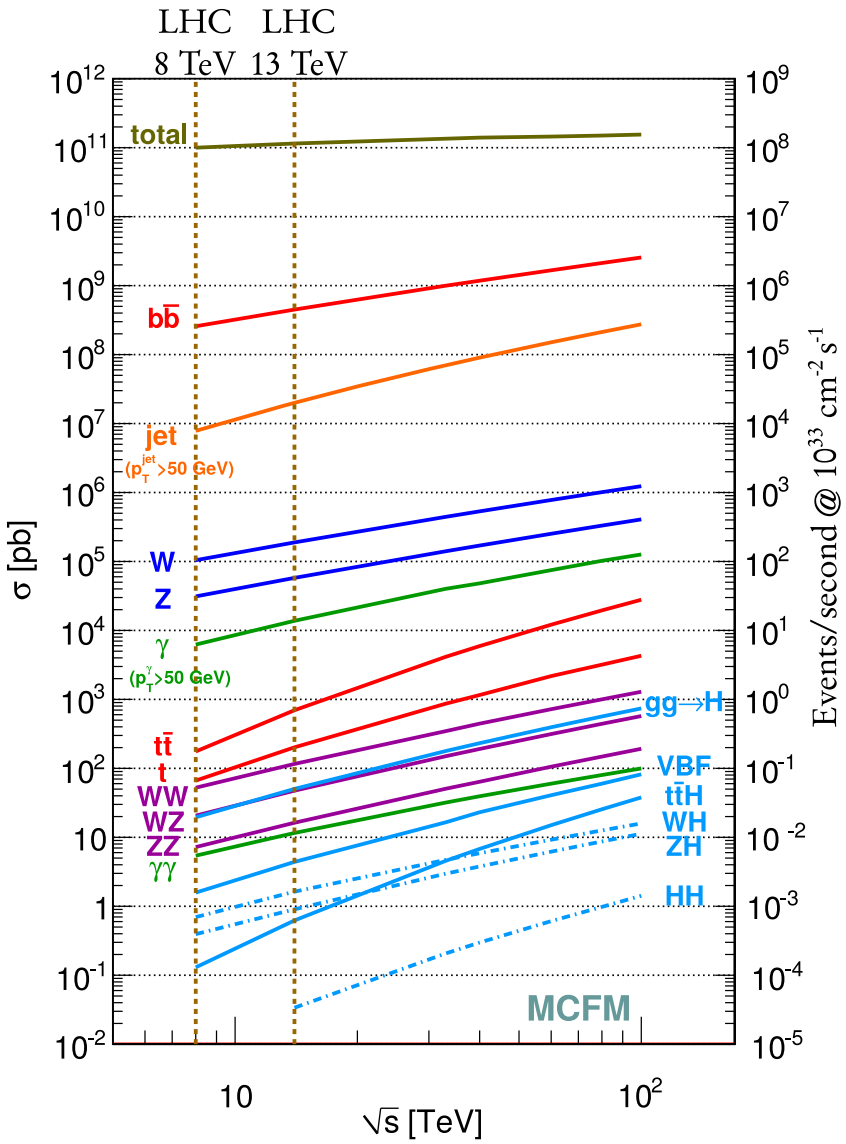


Figure 2.3: Different processes' cross sections and event rates at a hadron collider, as a function of the center-of-mass energy, for a value of the luminosity $L = 10^{33}$ cm^{-2} s^{-1}.

2.1.2 Luminosity and pile-up

In order to produce enough statistics of interesting events and considering the orders of magnitude separating the total $p-p$ cross section and the cross section for interesting events, the LHC has to provide a very high number of protons collisions every second. This is motivated by the fact that the rate of interesting physics processes that need be recorded compared to the inclusive pp cross section is many orders of magnitude smaller than the total event rate (Figure 2.3). For this reason, during nominal conditions, the LHC beam includes 2808 bunches, each made of $1.15 \cdot 10^{11}$ protons. The time interval between two consecutive bunches is 25 ns.
To quantify the rate of interesting physics processes produced by an accelerator and to make design decisions or forecasts, the introduction of the concept of *instantaneous luminosity* is needed. The instantaneous luminosity L is a measure for the efficiency of a particle accelerator. It depends on the number of protons per bunch N_p, the number k of circulating particle bunches, the revolving frequency f and the Gaussian transverse beam profiles in the horizontal and vertical directions σ_x and σ_y:

$$L = F\frac{N_p^2 f k}{4\pi\sigma_x\sigma_y}$$

F is a geometric luminosity reduction factor. It depends on the crossing angle θ_c, the RMS bunch length σ_z and the RMS bunch transverse size σ_{xy}:

$$F = \frac{1}{\sqrt{1+\left(\frac{\theta_c\sigma_z}{2\sigma_{xy}}\right)^2}}$$

The rate of a physics process $\frac{dN_i}{dt}$ is directly proportional to its cross section and the instantaneous luminosity.

$$\frac{dN_i}{dt} = L\sigma_i \tag{2.1}$$

The complexity of an event produced in a bunch crossing depends on the *pile-up* (μ), which is the average number of simultaneous p-p collisions in an event.
The instantaneous luminosity includes the information on the number of protons and the

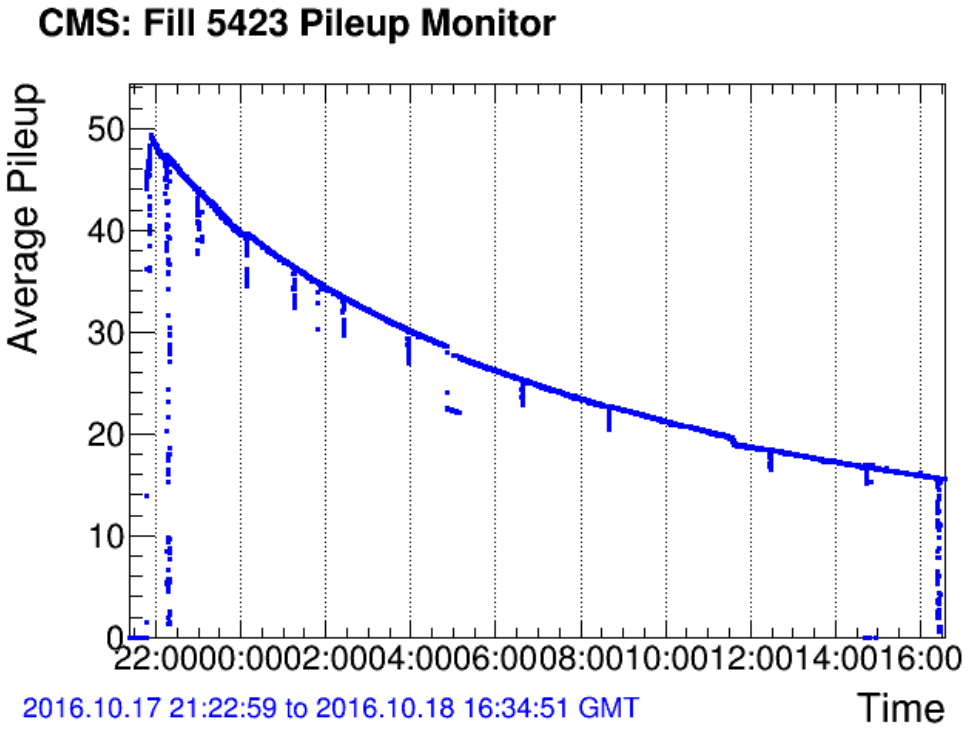


Figure 2.4: Average pile-up for a LHC fill in 2016. A peak average pile-up of 49.357 was reached, which corresponds to a value of the luminosity of $1.531 \cdot 10^{34}$ cm^{-2} s^{-1}[75].

size of the bunches. In fact, the pile-up is directly connected to it:

$$\mu = \frac{L\sigma_{\text{inel}}}{f_{BX}} \tag{2.2}$$

where σ_{inel} and f_{BX} refer to the inelastic pp cross section and the frequency of bunch crossings in the collision area respectively.
At $\sqrt{s} = 13$ TeV, the inelastic pp cross section as measured by CMS is [76]:

$$\sigma_{inel} = 71.3 \pm 0.5(\text{exp.}) \pm 2.1(\text{lum.}) \pm 2.7(\text{ext.})\text{mb}.$$

During a run the instantaneous luminosity is not constant, but decreases exponentially. This decrease is caused by collisions, beam-beam interactions and scattering with gas molecules in the beam-pipe. Therefore, the integrated luminosity in a run $\mathcal{L}$ depends on the peak instantaneous luminosity L_{peak}, the duration T of the run and the beam lifetime τ:

$$\mathcal{L} = L_{\text{peak}}\tau(1 - e^{-T/\tau}). \tag{2.3}$$

This behavior is visible in Figure 2.4 showing the measurements of the pile-up of the LHC beam in time.
During the run shown in Figure 2.4, the peak of instantaneous luminosity reached the value of $1.531 \cdot 10^{34}$ cm^{-2} s^{-1}, the maximum instantaneous luminosity ever recorded at the LHC.

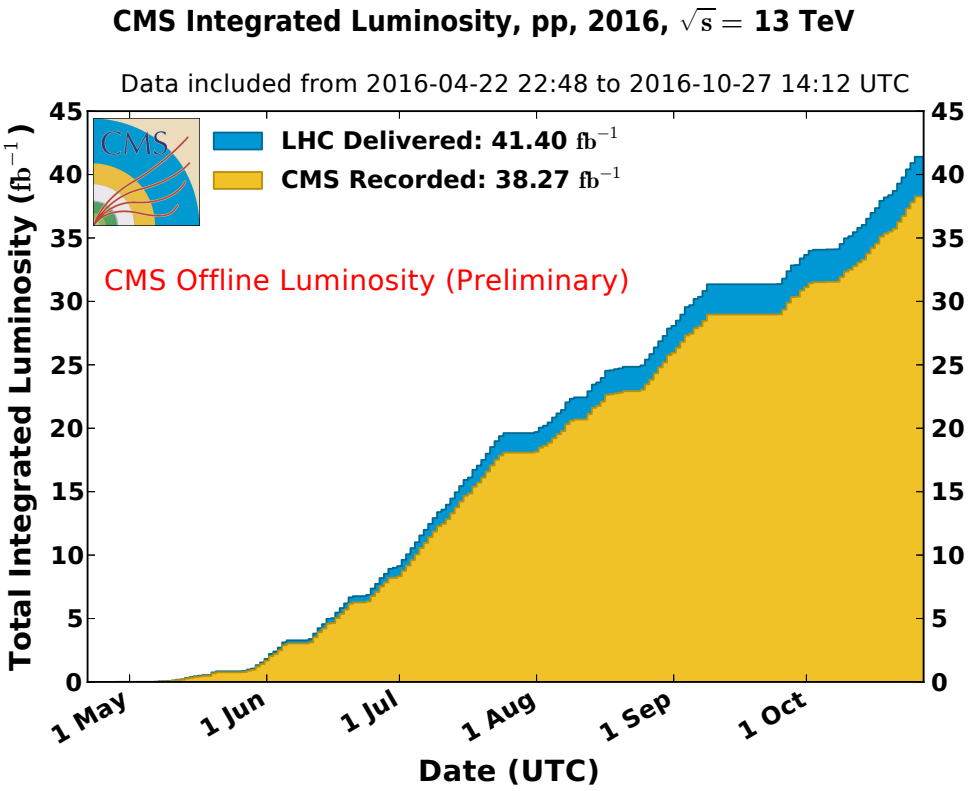


Figure 2.5: Integrated luminosity versus day delivered by LHC (blue) in 2016; the luminosity recorded by the CMS experiment is also reported (orange). The delivered luminosity accounts for the luminosity delivered from the start of stable beams until the LHC requests CMS to turn off the sensitive detectors to allow a beam dump or beam studies [77].

The integrated luminosity recorded by the CMS experiment during 2016 is compared to the luminosity delivered by the LHC in Figure 2.5.
Due to the demand of electric power needed to cool down the LHC superconductive magnets, the LHC cannot operate during winters, limiting the days of operation to about 200 days per year. Winters are usually used for upgrades and maintenance operations.

2.2 The Compact Muon Solenoid detector

Modern General-Purpose High-Energy Physics Detectors

General-purpose detectors at high-energy colliders have a cylindrical structure, built with different layers around the beam line.

Interactions take place along the beam line. Charged particles produced enter a *vertex tracker* detector first, leaving *hits* in the sensitive electronics.
Hits are used to reconstruct charged particles' trajectories and their origins producing *tracks* and *vertices*.

The measurement of the particles' momentum and electric charge is made possible by embedding the tracker in a magnetic field, which bends trajectories. The transverse component of the momentum, p_{T}, of a charged particle in a uniform magnetic field B is given by:

$$p_{\mathrm{T}} = \gamma m v = qB\rho, \tag{2.4}$$

where m is the mass of the particle, q its charge and ρ its bending radius. Under the hypothesis that a particle traverses the full radius of the magnet L, the sagitta can be written as:

$$s = \frac{L^2}{8r} = \frac{qBL^2}{8p_{\mathrm{T}}}. \tag{2.5}$$

The resolution on the transverse momentum is related to the size of the magnet and its strength by:

$$\frac{dp_{\mathrm{T}}}{p_{\mathrm{T}}} \propto \frac{p_{\mathrm{T}}}{BL^2}. \tag{2.6}$$

An *electromagnetic calorimeter* (ECAL) absorbs electrons and photons. Hadrons are absorbed in a subsequent *hadron calorimeter* (HCAL). Clusters of firing cells are reconstructed around showers. This allows the direction and the energy of the original particle to be determined.

Muons and neutrinos can escape the calorimeters. Muons interact with another tracking detector, usually called *muon chamber*. Neutrinos remain undetected.

The *Compact Muon Solenoid* Experiment (CMS) is one of the two multi-purpose experiments at the LHC [1] together with ATLAS [2]. It is built to measure the various

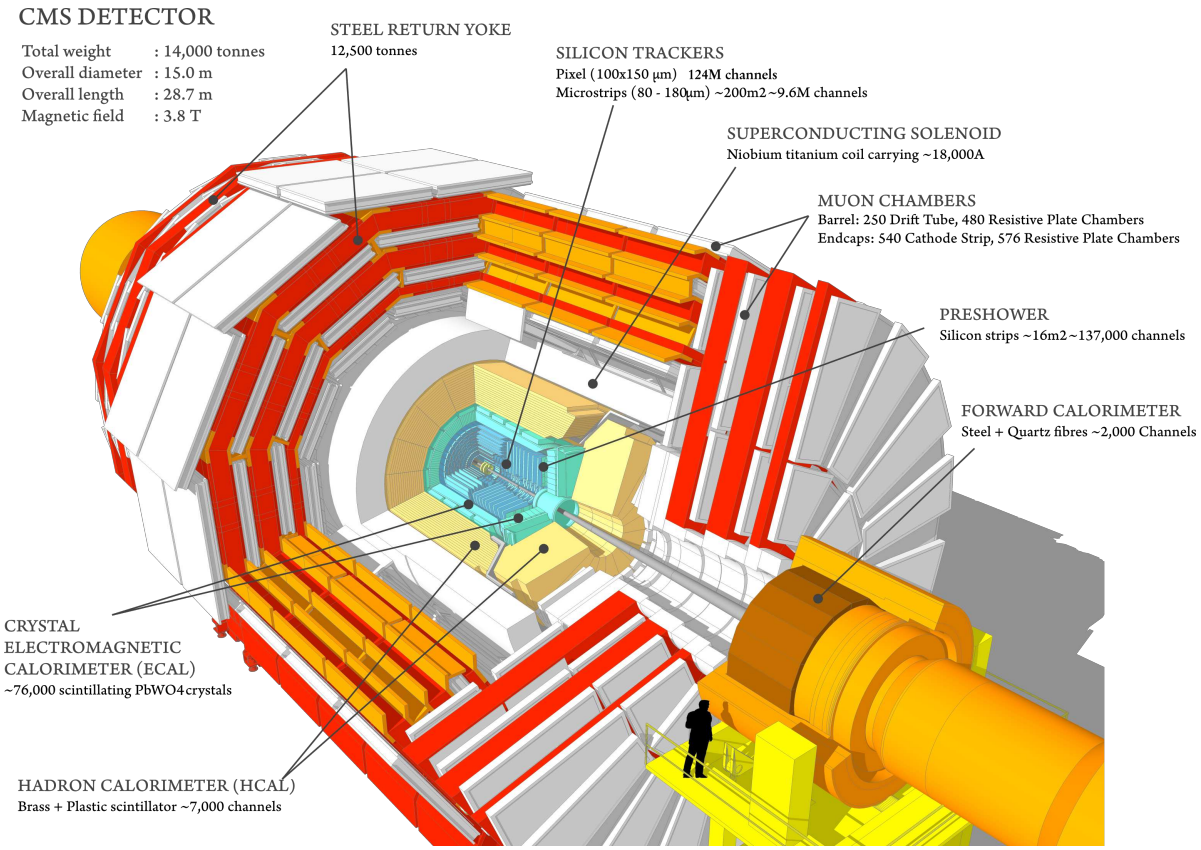


Figure 2.6: A schematic view of the CMS detector [78].

particles originating from proton-proton collisions provided by the LHC, covering as much solid angle around the interaction point as possible. Thus, its design is forward-backward symmetric with a cylindrical shape (Figure 2.6).

The CMS detector is located at Point 5 of the LHC, near the village of Cessy, in France. CMS is 21.6 m long, 15 m in diameter and weighs about 14000 t. The innermost part of CMS comprises two silicon tracking devices: the Pixel detector and the Strip detector. Their 76 million read-out channels provide a coverage in the $|\eta| < 2.5$ pseudo-rapidity interval.

The Electromagnetic Calorimeter (ECAL) is made of 76000 scintillating $PbWO_4$ crystals. Its coverage extends up to $|\eta| = 3$.

The Hadron Calorimeter (HCAL) is made of brass and plastic scintillators and provides the same pseudo-rapidity coverage as the ECAL. The HCAL is complemented at very high pseudo-rapidities ($3 < |\eta| < 5$) by the forward calorimeter (HF), made of steel and quartz fibers.

Table 2.1: Properties for the barrel of the CMS detector. The tracker thickness is given in number of radiation lengths. The transverse segmentation is expressed in radians and the energy in GeV [79].

Magnetic Field	3.8 T
Lever Arm	1.29 m
Bending Power	4.9 Tm
Pion reconstruction efficiency ($p_{\mathrm{T}} = 5$ GeV)	$90 - 95\%$
Tracker thickness at $\eta = 0$	$0.35 X_0$
ECAL Molière radius	2.2 cm
ECAL transverse segmentation $\Delta\phi \times \Delta\eta$	0.017×0.017
ECAL energy resolution	$\frac{\sigma(E)}{E} = \frac{2.8\%}{\sqrt{E(\mathrm{GeV})}} \oplus \frac{12\%}{E(\mathrm{GeV})} \oplus 0.3\%$
ECAL longitudinal segmentation	no
HCAL transverse segmentation	0.085×0.085
HCAL energy resolution	$\frac{\sigma(E)}{E} = \frac{110\%}{\sqrt{E(\mathrm{GeV})}} \oplus 9\%$

The Tracker, the ECAL and the HCAL are immersed in a 3.8 T solenoidal magnetic field, produced by a NbTi superconducting magnet operating at 4.5 K.
The Muon chambers and the 12500 t return steel return yoke, surround the superconducting magnet, providing pseudo-rapidity coverage up to $|\eta| = 2.4$.
The main properties of the CMS detector are described in Table 2.1.

CMS cylindrical coordinate system

The CMS coordinate system (Figure 2.7) is oriented such that the x-axis points south to the centre of the LHC ring, the y-axis points vertically upward, and the z-axis, also called beam axis, is in the direction of the beam to the west. The azimuthal angle ϕ is measured from the x-axis in the (x, y) plane, while the radial coordinate in this plane is denoted R. The polar angle θ is defined in the (R, z) plane.
The pseudo-rapidity is defined by:

$$\eta = -\ln[\tan\left(\frac{\theta}{2}\right)]. \tag{2.7}$$

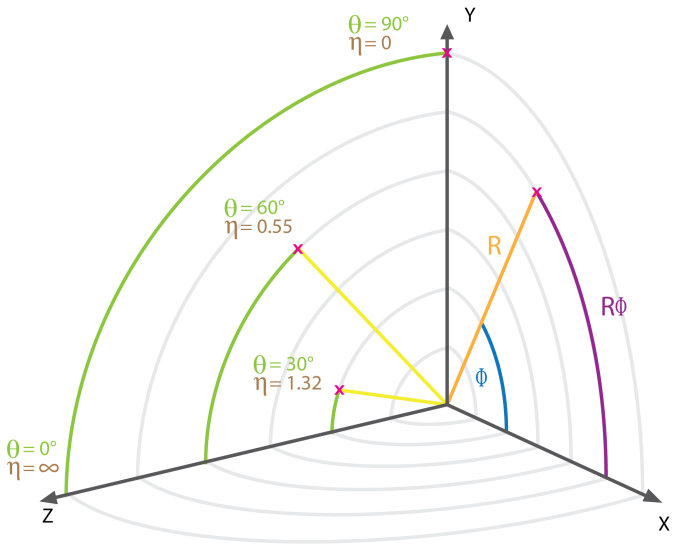


Figure 2.7: CMS coordinate system. The x-axis points south to the centre of the LHC ring, the y-axis points vertically upward, and the z axis, also called beam axis, is in the direction of the beam to the west.

The momentum component transverse to the beam axis, denoted p_{T}, is computed from the x and y components. The transverse energy is defined as:

$$E_{\mathrm{T}} = E \sin\theta \tag{2.8}$$

The superconducting solenoid magnet [80], capable of reaching a magnetic field of 3.8 T in its contained volume, is a central element in the design of CMS. During regular measurement mode, the magnet is cooled to 4.5 K in order to operate the NbTi material in its superconducting regime.

2.2.1 Silicon Vertex Tracker

The first detector component which is traversed by particles emerging from proton collisions is the *silicon vertex tracker* [82] [83]. Its mandate is to measure the trajectory of charged particles in order to measure their momentum and find primary or secondary vertices. Two different technologies have been employed to build the tracker.
The innermost part needs to be more granular, resistant to radiation and providing a

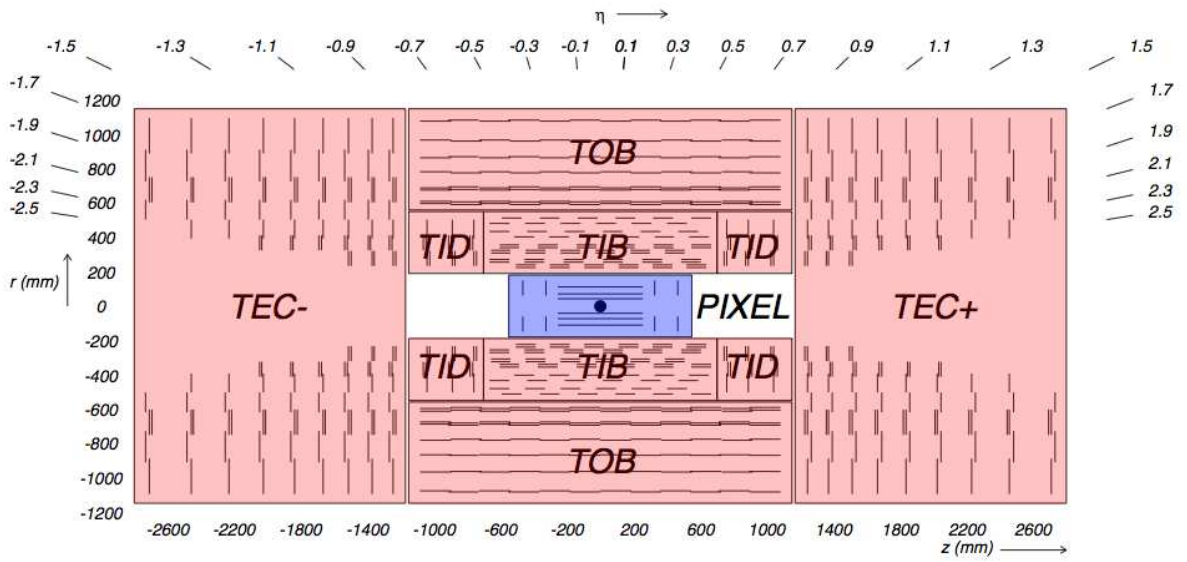


Figure 2.8: Arrangement of the different detectors in the silicon tracker. PIXEL (blue) refers to silicon Pixel detectors while Tracker Inner Barrel (TIB), Track Outer Barrel (TOB), Tracker Inner Disc (TID) and the Tracker End Cap (TEC), in red, all refer to silicon Strip detectors [81].

better position resolution. For these reasons the *pixel* technology has been used.
At a larger distance from the interaction point, the requirements on occupancy and radiation hardness are less stringent, hence making *silicon strips* the most cost-effective solution.
The geometric arrangement of these two sub-detectors is shown in Figure 2.8 and Figure 2.9.

The tracker was designed to operate with a 3.8 T solenoidal magnetic field at the luminosity of $1 \cdot 10^{34}$ cm^{-2} s^{-1}. At this luminosity, an average of 1000 particles is expected from more than 20 overlapping proton-proton interactions traversing the tracker every 25 ns. Hence, the Tracker has to be radiation-hard, fast and granular. These last two requests are addressed by a high density of sensors along with its cooling system. The *material budget* has to be as low as possible to reduce effects like multiple scattering, which introduce uncertainties in the reconstruction of the propagation of the particles [81] (Figure 2.10). Contributions to the material budget come from active materials (the silicon sensors) or passive materials, e.g. support structures, cables, cooling pipes or gas.

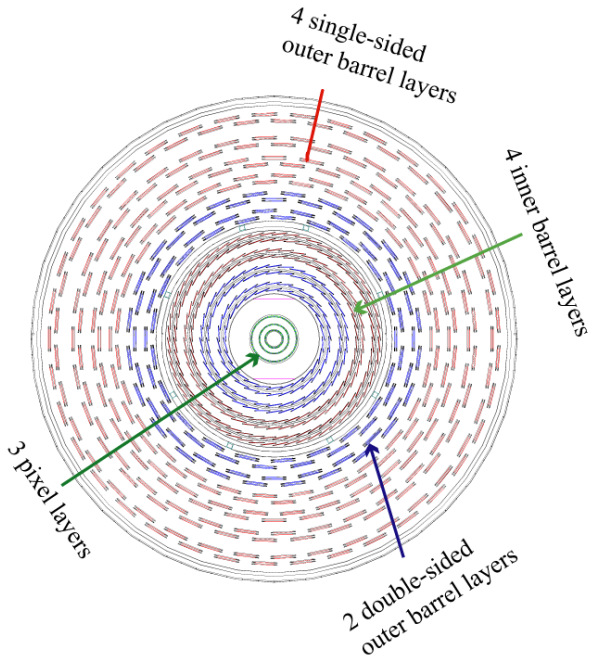


Figure 2.9: CMS Tracker layers shown in the plane perpendicular to the beam line [81].

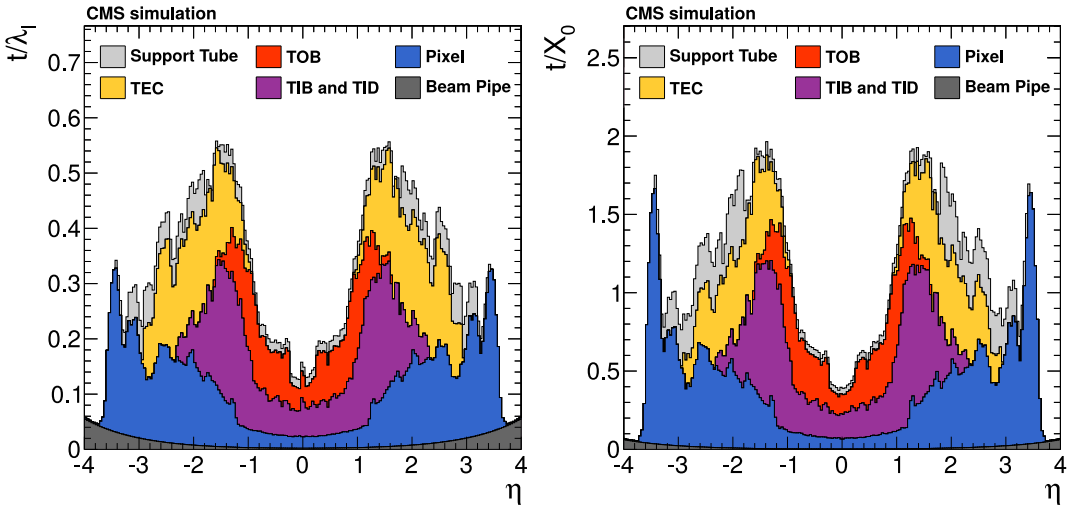


Figure 2.10: Material budget of the inner tracking system in number of interaction lengths (left) and radiation lengths (right) [81]. Contributions to the material budget come from active materials (the silicon sensors) or passive materials, e.g. support structures, cables, cooling pipes or gas.

Pixel Detector

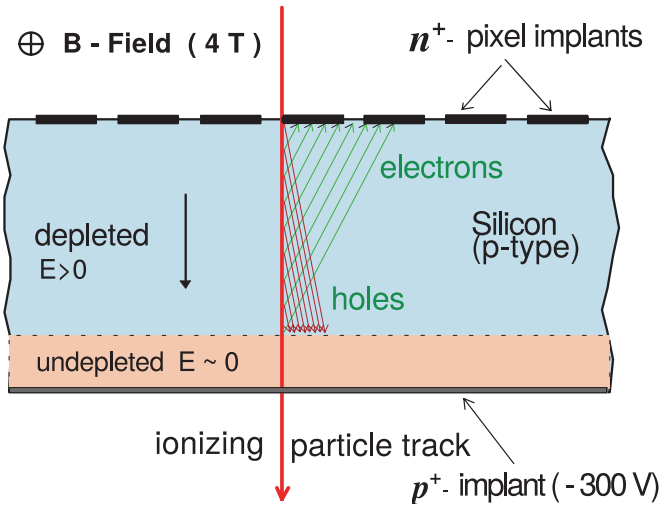


Figure 2.11: Schematic view of the doped Silicon layer used in a Pixel detector chip. As an example, a proton traversing a CMS Pixel module leads to the release of charge carriers. These carriers are collected at the n-implant.
Charge sharing is due to the Lorentz force and to the incidence angle of the hitting proton with respect to the Pixel module.

The innermost part of the Silicon Tracker, closest to the interaction point, is made of Pixel detectors. Each pixel element is about $100 \times 150 \mu m^2$ wide. Each sensor is 285 μm thick with a planar n^+-on-n configuration. High dose n-implants placed into a resistive n-substrate form the pixels, while the junction is formed with a p-implant on the back-side. The passage of an ionizing particle through the depleted region of the sensor leads to the release of charge carriers which are then collected at the n-implant (Figure 2.11). The Lorentz force deflects the carriers' trajectory with respect to the electric field lines, making adjacent pixels to share the carriers' charge. If the pulse produced in a pixel sensor exceeds a configurable threshold, the sensor passes the zero-suppression. In CMS, this threshold is set to the equivalent charge of 3200 electrons. By interpolating the charge deposit in a cluster of pixels, this charge sharing mechanism improves the spatial resolution, which reaches a value of $\sim 10 \mu$m. This space resolution, an SNR of ~ 70 and a very low occupancy, enable CMS to achieve a good p_T resolution up to TeV energies and a good b-tagging efficiency.

A custom *Readout Chip* (ROC) handles the 40 MHz analog read-out for a rectangular matrix made of 52×80 pixels (Figure 2.12).

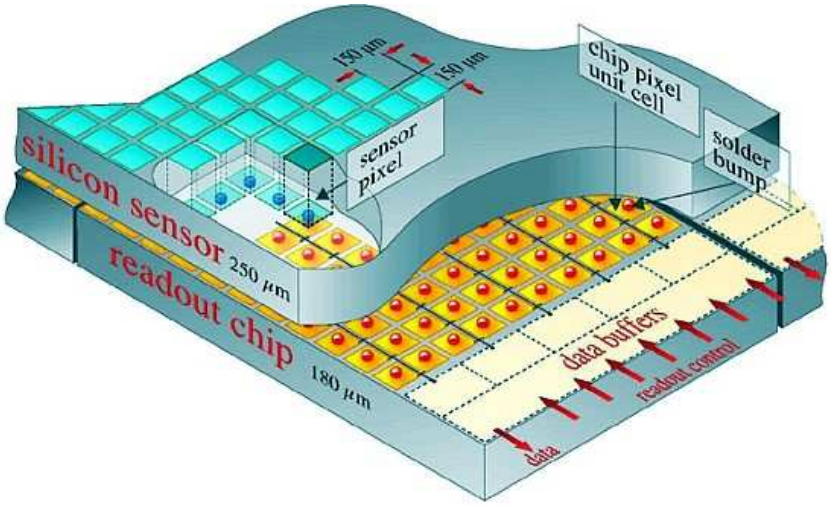


Figure 2.12: Simplified scheme of the Pixel detector module.

The Pixel detectors are arranged in three barrel layers and two end-cap disks. The three barrel layers are placed at a distance of 4.4 cm, 7.3 cm and 10.2 cm from the beam. The end-cap layers are placed in correspondence of both the ends of the barrel layers. With 6 cm and 15 cm of inner and outer radius, they are located at $z = \pm 34.5$ cm and $z = \pm 46.5$ cm.
The position of the barrel and end-cap regions allows a three-hits coverage for tracks in a pseudorapidity range up to $|\eta| = 2.5$. For this reason, Pixels play a crucial role in track seeding. They are also used for fast tracking and vertexing at trigger level.
The short distance from the beam-spot makes the Pixel detectors absorb a high radiation dose whose effects are leading to a lower charge collection or displacement damage [84]. For these reasons, the Pixel detector has been replaced by a new one at the beginning of 2017.

Strip Detector

At a radius of ~ 20 cm from the beam line, the flux of particles is much lower than in the Pixel detector and the occupancy requirement is also less rigid. Starting from $r = 20$ cm to $r = 130$ cm, with ten layers in the barrel and twelve in the end-caps, the silicon Strip Tracker surrounds the Pixel detector. Its working principle is very similar to the Pixel detector's one.

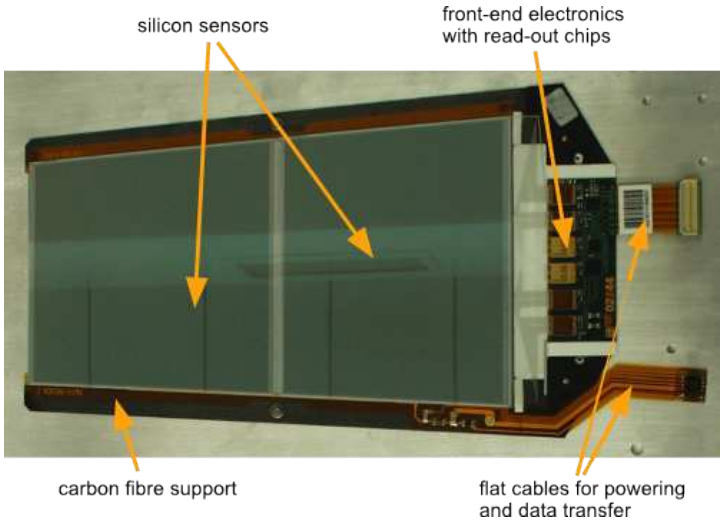


Figure 2.13: Two sensors are mounted side-by-side and connected via bond wires. The effective strip length is about 20cm. The signals are read-out at one end by front-end electronics with dedicated read-out custom electronics. The sensors and the electronics are supported by carbon fiber structures.

A detector module is the basic functional sub-unit of the silicon tracking system. Each module consists of a set of single sided sensors, a mechanical support structure and the readout hybrid, as shown in Figure 2.13.
The sensors are arranged in a total number of about 20000 modules.
The read-out system [85] comprises the analogue APV25 read-out chip , analogue optical links and an *Front-End Driver* (FED) processing board. A single read-out chip receives signals every 25 ns from 128 different channels of the same strip sensor. These signals are amplified and buffered. When an external trigger is received, the height of the pulse are transmitted to the FED. The FED digitizes these signals and a network packet is created that is sent to the High-Level Trigger farm, described in Section 2.2.5. The FED executes the zero-suppression which removes:

- Strips with a charge that does not exceed the expected noise by at least a factor 5.
- Pairs of adjacent strips, whose sum of charges does not exceed the expected noise by at least a factor 2.

As illustrated in Figure 2.8, the Strip Tracker can be divided into four areas, depending on the distance from the beam-spot and pseudorapidity: *Tracker Inner Barrel* (TIB), *Track Outer Barrel* (TOB), *Tracker Inner Disc* (TID) and the *Tracker End Cap* (TEC).

Table 2.2: Mechanical dimensions and numbers of the CMS silicon Strip detectors. The stated radii from the beam are average values for barrel modules and measured in the center for the disks.

Layer	Radius [mm]	Sides	Modules	Pitch [μm]	Strips
TIB1	250	2	336	80	768
TIB2	340	2	456	80	768
TIB3	430	1	552	120	512
TIB4	520	1	648	120	512
TOB5	610	2	504	122-183	768-512
TOB6	696	2	576	122-183	768-512
TOB7	782	1	648	183	512
TOB8	868	1	720	183	512
TOB9	965	1	792	122	768
TOB10	1080	1	888	122	768
TID1	277	2	144	81..112	768
TID2	367	2	144	113..143	768
TID3	447	1	240	124..158	512
TEC1	277	2	144	81..112	768
TEC2	367	2	288	113..143	768
TEC3	447	1	640	124..158	512
TEC4	562	1	1008	113..139	512
TEC5	677	2	720	126..156	768
TEC6	891	1	1008	163..205	512
TEC7	991	1	1440	140..172	512

2.2.2 Electromagnetic Calorimeter

The *Electromagnetic Calorimeter* (ECAL) [86] is the detector that follows the tracking system. It is made of lead tungstate ($PbWO_4$) crystals, 61200 in the barrel region, 7324 in the two end-cap regions (Figure 2.14). In the endcaps, a preshower detector is located before the calorimeter to initiate electromagnetic cascades. Preshower detectors are made of two lead radiators, of about two and one radiation lengths respectively. Each lead radiator is followed by a silicon strip layer.
Lead tungstate is characterized by a fast response as $\sim 80\%$ of light is emitted in 25 ns, the time between two bunch crossings at the LHC.

The calorimeter must provide a very good measurement of energy from the low energy region, $\sim$ 1GeV, up to the TeV scale. An estimate of the absolute calibration of the ECAL response to electrons and photons was carried out with data collected at $\sqrt{s} = 7$ and 8 TeV [87]. Its energy resolution can be parametrized as follows:

$$\frac{\sigma(E)}{E} = \frac{2.8\%}{\sqrt{E(\text{GeV})}} \oplus \frac{12\%}{E(\text{GeV})} \oplus 0.3\% \tag{2.9}$$

The first term is due to stochastic sampling and amplification fluctuations, the second one to electronic noise and pile-up while the last one to inhomogeneities of the material in front of the calorimeter and the non-perfect inter-calibration of the cells.
Due to the different radiation profiles and magnetic field setups the barrel and the end-caps use two different readout systems. The light in the barrel region is read by avalanche silicon photo-diode (APD), while in the end-caps by vacuum photo-triodes (VPT).

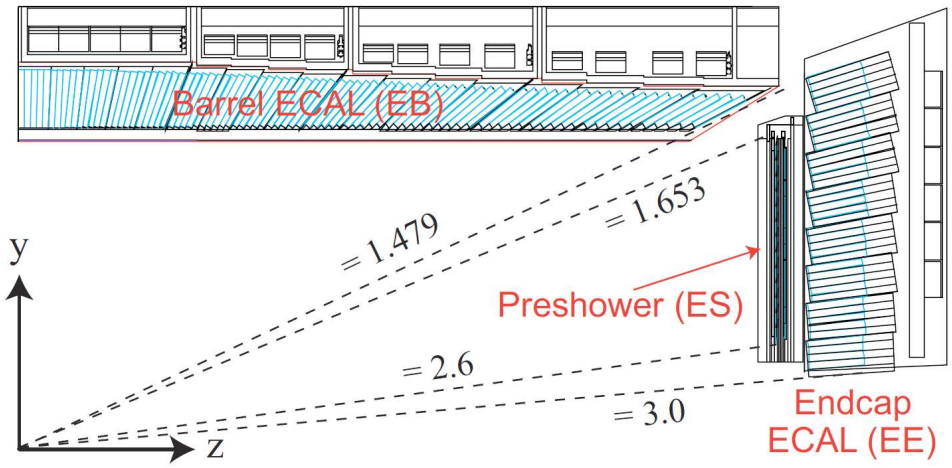


Figure 2.14: Geometric view of one-quarter of the ECAL.

2.2.3 Hadronic Calorimeter

The *Hadronic Calorimeter* (HCAL) [88] is constituted by four elements, as described in Figure 2.15.
The *barrel HCAL* (HB) covers the region $|\eta| \leq 1.3$ and the *end-caps HCAL* (HE) covers the region $1.3 < |\eta| \leq 3$. They are both sampling calorimeters made of scintillating tiles and metal absorber plates.
The HB and HE are fully located within the 3.8 T magnetic field generated by the solenoid magnet for which the absorber material is brass, except for the first and last two plates for which stainless steel was used.
The *outer HCAL* (HO) is located outside of the magnetic solenoid in the central region of the CMS detector and covers the region of $|\eta| < 1.3$. Its purpose is to measure the energy of those particles that have not been stopped by the electromagnetic EB and the HB.
The *forward HCAL* (HF), a steel and quartz fiber Cherenkov calorimeter placed outside the magnet return yokes at 11 m from the interaction point, extends the calorimeter coverage to $|\eta| = 5.3$.

The HCAL energy resolution was measured in a pion test beam [89] to be:

$$\frac{\sigma(E)}{E} = \frac{110\%}{\sqrt{E(\text{GeV})}} \oplus 9\% \tag{2.10}$$

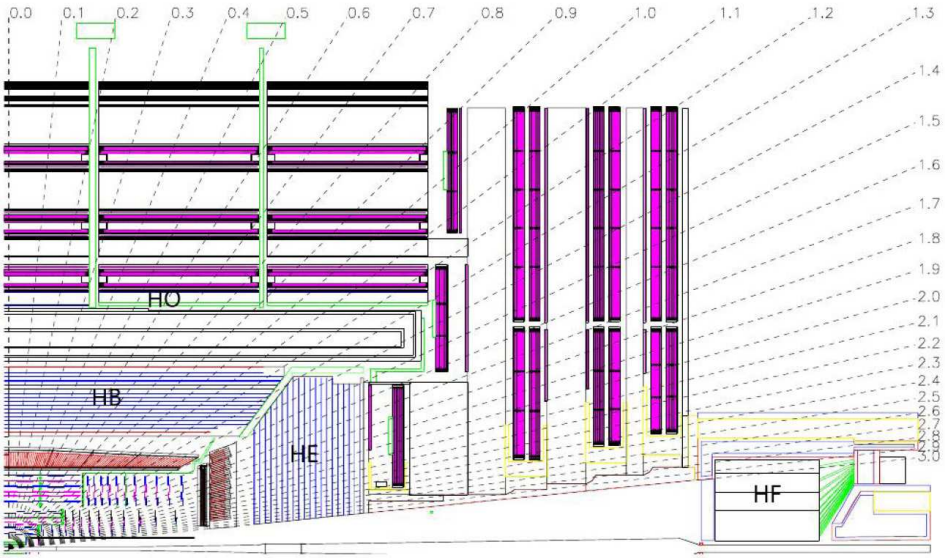


Figure 2.15: Geometric view of one-quarter of the CMS detector. The HCAL detector components HB, HE, HO, HF are indicated.

2.2.4 Muon System

The *Muon System* [90] [91] is made of different detectors located around and between the layers of the magnetic return yoke as shown in Figure 2.16.

Drift tubes (DT) cover the $|\eta| < 1.2$ barrel region in the innermost five slices of the return yoke.
Tubes have been chosen over conventional drift chambers to keep each wire protected from the noise produced in other cells and as a safety measure against possible wire breaking.

Due to the noise from neutron punch-through, DTs cannot be employed in the end-caps where *Cathode Strip Chambers* (CSC) are used: between $0.9 < |\eta| < 2.4$. In CSCs, anode wires are stretched between two cathodes: one is segmented in strips perpendicular to the wires, instead the other is not (Figure 2.17).

The CSCs can reach a hit efficiency of 99% and determine the impact point of a muon with a precision of 2 mm, which can become 75 µm in the first layer and 150 µm in the others, after applying offline corrections.

The *Resistive Plate Chambers* (RPC) are gaseous parallel-plate detectors, which are used

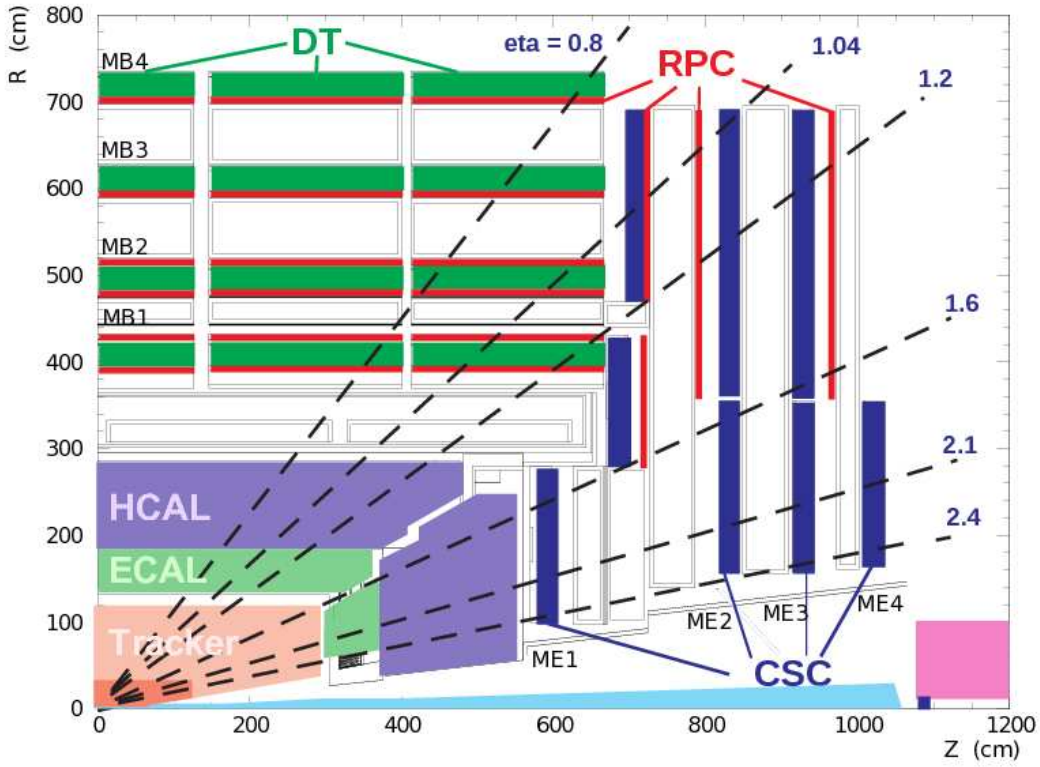


Figure 2.16: Geometric view of one-quarter of the muon system, showing drift tubes, cathodic strip chambers (CSC) and resistive plate chambers (RPC).

as a very fast trigger system (Figure 2.18). RPCs are capacitors with parallel electrodes, made of bakelite, a material with high resistivity, coated with conductive graphite paint. When an ionizing particle traverses the region between the plates, a short discharge happens, rapidly stopped when the free charge on the conductive surface is exhausted. The readout is made using the induced charge in strips placed outside the capacitor and isolated from the electrodes. RPCs are operated in avalanche mode, where charge multiplication is not very high, to allow for high rates but requiring a strong amplification. A very fast but approximate measurement of the muon momentum is given by a logic processor which matches hits patterns which are associated with different momentum intervals.

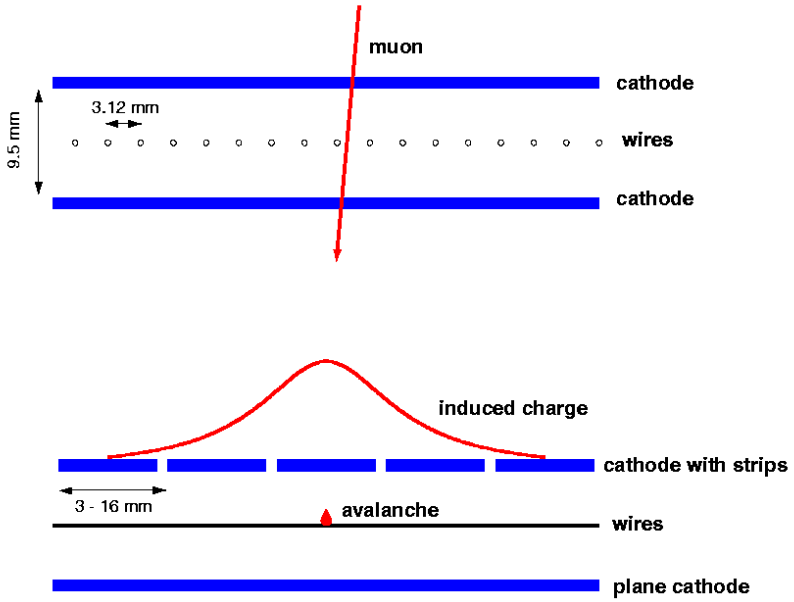


Figure 2.17: Working principle of a CMS Cathodic Strip Chamber.

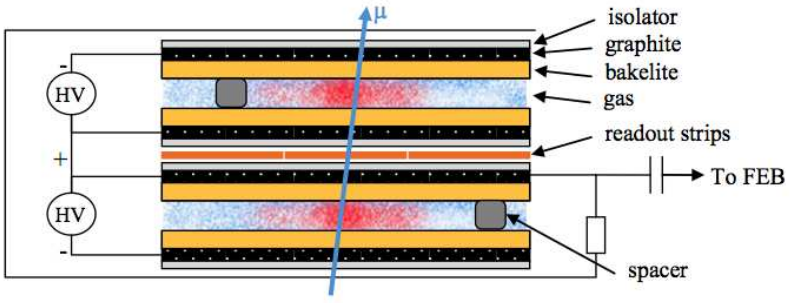


Figure 2.18: Scheme of a CMS Resistive Plate Chambers.

2.2.5 Trigger and Data Acquisition System

The *Trigger and Data Acquisition* [92] system of the CMS experiment provides essential preliminary online analysis of the raw data for the purpose of selecting potentially interesting events and transfer them the data storage system. Real-time handling of raw data at the design input rate of 40 MHz is required.
To keep the overall data rate within the current capability of the storage system of few kHz, the trigger system must achieve a rejection ratio for background events of at least 10,000:1. The Trigger system is structured in two levels, as described in the following sections.

Level 1 Trigger

The standard first level of trigger is completely based on programmable electronics, typically FPGAs (Field Programmable Gate Arrays) and ASICs (Application Specific Integrated Circuits), and in charge of reducing the 40MHz input rate to at most 100kHz [93]. It is completely *synchronous*, and trigger decisions must be taken within a few microseconds. This time depends on the size of the buffers and on the trigger rate. Since the response time is critical, the Level 1 exploits only information coming from the muon system and calorimeters. This information is extracted in the form of so-called trigger primitives which indicate the presence and the number of objects like electrons, photons, muons, jets and E_{T}^{miss}.

High-Level Trigger

The events accepted by the L1 trigger are read-out by the front-end electronics, assembled by the DAQ system and reconstructed, analyzed and filtered by the *High-Level Trigger* (HLT).
The HLT represents the second level of the CMS trigger system [9]. In 2016, the HLT runs on a cluster of commercial computers, made of $\sim$ 22000 Intel Xeon cores. This cluster runs a streamlined version of *CMSSW*, the C++ offline reconstruction software, processing many events in parallel [94]. Hence, the physics reconstruction at the HLT

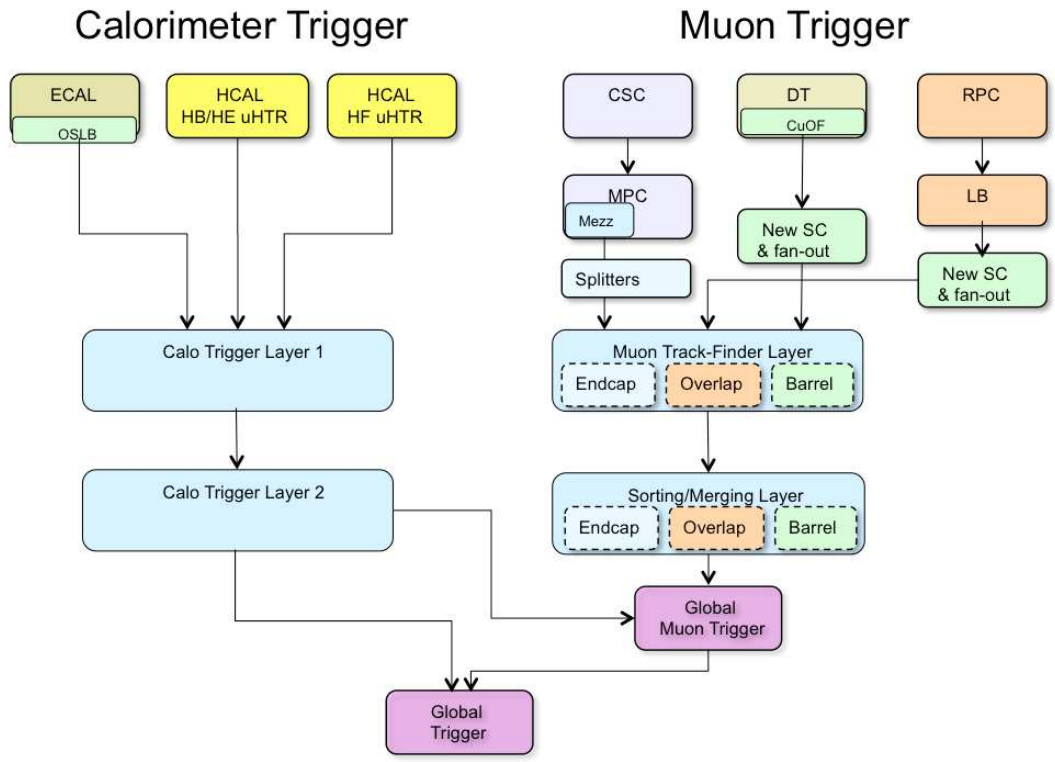


Figure 2.19: Data flow of the CMS Level 1 trigger [93].

should be as close as possible to the offline reconstruction: similar algorithms and calibration are executed but optimized for lower latency and the selection criteria are looser than those in the final analyses.

An HLT menu is a set of possible paths (Figure 2.20). Each path is an ordered progression of filter or producer modules, which are arranged in blocks of increasing complexity. In this way, faster modules are executed first and their products are filtered. If the conditions of the filter are not satisfied, the filter fails and the path is aborted.

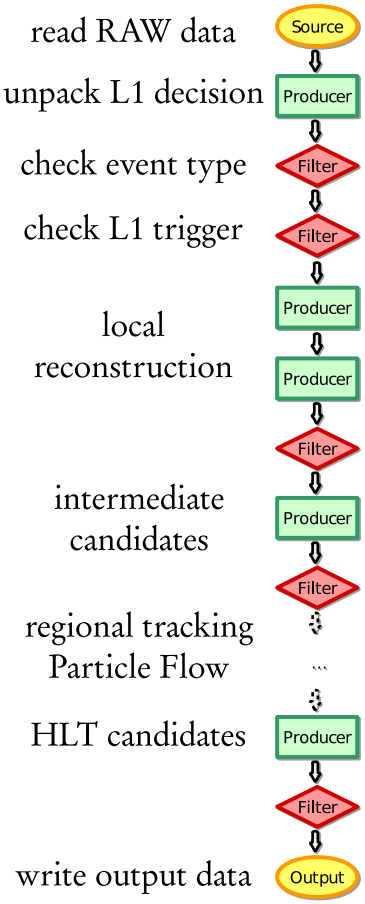


Figure 2.20: A scheme of an HLT trigger path.

The design of an HLT menu has two main constraints:

- The time budget of the HLT farm, which is given by the average time between two events at the same processing unit.
  In 2016, 22000 cores and an input rate of 100 kHz gives a maximum average time budget of 220 ms per event.

- The output rate to storage, which is limited to 1 kHz.

These constraints are a challenge for the development and operation of the HLT.
The expected increase of event complexity starting will require a fundamental redesign of the online reconstruction algorithms and software architecture, which is discussed in Chapters 3 and 5.

2.3 Offline and Computing

In order to make resources and data available in many countries and institutes, CMS employs a distributed computing model with services and resources communicating in what is called the *Worldwide LHC Computing Grid* (WLCG) [95]. Independently of the location of a user, the WLCG implements services to store and analyze data produced by the experiments. These services come in form of a middleware layer, on top of which experiments can build different software layers and interface with the WLCG infrastructure.
Every WLCG site is organized in the following logical components:

- The *User Interface* is made of nodes where the interaction between the user and the WLCG site happens.
- Data catalogs, workload management systems are some instances of the *Central Services*.
- Jobs submitted by the user are managed by the *Computing Element* and eventually executed by *Worker Nodes*.
- The *Storage Element* is used for managing the access to data. Usually, tapes are used for long-term storage, while hard-drives and solid-state drives are used for quick and short-term access to data.

The architecture of the WLCG is hierarchical with different responsibilities for different sites [96]. The WLCG comprises:

- 1 logical Tier-0, distributed on two physical computing centers: the CERN Data Center and the Wigner Institute computing center in Budapest, Hungary. The responsibilities of the Tier-0 in the CMS computing model include the prompt reconstruction of RAW data collected by the detector and the low-latency calibration and validation work-flows needed to ensure high-quality data collection. RAW data are archived on tape and distributed to the Tier-1s as well.
- 15 Tier-1 sites, are usually big national data centers and communication nodes between Tier-2s. The main focuses of the Tier-1s' resources utilization by CMS are to archive and serve RAW and reconstructed data, while they process and skim the data samples stored, and perform reconstruction and digitization on simulated

samples. If resources are underutilized, Tier-1s can be also for user analysis and for simulation.

- 168 Tier-2s are used primarily for user analyses and simulation activities. Although, when available, user analyses can be executed at Tier-1s, Tier-2 sites still continue to serve as the main analysis resource. In fact, in 2016 only 7% of the analysis jobs were executed on a Tier-1 site.

- CMS computing resources can elastically expand for short periods of times by exploiting opportunistic resources either via Cloud (e.g. Amazon Web Services [97]), or by exploiting supercomputers draining time. The CMS HLT farm has also been commissioned as an opportunistic resource for offline processing. During LHC shutdowns, its 22000 cores become available for offline production activities. In 2016, the CMS HLT farm offered to CMS a sizable contribution corresponding to 14% of the 2016 CPU usage at all the Tier-1s.

Overall, CMS grid infrastructure comprises 150000 cores on the Working Nodes, 75 PB of disk storage and 100 PB of used tape storage. The average volume of data which is transferred every week amounts roughly to 6 PB [96].
As a comparison, the most powerful supercomputer in Germany today is *Hazel Hen*, at the Höchstleistungsrechenzentrum in Stuttgart. About 102 different projects have been executing on Hazel Hen's 185,088 cores [98]. On the storage side, the daily incoming data traffic of world's largest social network today, Facebook, is roughly 600 TB [99].

2.3.1 Simulation

Simulation of specific physical processes is important to compare the predictions of theoretical models with the experimental results.
The first step consists in simulating the physics using event generators like PYTHIA [100], Herwig [101], MadGraph [102], Sherpa [103], Alpgen [104].
Then, the output of the event generators is then used by the software framework GEANT4 [105] to simulate the interaction of the entire detector with particles and their decays. Simulated energy deposits are then used to produce a response in the simulated detector. This response will be the input of the standard reconstruction work-flow. Reconstruction of simulated data can be executed at any level of the WLCG.

Event generation and simulation (GENSIM) is executed at Tier-1s, Tier-2s. The HLT farm can be used to produce GENSIM datasets during shut-down periods.

2.3.2 Particle Flow Event Reconstruction

The reconstruction of an event can be carried out combining two approaches together:

- A horizontal mode employs all the information coming from one detector to reconstruct *physics objects* specific to that detector.

- A vertical mode exploits the information coming from a *topological link* between physics objects of different detectors to reconstruct each particle in an event. The vertical approach has other advantages such as the cross-calibration and validation of different detectors or the identification of some kind of detector backgrounds.

This method is the basis of the *Particle Flow* reconstruction algorithm [106] [79].

Track Reconstruction

Precise reconstruction in the Silicon Tracker plays a crucial role in Particle Flow. This is motivated by the fact that the granularity of sub-detectors limits the probability to connect together the correct objects coming from different sub-detectors. The production of high quality Particle Candidate is tightly connected to the purity of the Physics Objects produced by the Silicon Tracker. These Particle Candidates are the input of downstream algorithms like *jet reconstruction*. As an example, the τ lepton can be reconstructed by studying three possible decay channels: one charged hadron, one charged hadron and a π^0, and three charged hadrons. The correct identification of τ leptons, with the minimization of the mis-identification rate, relies heavily on the precision of the information coming from the track reconstruction. More details about the CMS track reconstruction can be found in Chapter 3.

Cluster Reconstruction in Calorimeters

Cluster reconstruction consists in the collection of energy deposits in calorimeters and their consequent aggregation and association to the particles that produced them. Clustering is of paramount importance for several reasons:

- It allows to measure the direction of stable neutral particles such as neutrons and photons, which would remain undetermined.
- The deposited energy can be measured by collecting the energy of nearby channels.
- Bremsstrahlung photons can be associated to the electron which produced them.

Clustering is performed in three steps [106]:

1. *Seeding*: local energy maxima are found. A minimum energy threshold may be set as a requirement for a cell to become a seed.
2. *Topological clustering*: a *PF Cluster* is grown around each seed. The conditions for adding a cell to the PF Cluster are based on its adjacency list, as a cell has to share at least one side with a cell already belonging to the cluster, and its energy should be at least two standard deviations of the noise.
3. *Energy sharing*: It can happen that a cell belongs to two or more PF Clusters. In order not to account for its energy more than once, its energy is divided among the sharing clusters. Energy fractions are calculated for each PF Cluster based on the clusters' energy and the distance between the cell and the clusters' centers of gravity.

The Silicon Tracker material (Figure 2.10), located upstream with respect to the calorimeters, and the magnetic field are taken into account in the *Superclustering* step. Superclustering allows to associate radiation produced by bremsstrahlung photons and conversion tracks to the particle that produced it.

Particle Flow Linking

Signals in different sub-detectors are linked using geometrical criteria. For instance, the extrapolation of the trajectory of a particle, reconstructed in the Silicon Tracker, onto

the calorimeter creates a window in η and ϕ. This search window accounts for multiple scattering and other uncertainties, like gaps in a sub-detector. If a cell belonging to a cluster lies inside the window, a link between the cluster and the track is created. In order to avoid exploring all the possible linkings between elements of different sub-detectors, a class of data structures called k-dimensional trees are used, which are described in more details in Section 5.2.
Linked objects form *PF Blocks*. Blocks are used to reconstruct *Particle Candidates* based on the combined information coming from the linked Physics Objects.
For instance, the Muon system, described in Section 2.2.4, allows the identification of muons with high purity thanks to the absorption of the calorimeters. *Standalone tracks* are created from hits in the Muon system. Standalone tracks are then matched to a track in the Inner Tracker by comparing the track parameters propagated onto a common surface.
Like for muons, also electrons receive a special treatment. Their reconstruction is seeded by a cluster in the ECAL detector [107]. The position of the cluster defines the solid angle where to match hits in the Silicon Tracker. However, due to the material budget, electrons lose a considerable fraction of their energy producing bremsstrahlung photons. A precise measurement of the energy of electrons require the collection of all the hits. However, the bremsstrahlung energy loss distribution of electrons propagating in matter is not Gaussian. Applying the same fitting techniques used for muons to electrons would lead to a small number of hits and high χ^2. For this reason a *Gaussian-sum filter* techniques is employed, which models the bremsstrahlung energy loss with a mixture of Gaussian distributions [108].

Jets Reconstruction

Jets are reconstructed by clustering objects together using the anti-k_{T} clustering algorithm [109] [110]. The anti-k_{T} algorithm defines a distance between two objects as:

$$d_{ij} = \min(p_{\mathrm{T}}^{-2}(i), p_{\mathrm{T}}^{-2}(j)) \frac{(\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2}{R^2}, \tag{2.11}$$

where $R = 0.4$ is the radius parameter. The anti-k_{T} algorithm proceeds hierarchically starting to cluster together the two objects with the smallest distance and continuing with larger distances. A cluster becomes a jet when its p_{T}^{-2} is smaller than any other

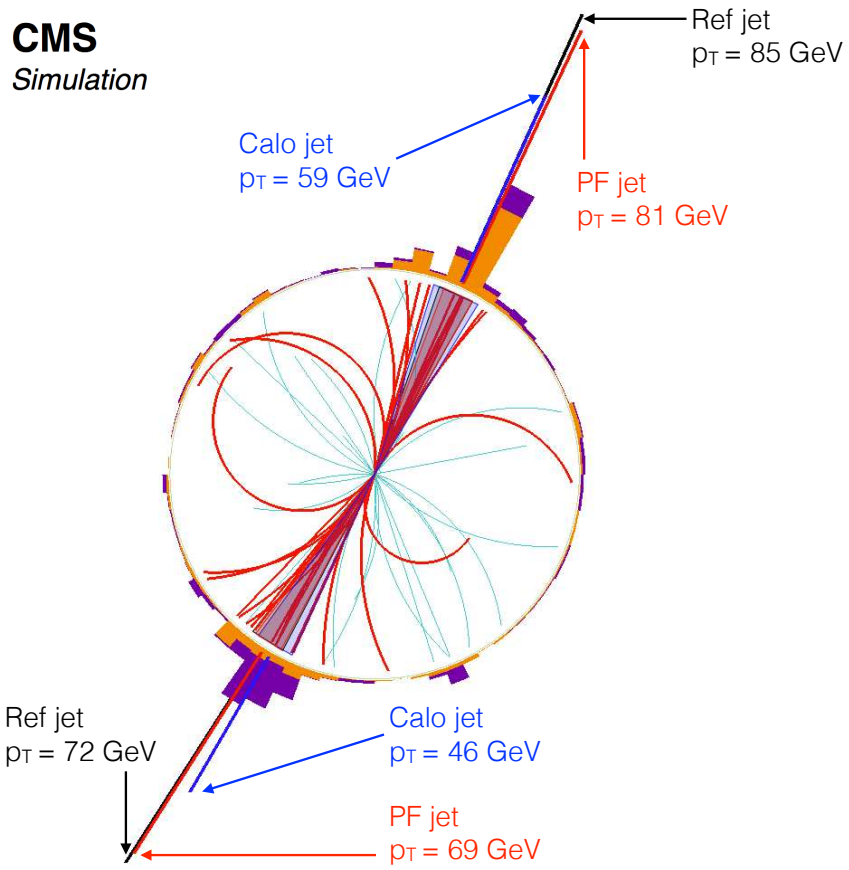
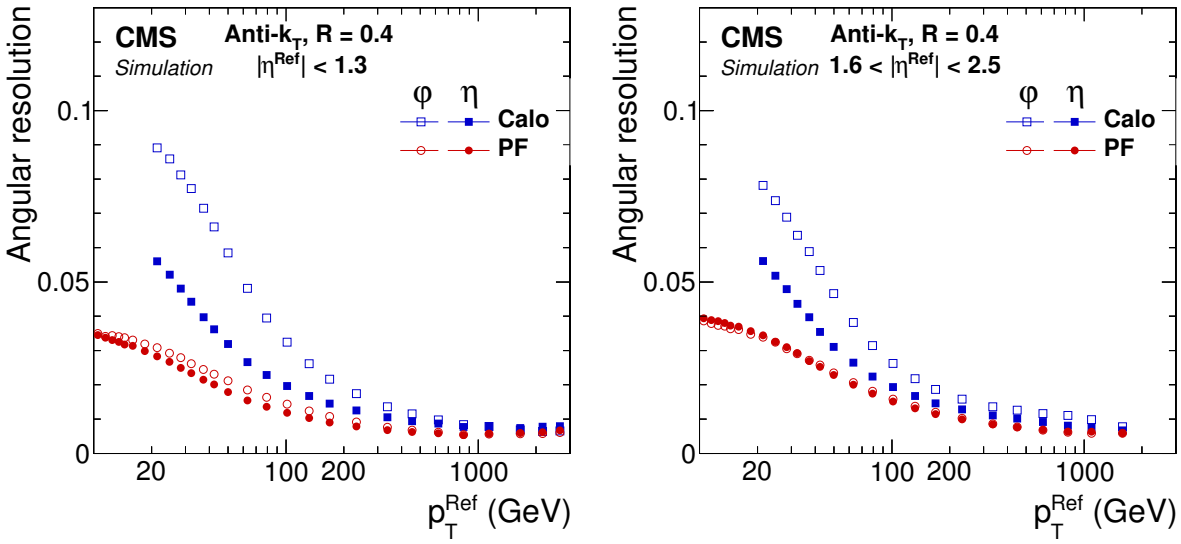


Figure 2.21: Jet reconstruction in a simulated dijet event. The particles clustered in the two PF jets are displayed with a thicker line. For clarity, particles with $p_T < 1$ GeV are not shown. The PF jet transverse momentum, indicated as a radial line, is compared to the transverse momentum of the corresponding generated (Ref) and calorimeter (Calo) jets. In all cases, the four-momentum of the jet is obtained by summing the four-momenta of its constituents, and no jet energy correction is applied [79].

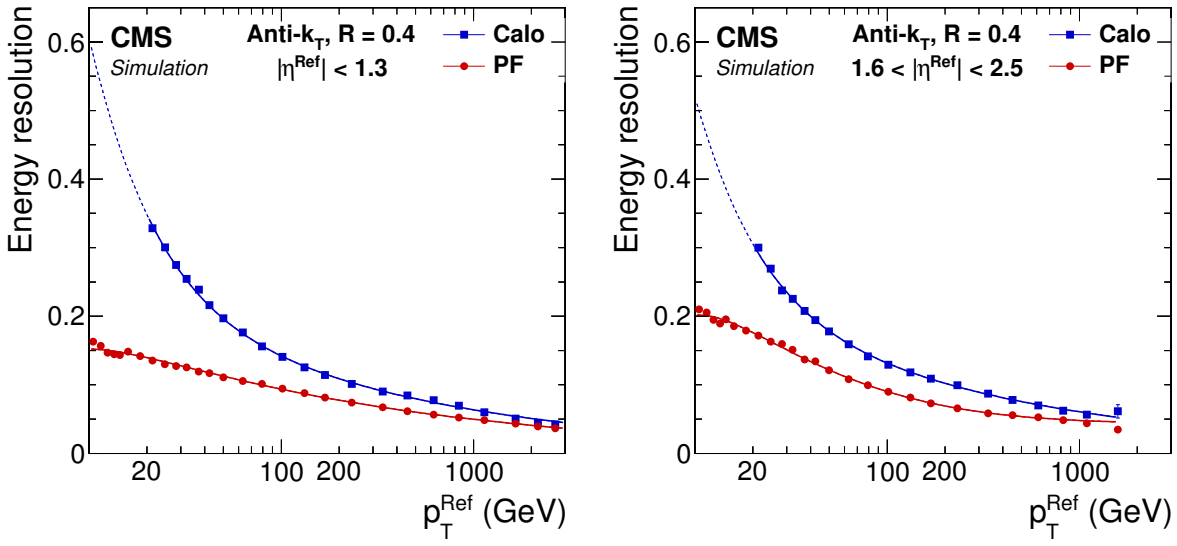


(a) (b)

Figure 2.22: Jet angular resolution in the barrel (left) and endcap (right) regions, as a function of the transverse momentum of the reference jet. The ϕ resolution is expressed in radians [79].

remaining distance.

The anti-k_T algorithm can cluster together either particles reconstructed using the Particle Flow algorithm creating *PF jets*, and energy deposits in the calorimeters, producing *Calo jets*. Figure 2.21 shows a comparison between the PF jets and Calo jets with respect to the simulated reference (Ref jet) for a simulated dijet event. The reconstructed jet direction and energy is closer to the reference for PF jets with respect to Calo jets thanks to the PF interpretation using combined information coming from the tracker and calorimeters. The improvements in the angular resolution on the jet direction, achieved by using the PF algorithm, is shown in Figure 2.22. The different energy response between the Ref jet energy and the PF jet energy is corrected in order to bring the ratio between the two to unity [111]. The jet corrected energy resolution is then defined as the Gaussian width of the ratio between the corrected and reference jet energies [79](Figure 2.23).



(a) (b)

Figure 2.23: Jet energy resolution as a function of the reference jet transverse momentum in the barrel (left) and in the endcap (right) regions [79].

Missing Transverse Energy

The Particle Flow algorithm evaluates also the *missing transverse energy* (MET) $E_{\mathrm{T}}^{\mathrm{miss}}$, sometimes indicated as *missing transverse momentum* $p_{\mathrm{T}}^{\mathrm{miss}}$, defined as the negative modulus of the vector sum of the transverse momenta of all the reconstructed particles in one event [112]. MET has an important role for the identification of weakly interacting particles, like neutrinos and for studies of invisible particles foreseen by theories beyond the Standard Model. Reconstruction of MET requires the reconstruction of all the particles in an event and a hermetic detector. Figure 2.24 shows the improvement in relative $p_{\mathrm{T}}^{\mathrm{miss}}$ resolution and direction resolution using the Particle Flow description with respect to the reconstructed MET when using calorimetric information only [79].

Reconstruction in the Offline Software Infrastructure

Data coming from the HLT enters the CERN Tier-0 with a rate of 1 kHz. At the Tier-0, data is reconstructed for monitoring and calibration in the *express processing*. In the *Prompt Calibration Loop* (PCL), express data are used as input to automated calibration

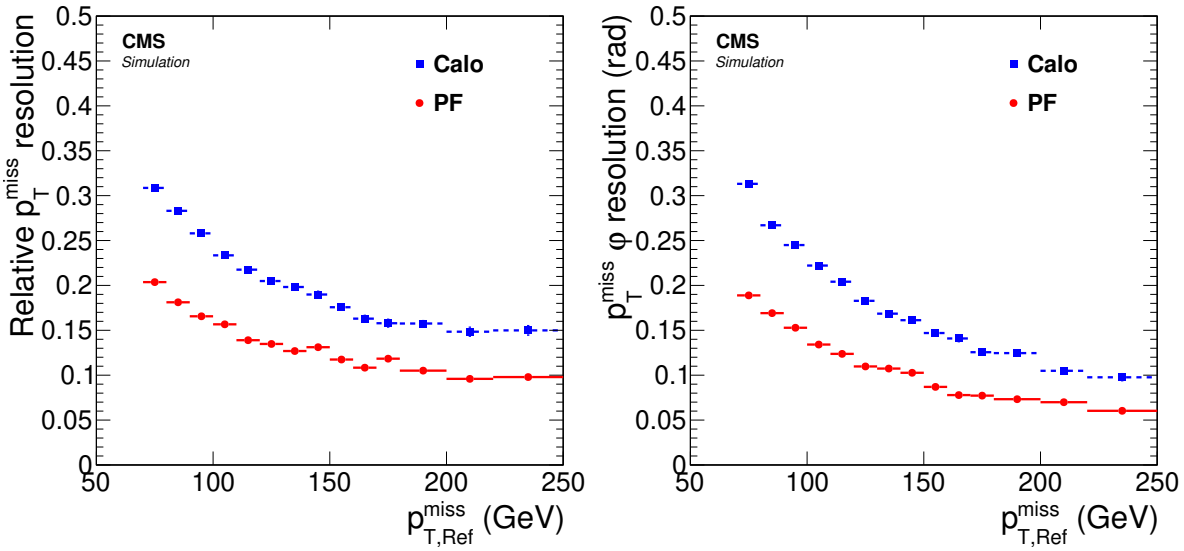


Figure 2.24: Relative p_T^{miss} resolution and resolution on its direction as a function of the generated $p_{T,Ref}^{miss}$ for a simulated $t\bar{t}$ sample [79].

workflows running online or at the Tier-0. During the *Prompt-RECO*, Physics events are reconstructed consuming calibrations computed by PCL using the *CMS Software (CMSSW)*.

Data from collisions are also reconstructed in Tier-1s and in part of the HLT farm. Data is then grouped in datasets based on physics interests and stored on tape. These datasets are also transmitted to the Tier-1s through a high-speed optical fiber network.

In order to have a small data format containing only the information that most of the physics analyses could use, the Mini-AOD data format has been developed [113]. CMS decides globally where data is placed dividing it in datasets, based on physics interests and keeping track of the datasets' locations.

2.4 LHC Upgrade Program

On May 30th 2013, at a special meeting of the European Commission in Brussels, the CERN Council adopted an updated European Strategy for Particle Physics [114]:

> "The discovery of the Higgs boson is the start of a major programme of work to measure this particle's properties with the highest possible precision

> for testing the validity of the Standard Model and to search for further new physics at the energy frontier. The LHC is in a unique position to pursue this programme. Europe's top priority should be the exploitation of the full potential of the LHC, including the high-luminosity upgrade of the machine and detectors with a view to collecting ten times more data than in the initial design, by around 2030."

The update was required to set a schedule which takes in consideration the recent discovery of the Higgs boson at the LHC. The ongoing LHC upgrade program spans over 20 years and aims at delivering an overall integrated luminosity of 3000 fb^{-1} by around 2030 (Figure 2.25), containing nearly 50 million Higgs bosons to analyze.

Upgrades and maintenance operations on the LHC and CMS are implemented during four shutdowns:

1. 2013 $\rightarrow$ 2014, Long Shutdown 1 (LS1): Center-of-Mass energy increases from 8 TeV to 13 TeV. Instantaneous luminosity reaches the value of $1.5 \cdot 10^{34}$ cm^{-2} s^{-1}, 50% higher than the LHC nominal design value.

2. 12/2016 $\rightarrow$ 04/2017, Extended Year-End Technical Stop (EYETS): The CMS enters its Phase-1 with the replacement of its Pixel detector and Hadron Calorimeter. This upgrade of the Pixel detector is described in details in Section 2.5.

3. 2018 $\rightarrow$ 2019, Long Shutdown 2 (LS2): The LINAC4 [115] replaces LINAC2, allowing instantaneous luminosity to reach $2.2 \cdot 10^{34}$ cm^{-2} s^{-1}, corresponding to a pile-up of nearly 90 collisions per bunch crossing.

4. 2023 $\rightarrow$ 2025, Long Shutdown 3 (LS3): The LS3 will start the High Luminosity LHC era (HL-LHC). The instantaneous luminosity will range between $5 \cdot 10^{34}$ cm^{-2} s^{-1} and $7 \cdot 10^{34}$ cm^{-2} s^{-1}, producing between 140 and 200 collisions per bunch crossing. CMS will enter its Phase-2: new tracking detectors [116] and new end-cap instrumentation for calorimetry [117] along with an increase in the capability of the online trigger and data acquisition systems.

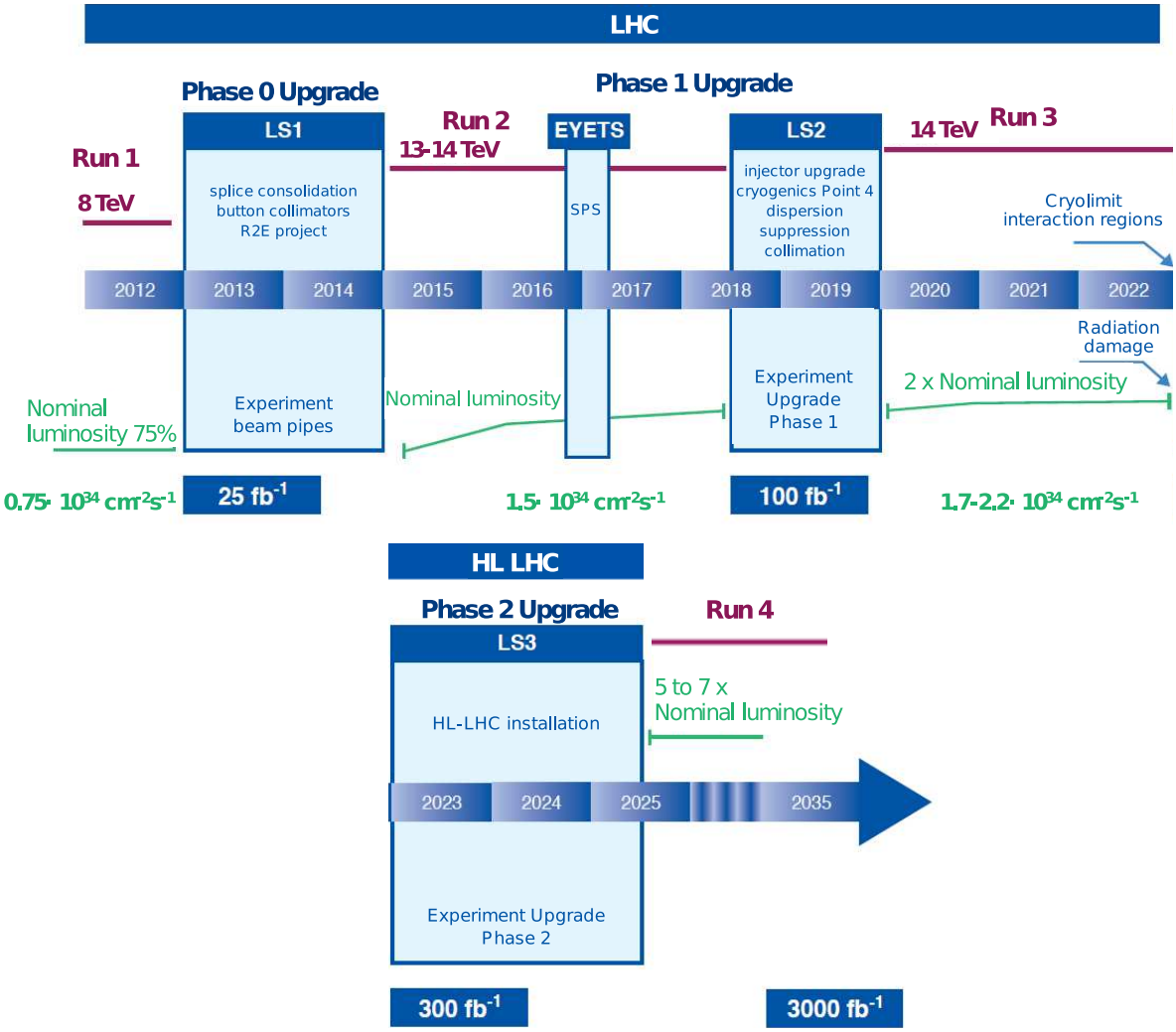


Figure 2.25: Time-line of the LHC scheduled upgrades and expected performance.

2.5 Phase-1 Pixel Detector upgrade

During the *Extended Year-End Technical Stop* (EYETS), occurring between the end of 2016 and the first half of 2017, the Pixel Detector was completely replaced (Figure 2.28). The motivation for this upgrade is connected to the declining performance of the existing detector in a high pile-up environment [118]:

- The pixel ROC starts to loose data because of insufficient buffer size and read-out speed. At a luminosity of $1 \cdot 10^{34}$ cm^{-2} s^{-1} with a bunch crossing time of 25 ns, the estimated hit inefficiency is about 4%. This inefficiency raises to about 15% with

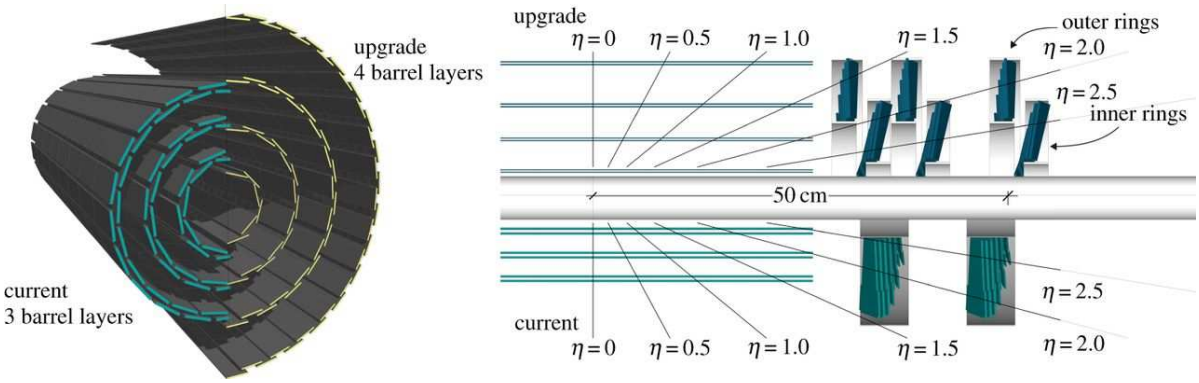


Figure 2.26: Comparison between the Phase-0 and the Phase-1 CMS Pixel Detector. The current and upgraded detectors are shown side by side in the transverse view (left). A longitudinal view shows the current detector below the beam pipe, compared to the upgraded detector on top (right) [118].

at a luminosity of $1 \cdot 10^{34}$ cm^{-2} s^{-1}. The replacement of the ROC will eliminate this inefficiency.

- An increased event complexity in high pile-up events leads to higher difficulty in track reconstruction. Requesting the same level of tracking efficiency would require accepting more fake tracks. The new Pixel Detector includes four barrel layers and three end-cap disks per direction (Figure 2.26). The four-hits coverage of the new Pixel Detector would help in keeping the efficiency high while still keeping the fake-rate under control.
- Hit efficiency deteriorates with radiation damage.
- The upgraded Pixel Detector has a lower material budget thanks to light support structures and cooling (Figure 2.27).

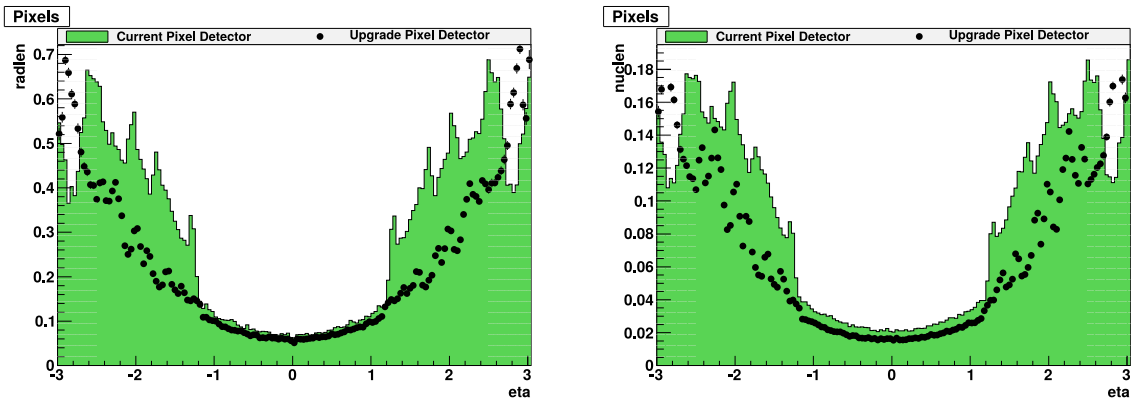


Figure 2.27: Material budget of the phase-0 (in green) and the upgraded phase-1 (black dots) Pixel Detectors system in number of radiation lengths (left) and nuclear interaction lengths (right) as a function of η [118]. Contributions to the material budget come from active materials (the silicon sensors) or passive materials, e.g. support structures, cables, cooling pipes or gas.

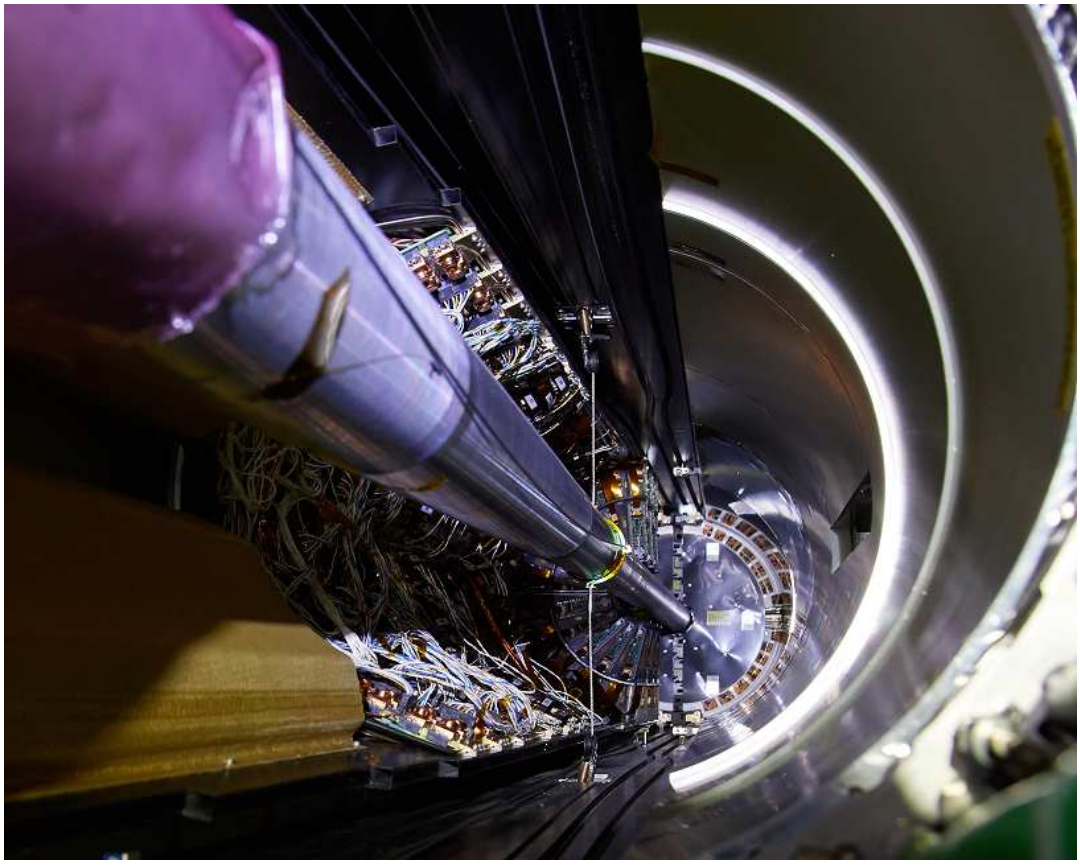


Figure 2.28: Installation of the Phase-1 Pixel Detector [119].

Like in the Phase-0 Pixel Detector, the new geometry presents two main sub-detectors: a barrel (BPix) and a forward (FPix) regions. These two sub-detectors provide an almost hermetic four-hits coverage up to $\eta = 2.5$. The BPix region is made of 79 million pixels (48 million pixels in Phase-0 BPix) distributed on four 548.8 mm long cylindrical layers, at radial positions of $R = \{29.5, 68, 109, 160\}$ mm. The FPix region comprises three disks on each side of the BPix sub-detector, located at $z = \{\pm\ 29.1, \pm\ 39.6, \pm\ 51.6\}$ cm. Each disk contains an inner and outer ring that provide a radial coverage from 4.5 cm to 16.1 cm. The overall FPix pixels count is about 45 million pixels, compared to 18 million pixels of the Phase-0 FPix layers. Figure 2.28 shows a picture of the installation of the Phase-1 Pixel barrel.

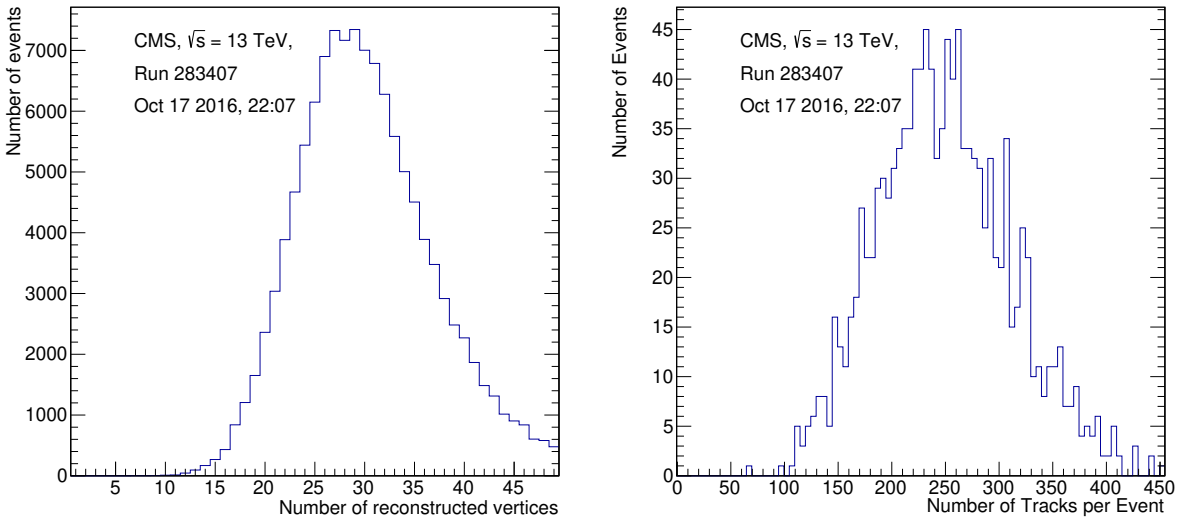
Chapter 3

CMS Tracking: status and upgrade

A track is the result of the fit of a set of measurement points (hits) in the Tracker that provides an estimate of a charged particle trajectory.
As it was previously described in Section 2.3.2, the reconstruction of many physics objects depends on tracking:

- The momentum of charged particles (muons, electrons and charged hadrons).
- Primary and secondary vertices identification.
- Reconstruction and tagging of jets coming from b-quark hadronization.
- Reconstruction and tagging of jets coming from τ decay.

Precise track reconstruction becomes challenging when pile-up grows, as the number of vertices and the number of tracks increase (Figure 3.1), making the classification of hits produced by the same charged particle become a combinatorial problem.



(a) (b)

Figure 3.1: On the left the distribution of the number of reconstructed interaction points (vertices) in each event for the run shown in Figure 2.4. On the right the distribution of the number of tracks per event for the same run.

3.1 Local Reconstruction

Local Reconstruction in the Pixel Detector

Charged particles, traversing the layers of the Pixel Detector, release charge carriers, as discussed in Section 2.2.1. An analog-to-digital converter (ADC) transforms the analog pulse into a digitized value, which can be then used in the reconstruction software. To mitigate the *out of time pile-up*, remnants of signals of a previous event, a threshold on the signal-to-noise ratio is applied, and only those pixels whose signal exceeds this threshold will be read in the next step.

Once in the computing system, the first step of the track reconstruction consists in the *local reconstruction*. During the local reconstruction these digitized information are unpacked and signals in neighboring pixels are grouped together in a process called *clusterization*. All the side and corner adjacent pixels whose charge is above the readout threshold will be included in the same pixel cluster. Each cluster's total charge has to exceed the equivalent charge of 4000 electrons in order to move on to the next step [81].

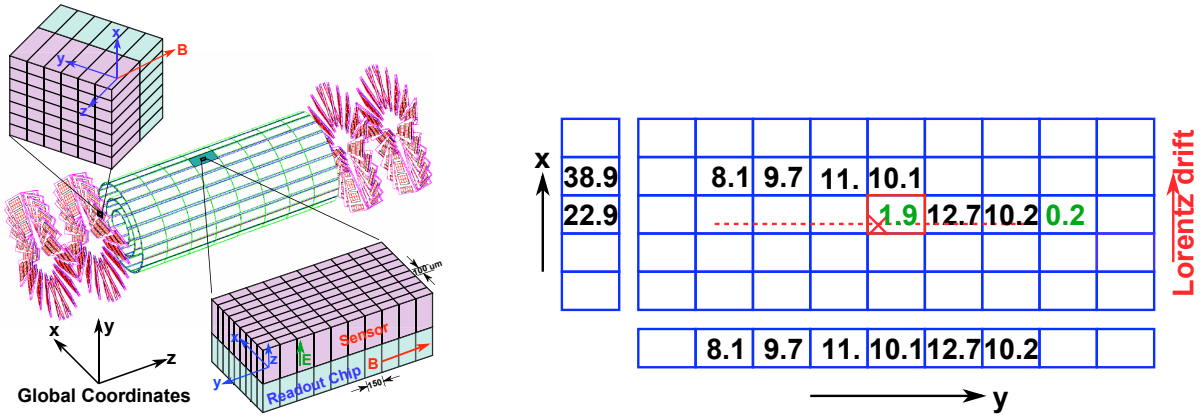
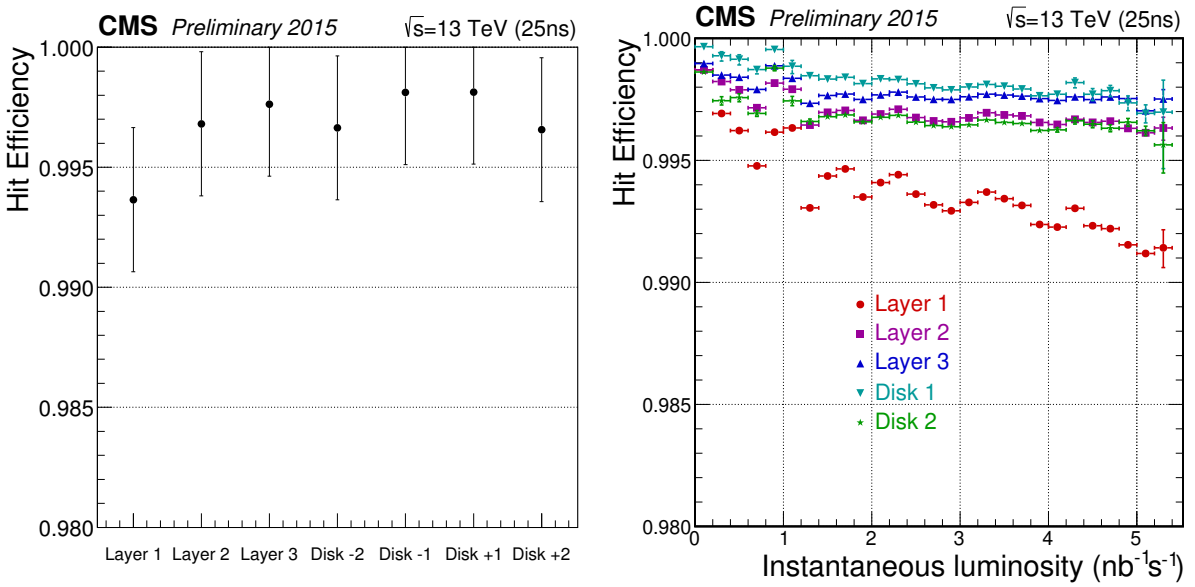


Figure 3.2: Left: Geometry of the CMS barrel and forward Pixel Detectors and the corresponding coordinate systems. Right: Charge deposition shown in thousands of electrons. In green, the pixels whose charge does not exceed the threshold and are not included in the cluster. The dotted red line shows the projection of the particle's trajectory. The red cross indicates the point in which the trajectory intersects the module's plane [120].

The shape of the clusters and the weights of the signals coming from each pixel are taken into account for the determination of the *hit position* (the center of gravity of the cluster), and its uncertainty along the local coordinates. An example of a cluster in the CMS Pixel Detector is shown in Figure 3.2.
The reconstruction of the hit position from the cluster includes two steps:

- Simulated cluster shapes are projected along the x and y local coordinates. They are extracted as function of the simulated global position of the cluster and track impact angles. This step is repeated for a simulated large sample of tracks. The generated cluster projection shapes are called *templates*.
- The results of the first steps are then used to generate information on biases, errors, corrections and goodness-of-fit. The expected cluster shape is moved until it best matches the observed cluster shape.

The result of this procedure is a bin number for each of the two dimensions and with the value of χ^2 [121].
The *hit efficiency* is defined as the probability to find a cluster in a sensor that has been traversed by a charged particle. Figure 3.3 shows that every layer produces hits with very good efficiency. However, the hit efficiency decreases as the instantaneous luminosity increases, as the limits of the working specifications of the ROC are reached.



(a) (b)

Figure 3.3: On the left, the average hit efficiency for layers or disks in the Pixel Detector excluding defective modules [122]. On the right, the average hit efficiency as a function of instantaneous luminosity [122].

Local Reconstruction in the Strip Detector

As described in Section 2.2.1, the front-end driver (FED) receives pulses from the read-out electronics of the strips and executes zero-suppression.
Clustering is seeded by sensors with a charge that is at least three times greater than the strip noise. For each seed, neighboring strips are included in the cluster if the strip charge is more than twice the strip noise. Eventually, a cluster produces a hit if its total charge exceeds the cluster noise by at least a factor 5. The cluster noise is defined by:

$$\sigma_{\text{cluster}} = \sqrt{\sum_i \sigma_i}, \tag{3.1}$$

where σ_i is the noise of the strip i of the cluster. The hit position is reconstructed from the cluster information by correcting the charge-weighted average of its strip position by $10/20\mu$m (TIB/TOB) to account for the Lorentz drift.

3.2 Track Reconstruction

Hits are the output of the local reconstruction. The reconstruction of the trajectories of the charged particles that left those hits is carried out in the *track reconstruction*.

3.2.1 Track Parameterization

In a solenoidal homogeneous magnetic field, the trajectory of a charged particle is a helix. The helix can be described by five parameters, two for the position, two for the direction and one for the curvature at a given reference position $v = (v_x, v_y, v_z)$ along the track. This defines a *state vector*.
In the CMS track reconstruction, the perigee point v is the point of closest approach of the helix to the z-axis.
The beam spot position $bs = (bs_x, bs_y, bs_z)$ is also used in the determination of the value of helix parameters [123].

The parameterization used in the CMS tracking is the following [124]:

1. The signed inverse momentum in GeV^{-1}

$$qoverp = q/|\vec{p}|. \tag{3.2}$$

2. $\lambda = \frac{\pi}{2} - \theta$, where θ is the polar angle at the point v.
3. The azimuthal angle ϕ at the point v.
4. The signed distance d_{xy} in the x-y plane between the straight line passing through (v_x, v_y) with the azimuthal angle ϕ and the point (bs_x, bs_y), in cm:

$$d_{xy} = -(v_x - bs_x)\sin\phi + (v_y - bs_y)\cos\phi \tag{3.3}$$

5. The signed distance d_{sz} in the s-z plane between the straight line l passing through v with angles (ϕ, λ), and the projection of the point bs on the s-z plane, in cm. The s-axis is defined as the projection of the line l on the x-y plane:

$$d_{sz} = (v_z - bs_z)\cos\lambda - [(v_x - bs_x)\cos\phi + (v_y - bs_y)\sin\phi]\sin\lambda \tag{3.4}$$

In this parameterization, the transverse impact parameter becomes:

$$d_0 = -d_{xy}, \tag{3.5}$$

the longitudinal impact parameter:

$$d_z = \frac{d_{sz}}{\cos\lambda}. \tag{3.6}$$

The transverse momentum is defined as:

$$p_{\mathrm{T}} = |p| \sin\theta. \tag{3.7}$$

3.2.2 Estimation of track parameters using a Kalman filter

The track reconstruction is based on a combinatorial track finder relying on Kalman filtering [8].

The *Kalman filter* (KF) is an iterative and adaptive process used for estimating the states of dynamic systems. The KF can be applied to track reconstruction assuming that the track can be interpreted as a discrete dynamic system [125] [126].

Consider a tracking device made of n sensitive surfaces. The state of the track at the surface k will be described by the state vector $\mathbf{p}_k$.
Given the state vector $\mathbf{p}_{k-1}$, which describes the state of the track at the surface $k-1$, the propagated state vector at the next surface $\mathbf{p}_k$ is defined by the *system equation*:

$$\mathbf{p}_k = \mathbf{f}_k(\mathbf{p}_{k-1}) + \mathbf{P}_k\delta_k. \tag{3.8}$$

In the system equation the following contributions can be identified:

- $\mathbf{f}_k$ is the model that describes a charged particle in a magnetic field between the surfaces $k-1$ and k.
- The effect of the interaction with the detector's material on different track parameters is described by $\mathbf{P}_k$. The term $\mathbf{P}_k\delta_k$ is fundamental in order to take into account energy loss in the detector material, where δ_k is a Gaussian distributed

random variable with average 0. The knowledge of the detector and its material budget distribution is used to derive the covariance matrix $\mathbf{Q}_k$ of the noise.

In the following, it will be assumed that $\mathbf{f}_k$ can be approximated with its first order Taylor expansion with derivative matrix $\mathbf{F}_k$:

$$\mathbf{f}_k(\mathbf{p}_{k-1}) \approx \mathbf{F}_k \mathbf{p}_{k-1}. \tag{3.9}$$

A measurement $\mathbf{m}_k$ is a function of the state vector $\mathbf{p}_k$:

$$\mathbf{m}_k = \mathbf{h}_k(\mathbf{p}_k) + \epsilon_k. \tag{3.10}$$

$\mathbf{h}_k$ projects the state vector's parameters onto the measurement.
The variable ϵ_k is Gaussian distributed noise with average 0 and its covariance matrix $\mathbf{V}_k$ is assumed to be known. Also in this case, it will be assumed that $\mathbf{h}_k$ is linear or can be approximated with its first order Taylor expansion with derivative matrix $\mathbf{H}_k$:

$$\mathbf{h}_k(\mathbf{p}_k) \approx \mathbf{H}_k \mathbf{p}_k. \tag{3.11}$$

The KF offers three different functionalities depending on the relation between the number j of measurements used to produce the estimate $\mathbf{p}_k$ and k:

1. $j < k$: the state vector is extrapolated to a *future surface* k taking into account j previous measurements. This step is called *prediction*.

2. $j = k$: the KF uses the previous and current measurements to estimate the present state. This step is called *filtering*.

3. $j > k$: the KF estimates a past state based on the knowledge of the past, present and future measurements. This step is called *smoothing*.

In the following, the footer $A|B$ will indicate that an operand refers to the surface A and its value was estimated using a number B of measurements. Therefore, the state vector at the surface k based on j measurements will be indicated with $\mathbf{p}_{k|j}$ and its covariance matrix $\mathbf{C}_{k|j}$.
Furthermore, the residual $\mathbf{m}_k - \mathbf{H}_k \mathbf{p}_{k|j}$ will be denoted with $\mathbf{r}_{k|j}$ and its covariance matrix with $\mathbf{R}_{k|j}$.

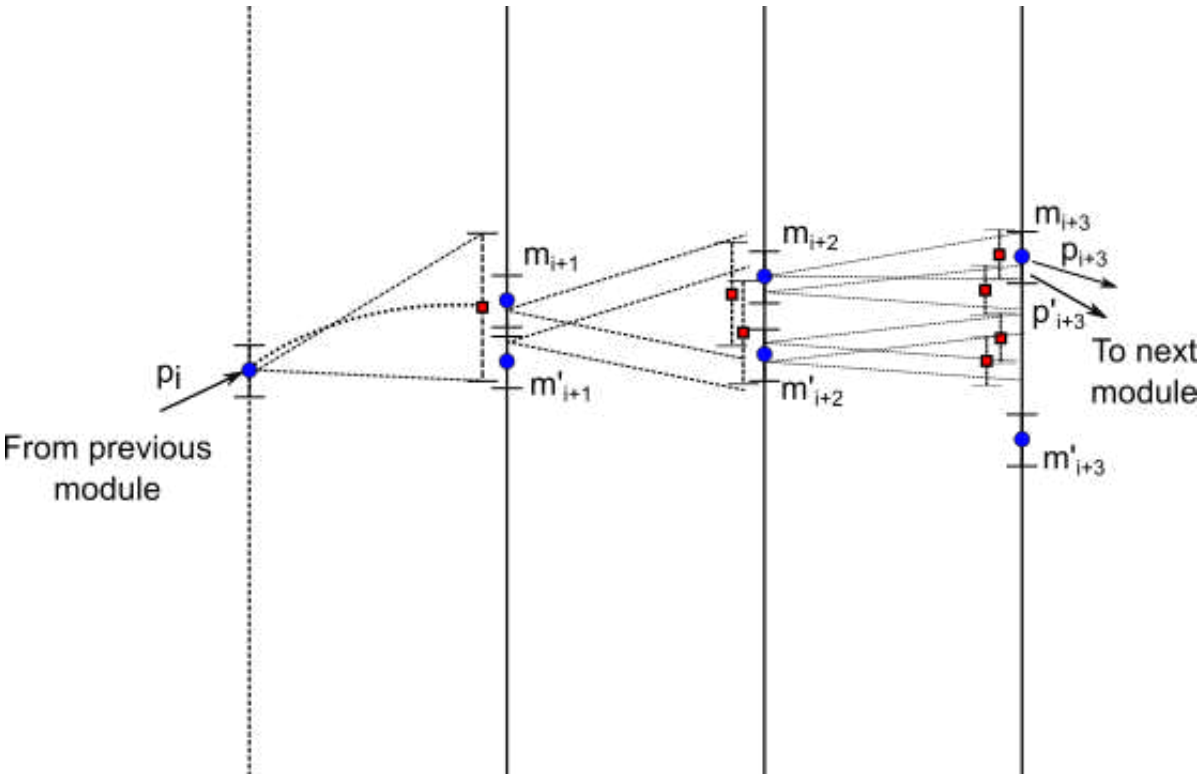


Figure 3.4: Progressive adjustment of the fitted parameters by propagation. The combinatorial character is visible when multiple measurements are found on a surface.

Prediction

Suppose that $\mathbf{p}_{k-1|k-1}$ and $\mathbf{C}_{k-1|k-1}$ have been evaluated.
In the prediction approach it is possible to calculate $\mathbf{p}_{k|k-1}$:

$$\mathbf{p}_{k|k-1} = \mathbf{F}_k \mathbf{p}_{k-1|k-1} \tag{3.12}$$

and its covariance matrix:

$$\mathbf{C}_{k|k-1} = \mathbf{F}_k \mathbf{C}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{P}_k \mathbf{Q}_k \mathbf{P}_k^T. \tag{3.13}$$

Given the measurement m_k the residuals can be computed:

$$\mathbf{r}_{k|k-1} = \mathbf{m}_k - \mathbf{H}_k \mathbf{p}_{k|k-1} \tag{3.14}$$

and their covariance matrix:

$$\mathbf{R}_{k|k-1} = \mathbf{V}_k + \mathbf{H}_k \mathbf{C}_{k|k-1} \mathbf{H}_k^T \tag{3.15}$$

Filtering

In the filtering approach, the state vector $\mathbf{p}_{k|k}$ is evaluated given the knowledge of the measurement $\mathbf{m}_k$ from the state vector $\mathbf{p}_{k|k-1}$:

$$\mathbf{p}_{k|k} = \mathbf{p}_{k|k-1} + \mathbf{K}_k \mathbf{r}_{k|k-1} = \mathbf{p}_{k|k-1} + \mathbf{K}_k(\mathbf{m}_k - \mathbf{H}_k \mathbf{p}_{k|k-1}), \tag{3.16}$$

where we have denoted with $\mathbf{K}_k$ the *Kalman gain matrix*:

$$\mathbf{K}_k = \mathbf{C}_{k|k-1} \mathbf{H}_k^T (\mathbf{V}_k + \mathbf{H}_k \mathbf{C}_{k|k-1} \mathbf{H}_k^T)^{-1}. \tag{3.17}$$

By denoting with $\mathbf{G}_k = \mathbf{V}_k^{-1}$ and using the relation:

$$\mathbf{C}_{k|k-1}^{-1} = \mathbf{C}_{k|k}^{-1} - \mathbf{H}_k^T \mathbf{G}_k \mathbf{H}_k, \tag{3.18}$$

it is possible to write the Kalman gain matrix as follows:

$$\mathbf{K}_k = \mathbf{C}_{k|k} \mathbf{H}_k^T \mathbf{G}_k. \tag{3.19}$$

The covariance matrix can now be written as the sum of the prediction and a correction term that takes into account the current measurement:

$$\mathbf{C}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{C}_{k|k-1}. \tag{3.20}$$

The size of this correction is given by the Kalman gain $\mathbf{K}_k$.
The filtered residuals and their covariance matrix can also be written in terms of the Kalman gain:

$$\mathbf{r}_{k|k} = \mathbf{m}_k - \mathbf{H}_k \mathbf{p}_{k|k} = (\mathbf{I} - \mathbf{H}_k \mathbf{K}_k) \mathbf{r}_{k|k-1}, \tag{3.21}$$

$$\mathbf{R}_{k|k} = (\mathbf{I} - \mathbf{H}_k \mathbf{K}_k) \mathbf{V}_k = \mathbf{V}_k - \mathbf{H}_k \mathbf{C}_{k|k} \mathbf{H}_k^T. \tag{3.22}$$

The total χ^2, accounting for all the previous $k-1$ filtering steps, can be updated as well:

$$\chi^2_k = \chi^2_{k-1} + \mathbf{r}^T_{k|k}\mathbf{R}^{-1}_{k|k}\mathbf{r}_{k|k}. \tag{3.23}$$

Smoothing

The smoothing is usually applied once the last measurement $\mathbf{m}_n$ has been obtained. The state vector $\mathbf{p}_n$ contains all the information coming from all the measurements and can be used to update all the filtered estimates:

$$\mathbf{p}_{k|n} = \mathbf{p}_{k|k} + \mathbf{A}_k(\mathbf{p}_{k+1|n} - \mathbf{p}_{k+1|k}). \tag{3.24}$$

The matrix $\mathbf{A}_k$ is called *smoother gain matrix*:

$$\mathbf{A}_k = \mathbf{C}_{k|k}\mathbf{F}^T_{k+1}\mathbf{C}^{-1}_{k+1|k} \tag{3.25}$$

The smoothed covariance matrix is given by:

$$\mathbf{C}_{k|n} = \mathbf{C}_{k|k} + \mathbf{A}_k(\mathbf{C}_{k+1|n} - \mathbf{C}_{k+1|k})\mathbf{A}^T_k. \tag{3.26}$$

The smoothed residuals and relative covariance matrix can be written as follows:

$$\mathbf{r}_{k|n} = \mathbf{m}_k - \mathbf{H}_k\mathbf{p}_{k|n} = \mathbf{r}_{k|k} - \mathbf{H}_k(\mathbf{p}_{k|n} - \mathbf{p}_{k|k}) \tag{3.27}$$

$$\mathbf{R}_{k|n} = \mathbf{R}_{k|k} - \mathbf{H}_k\mathbf{A}_k(\mathbf{C}_{k+1|n} - \mathbf{C}_{k+1|k})\mathbf{A}^T_k\mathbf{H}^T_k = \mathbf{V}_k - \mathbf{H}_k\mathbf{C}_{k|n}\mathbf{H}^T_k. \tag{3.28}$$

As for the filtering approach, a χ^2 can be written as:

$$\chi^2_k = \mathbf{r}^T_{k|n}\mathbf{R}^{-1}_{k|n}\mathbf{r}_{k|n}, \tag{3.29}$$

which is χ^2-distributed with $\dim(\mathbf{m}_k)$ degrees of freedom.

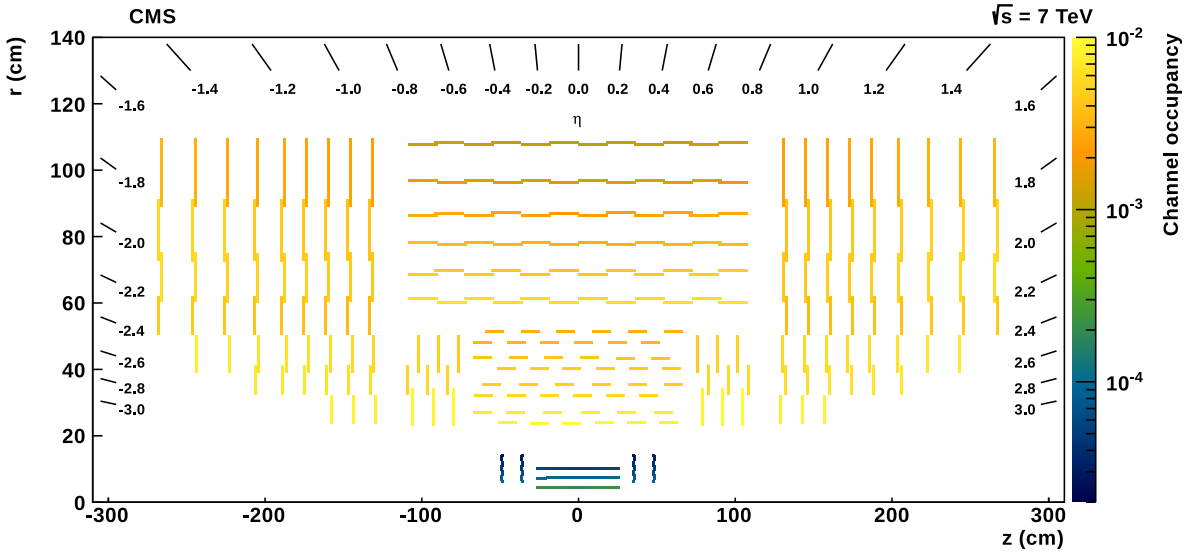


Figure 3.5: Channel occupancy for the CMS silicon Tracker in events from randomly selected non-empty LHC bunch crossings. The average number of pile-up events is nine [81].

Kalman filter in CMS track reconstruction

When multiple measurements are compatible with the propagated state vector, tracking becomes a combinatorial problem. The CMS tracking strategy is based on the *Combinatorial Kalman filter*. The hits which are closest to the prediction in a suitable χ^2 statistics are selected for the inclusion in the filter stage. This stage of the track reconstruction goes under the name *track building* or *pattern recognition*.

The KF alternates the use of prediction and filter for each encountered layer. This process continues until the propagation has reached the outermost layers, or a maximum number of skipped layers is reached. If no hit is matched on a given layer, the KF prediction is given the possibility to skip it and propagate to the next-to-next layer to improve efficiency.

In order to reduce the initial number of combinations to update and propagate, a track segment called *track seed* needs to be computed. The seed provides an initial estimate of the track parameters to be used as initial state vector by the Kalman filter in the track building. Track seeding requires at least two hits and a hypothesis of the beam-spot region or vertex, or three hits. Hits coming from the Pixel Detector are the optimal candidates for the track seeding step as they provide precise three-dimensional measurements and

low occupancy (Figure 3.5). Once the track building is complete, smoothing can be applied and the track parameters can be fitted. Eventually, a *track selection*, based on the track's normalized χ^2, contributes to the rejection of those tracks built using hits produced by different charged particles.

3.2.3 Reconstruction Quality Criteria

The main criteria for the quality of the track reconstruction that can be used to quantify the performance in tracking procedures and algorithms are:

- *efficiency*,
- *fake-rate*,
- *execution time*.

The efficiency indicates the fraction of the simulated tracks, N_{sim}, that have been associated with at least one reconstructed track, N_{rec}.

$$\text{efficiency} = \frac{N_{rec}}{N_{sim}} \tag{3.30}$$

It can be estimated by studying simulated events: a reconstructed track is associated with a simulated track if more than 75% of the hits that it contains come from the same simulated track.
The fake-rate is defined as the fraction of all the reconstructed tracks which are not associated uniquely to a simulated track. In the case of a fake-track the set of hits used to reconstruct the track does not belong to the same simulated track. The execution time is the CPU time spent in reconstructing tracks starting from hits.
Efficiency and fake-rate will hereby be referred to as *physics performance*; execution time will also be referred to as *computational performance*.

3.2.4 Seed generation parameters

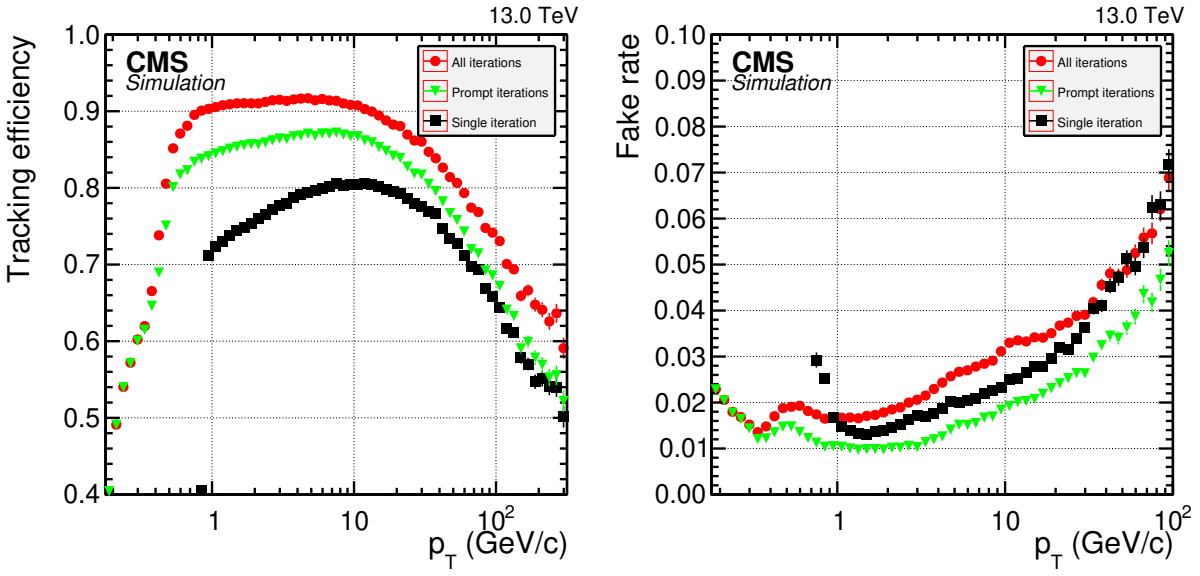
The criteria used to build seeds have a strong impact on timing and physics performance. The seeding criteria are:

- $p_{\mathrm{T}}^{\mathrm{min}}$: searching for low transverse momentum tracks can be very computationally expensive. Setting a minimum threshold for p_{T} limits the possible curvature, hence reducing the number of possible combinations of hits.
- R^{max} and z^{max}: the maximum transverse and longitudinal distance of closest approach with respect to the beam-spot. Tracks produced within a radius of less than 1 mm around the beam-spot are called *prompt tracks*. Searching for *detached tracks* requires loosening the value of R^{max}, which leads to an increase in combinatorics.
- n_{hits}: a request of a greater number of hits in the seeds leads to a more pure set of tracks and cuts can be loosened. A lower number of hits in the seed produces higher efficiency at the cost of a higher fake-rate.
  n_{hits} is controlled by the number of *seeding layers*. The seeding layers are sets of detector layers whose hits can be used to create seeds.

In principle, seeding could be created in the Outer Tracker where the density of tracks is lower. However there are several reasons why in the CMS track reconstruction, seeding is done in the pixel layers.
First, even though the track density is higher, the channel occupancy is much lower in the Pixel Detector (Figure 3.5). The channel occupancy is defined as the number of pixel or strip measurements in an event divided by the number of all active pixels or strips. Secondly, spatial measurements in the Pixel Detectors are three dimensional and their resolution is better. Finally, particles traversing the Pixel Detector have interacted with a smaller amount of material. Therefore, starting the seeding in the inner part of the Tracker leads to a higher efficiency.

3.2.5 Iterative Tracking

In order to maximize the efficiency, reduce the fake-rate and minimize the execution time, the tracking sequence *seeding, building, fitting, selection* is repeated in several *iterations* reconstructing prompt tracks with high transverse momentum in earlier iterations and searching for tracks that are more difficult to reconstruct in later stages (Figure 3.6). At the end of each iteration, the hits belonging to already found tracks are *masked*, i.e. removed from the pool of hits and the next iteration will start on a sample that does not



(a) (b)

Figure 3.6: Efficiency (a) and fake rate (b) versus simulated track transverse momentum of different iterative tracking approaches for multijet events without pile-up interactions. Only tracks with $|\eta| < 2.5$ are considered. The efficiency is determined only for tracks originating from a cylinder of radius 3.5 mm around the beam-spot, extending to $z = \pm 30$ cm along the beam-spot.
The plots shows respectively: in black a single iteration of the combinatorial track finder, in green all the iterations seeded by a prompt seed (originating from a cylinder of radius 200 μm around the beam-spot) having at least one hit in the Pixel Detector, in red all the iterations included those with displaced seeds [79].

contain them.

As an example, in the first iteration, seeds should requested to be promptly produced, to have three hits in the Pixel Detector and high p_{T}. Once the hits found in this iteration have been masked, seeds with looser constraints are produced (e.g. prompt triplets with lower p_{T}). Other iterations can be applied with more relaxed cuts depending on the requests on efficiency, fake-rate and execution time.

Offline Iterative Tracking during 2016

The 2016 offline track reconstruction spans ten iterations (Table 3.1) and its performance is shown in Figure 3.7.

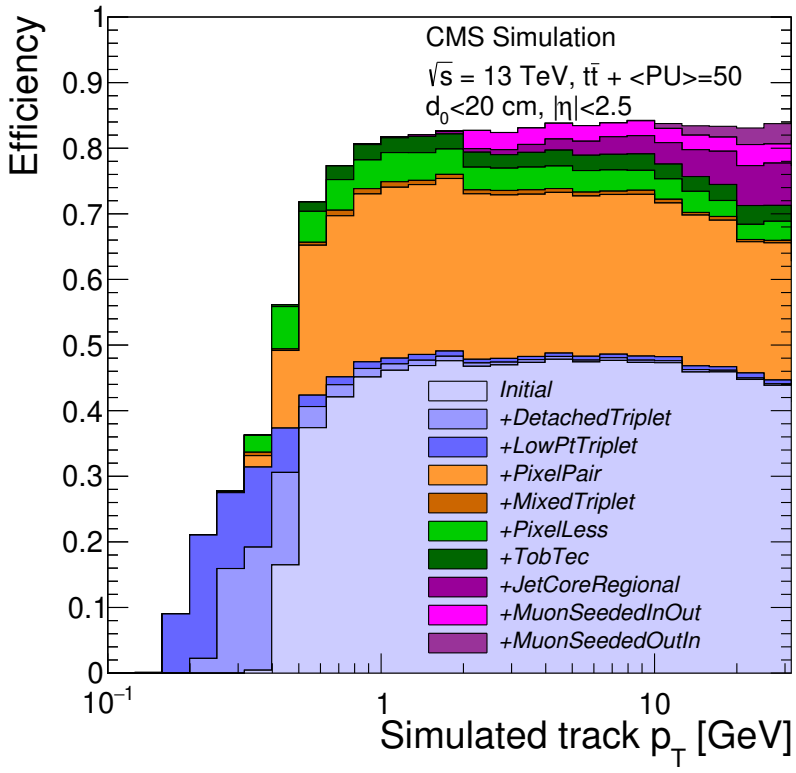


Figure 3.7: Performance of the offline iterative tracking in Run 2 during 2016 for simulated $t\bar{t}$ events with 50 superimposed pile-up collisions: efficiency vs p_T.

The *Initial, DetachedTriplet* and *LowPtTriplet* all target seeding layer sets made of three layers, hence producing triplets. The Initial iteration targets prompt tracks with high transverse momentum. Once the hits associated to tracks found in the Initial step have been removed from the hits' pool, the cut on the $R^{\max}$ can be relaxed and the DetachedTriplet can execute.

The *LowPtTriplet* step is then used to recover lower transverse momentum tracks.

PixelPair and *MixedTriplet* iterations are used to recover efficiency from tracks with a missing hit in the Pixel Detector.

PixelLess and *TobTec* steps search for very displaced tracks. For reducing the combinatorics complexity, they are executed after the previously described iterations.

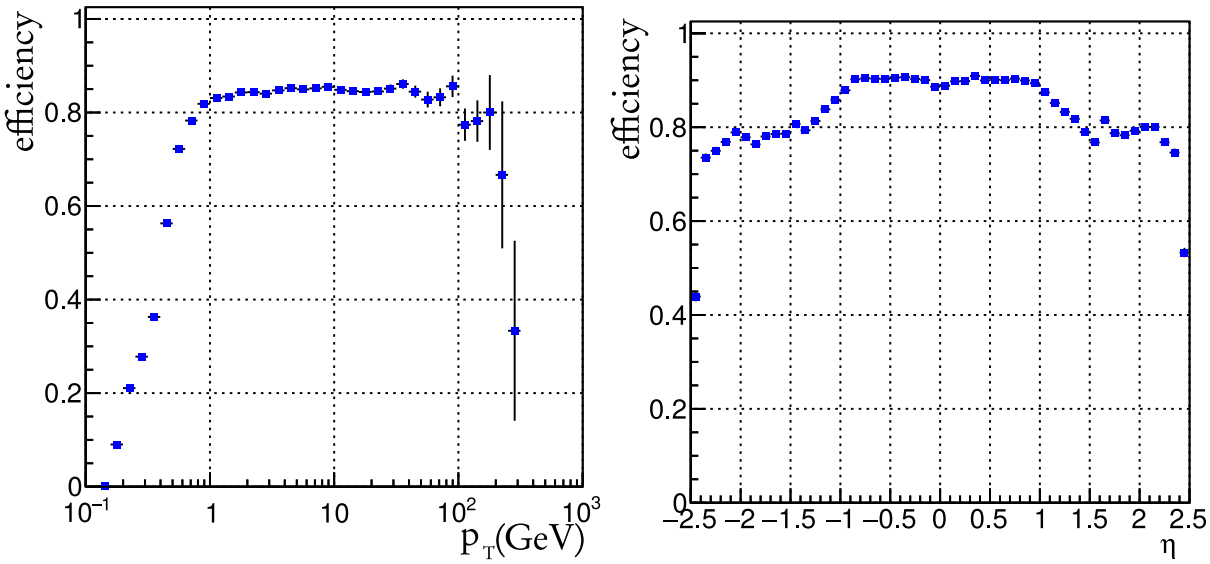
The *JetCoreRegional* step addresses tracking in very high momentum jets. Track segments produced in the Muon system are used during seeding to increase muon reconstruction efficiency in the last two iterations.

Figures 3.8 and 3.9 show the track reconstruction performance for $t\bar{t}$ events with 50

Table 3.1: Offline Tracking iterations used during 2016 data taking.

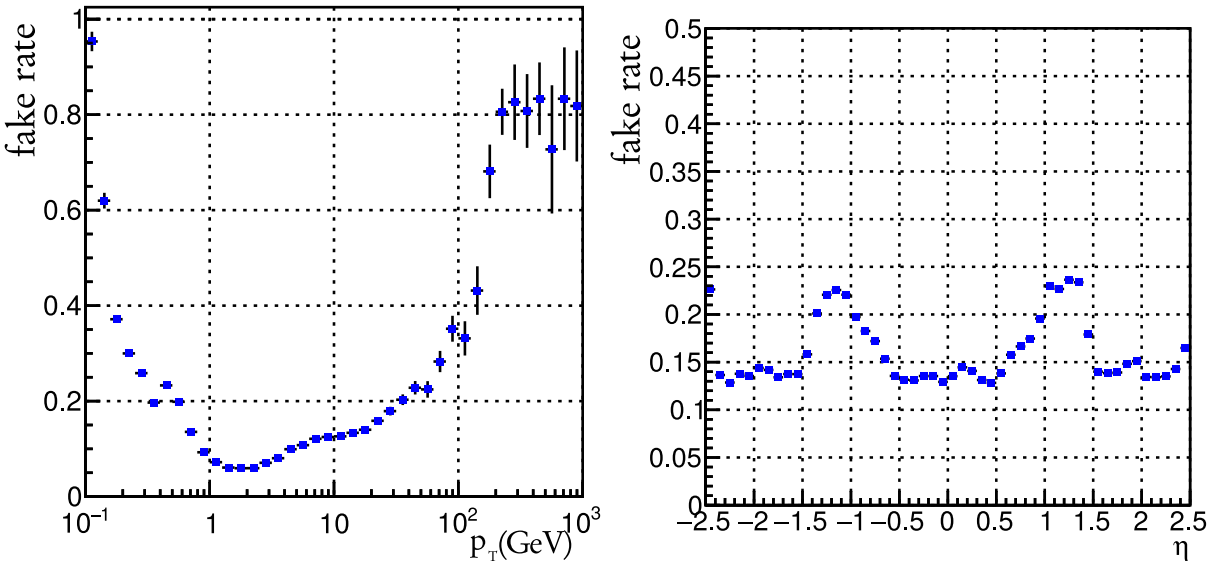
Name	Seeds	Target Tracks
Initial	pixel triplets	prompt, $p_{\mathrm{T}} > 0.6$ GeV
DetachedTriplet	pixel triplets	displaced, $R^{\mathrm{max}} = 5$ cm
LowPtTriplet	pixel triplets	prompt, $p_{\mathrm{T}} > 0.3$ GeV
PixelPair	pixel pairs	recover, $p_{\mathrm{T}} > 0.6$ GeV
MixedTriplet	pixel+strip triplets	displaced, $R^{\mathrm{max}} = 7$ cm
PixelLess	strip triplets/pairs	displaced, $R^{\mathrm{max}} = 25$ cm
TobTec	strip triplets/pairs	displaced, $R^{\mathrm{max}} = 60$ cm
JetCoreRegional	pixel+strip pairs	$p_{\mathrm{T}} > 10$ GeV, inside high p_{T} jets
MuonSeededInOut	muon-tagged tracks	muons
MuonSeededOutIn	muon detectors	muons

superimposed pile-up collisions. The efficiency grows for values of transverse momentum between 0.1 GeV and 1 GeV. In this region, the fake rate decreases, as tracks start satisfying the seeding criteria. A plateau of efficiency 83% is reached for values of p_{T} below 100 GeV. For greater values of transverse momentum the efficiency drops, due mainly to high-momentum jets. These boosted jets are very collimated and hits from different tracks in the jet can merge and may be associated to a single track. Efficiency is limited by the silicon detector pitch. Cluster splitting techniques are applied to disentangle hits from overlapping particles [127].



(a) Efficiency vs p_T (b) Efficiency vs η

Figure 3.8: Offline track reconstruction efficiency during 2016 for simulated $t\bar{t}$ events with 50 superimposed pile-up collisions. Only tracks with simulated $p_T > 0.9$ GeV are used.



(a) Fake rate vs p_T (b) Fake rate vs η

Figure 3.9: Offline track reconstruction fake rate using the experimental setup of 2016 for simulated $t\bar{t}$ events with 50 superimposed pile-up collisions. Reconstructed tracks with reconstructed $p_T > 0.9$ GeV are considered.

Iterative Tracking at the High-Level Trigger during 2016

Section 2.2.5 described the HLT as a time-constrained computing system, with a maximum average latency of $t_{th} = 220$ ms. This implies that the average processing time for one event cannot exceed t_{th}, otherwise the L1 trigger would create back-pressure and part of the incoming events would be lost.

Sequences of algorithms, *paths*, that require a greater amount of time to complete are executed less often and quota is applied on the fraction of events that can run a specific task in the unit time. The result is a time distribution for events, which peaks below 20 ms, with a tail that can exceed 3 s. The execution time does not increase linearly with the luminosity, as events become more complex, hence increasing the number of possible combinations that the track reconstruction needs to follow (Figure 3.10). An example of the overall HLT time distribution is shown in Figure 3.11.

The three iterations that compose the online iterative tracking are described in Table 3.2. All these iterations follow a fast tracking step, called *Pixel Tracks*, that uses only the information coming from the Pixel Detector for quickly reconstructing tracks and primary vertices. The Pixel Tracks step evaluates prompt triplets ($R^{\max} = 2$ mm and $z^{\max} = 24$ cm), having transverse momentum greater than $p_{\mathrm{T}}^{\min} = 0.9$ GeV.

The execution time of this algorithm grows with the number of possible combinations of three hits satisfying seeding criteria. At an instantaneous luminosity of $1.5 \cdot 10^{34}$ cm^{-2} s^{-1}, with a pile-up of about 50, evaluating PixelTracks for every event would require an increase of the time threshold t_{th} of about 100 ms.
The following techniques can be implemented in order to mitigate the execution time requirements:

- Use a regional track reconstruction, where tracking is done only within regions-of-interest. These regions are defined by a $\eta - \phi$ cone in the direction of an already available Physics Object (e.g. muons, electrons, jet candidate seeded by calorimeters or muon chambers);

- Increase the threshold on the minimum p_{T} at the seeding step to values larger than 1 GeV to reduce the number of seeds produced;

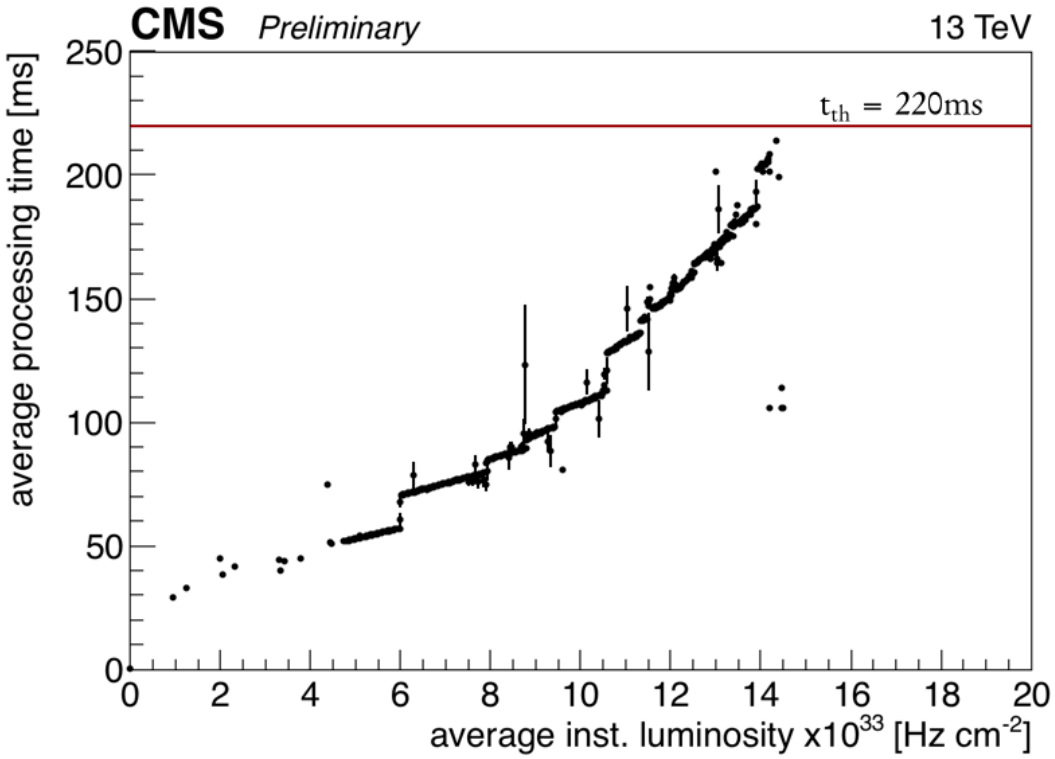


Figure 3.10: Processing time distribution of the HLT menu as function of the non-calibrated instantaneous luminosity. The timing was measured using events from fill 5393 [128].

- Select only the track phase-space in which tracks mostly come from the primary interaction region;
- Sacrifice resolution on primary vertices and run a proto-tracking using only clusters' positions. Clusters are projected in the direction of a jet candidate onto the beam-spot and their frequency is added in a histogram. If a bin of the histogram exceeds a threshold, then the corresponding z-position is used as primary vertex position [129].

The performance of the online iterative tracking is shown in Figures 3.12 and 3.13. The *iter0* iteration makes use of the reconstructed Pixel Tracks as seeds. The seeding criteria are tighter with respect to those used offline, due to the limited amount of time. For this reason, the efficiency starts to grow for values of transverse momentum $p_{\mathrm{T}} > 0.3$ GeV, reaching 50% around $p_{\mathrm{T}} = 1$ GeV. Hits used to build tracks in *iter0*, are masked and a low momentum iteration, *iter1* runs on the remaining hits. This iteration allows to

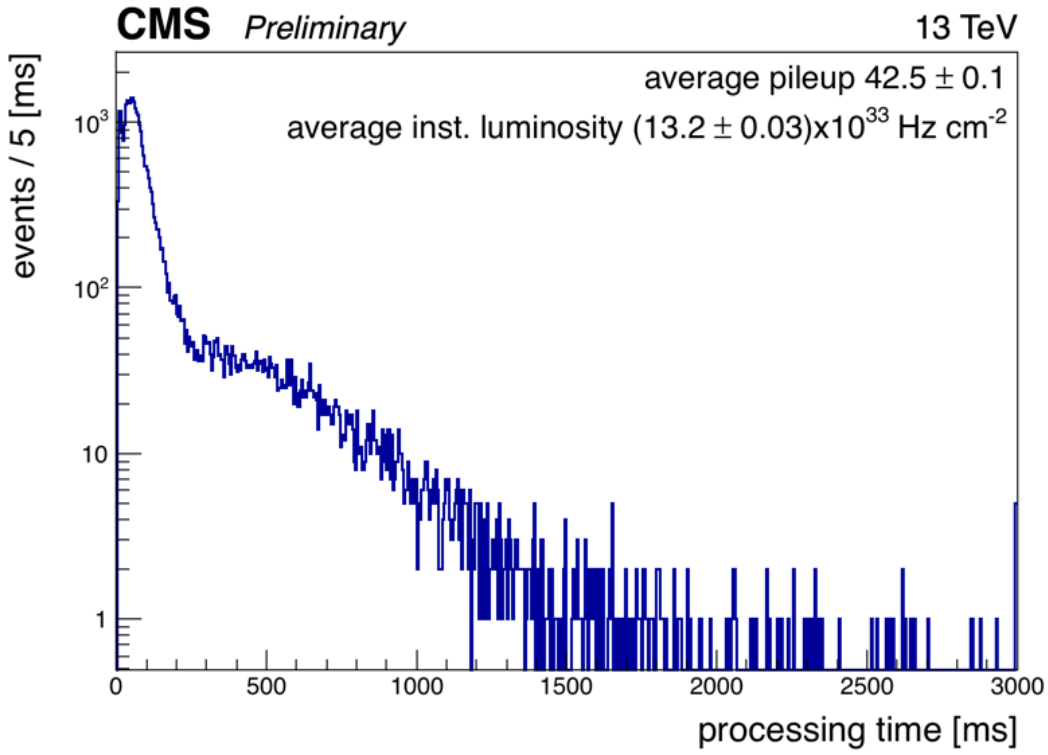
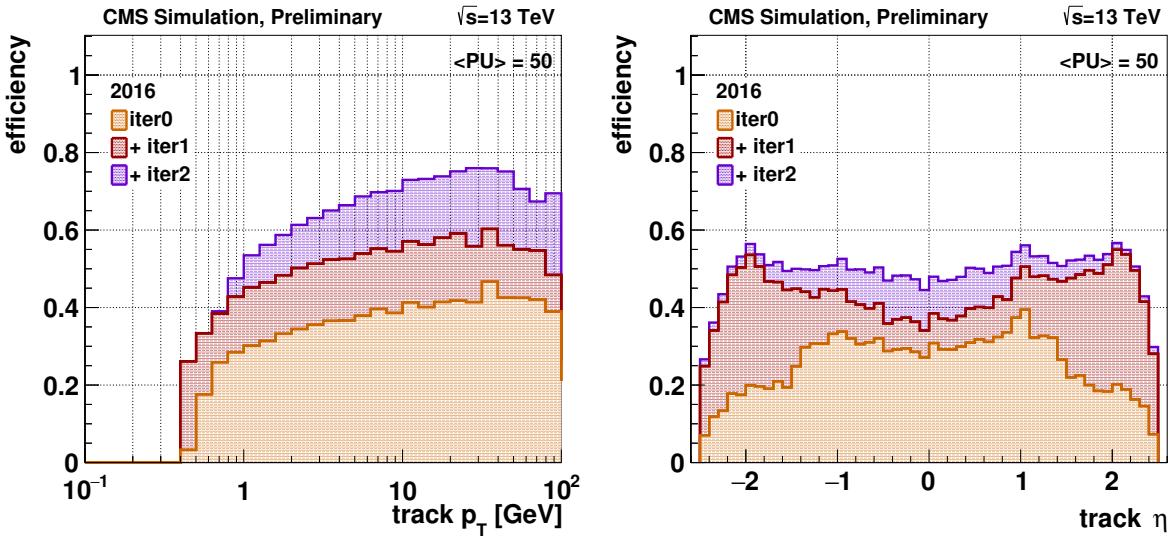


Figure 3.11: Processing time distribution of one of the HLT menus used during the 2016 data taking operations. The timing was measured using events from run 279975, with an average instantaneous luminosity of $1.32 \cdot 10^{34}$ cm^{-2} s^{-1} corresponding to a pile-up of 42.5 [128].

recover efficiency for values of $|\eta| > 1$. Tracks with a missing pixel hit are recovered in *iter2* by loosening the requirement on the number of hits to two pixel hits. The limited number of iterations and the absence of iterations targeting displaced tracks keeps the efficiency from reaching 80% efficiency.

Table 3.2: Online Tracking iterations used during 2016 data taking.

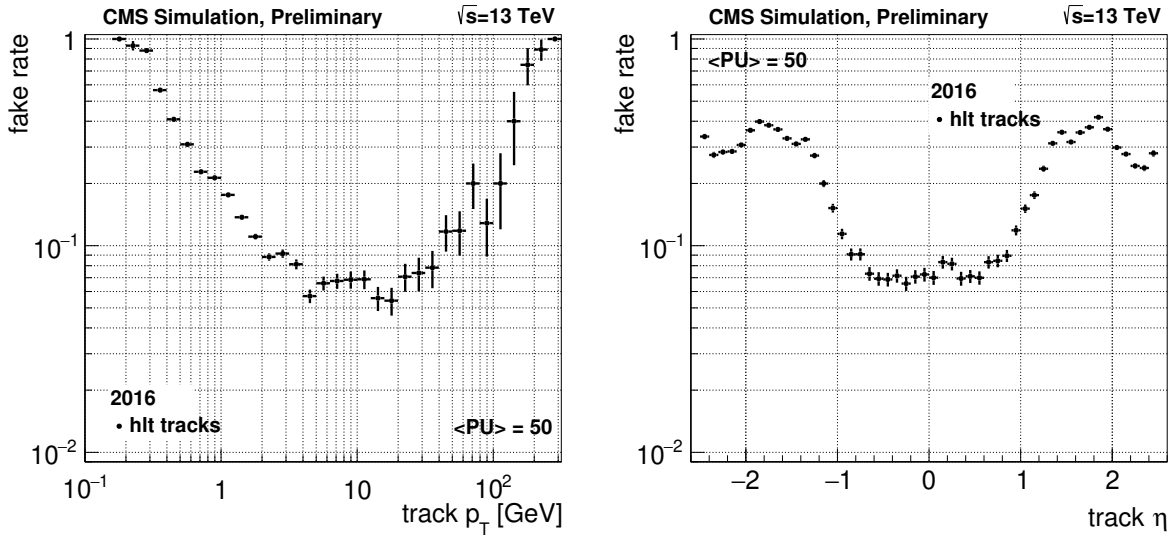
Name	Seeds	Target Tracks
iter0	Pixel Triplets	prompt, $p_T > 0.9$ GeV
iter1	Pixel Triplets	prompt, $p_T > 0.5$ GeV
iter2	Pixel Pairs	recover, $p_T > 1.2$ GeV



(a) Efficiency vs p_T

(b) Efficiency vs η

Figure 3.12: Online track reconstruction efficiency during 2016 for simulated $t\bar{t}$ events with 50 superimposed pile-up collisions.



(a) Fake rate vs p_T

(b) Fake rate vs η

Figure 3.13: Online track reconstruction fake rate during 2016 for simulated $t\bar{t}$ events with 50 superimposed pile-up collisions.

3.3 Pixel Tracking during CMS Phase-1

The already complex online and offline track reconstruction will have to deal not only with a higher occupancy environment but also with data coming from a more complex detector, described in Section 2.5. The new offline pixel-only iterations and the evolution of the online track reconstruction to adapt to the upgraded Pixel Detector are shown in Table 3.3. Quadruplets iterations replace those based on triplets. Iterations based on pairs disappear to keep the fake-rate under control in high pile-up events. Track seeding combinatorial complexity could easily be dominated by track density and become the bottleneck of the High-Level Trigger and offline reconstruction execution times (Figure 3.14). The CMS HLT farm and its offline computing infrastructure rely on technologies that have evolved extremely rapidly but that cannot rely anymore on an exponential growth of frequency guaranteed by the manufacturers (Section 4.2).

Innovative hardware and algorithmic solutions have been studied in the context of this thesis work. These solutions are going to be described in the following chapters.

Table 3.3: Evolution of Offline Pixel tracking iterations and Online track reconstruction to adapt to the Phase-1 Pixel Detector.

Name	Phase-0 Seeds	Phase-1 Seeds
Offline Pixel Track reconstruction		
InitialStep	Pixel Triplets	Pixel Quadruplets
LowPtQuad	N/A	Pixel Quadruplets
HighPtTriplet	N/A	Pixel Triplets
LowPtTriplet	Pixel Triplets	Pixel Triplets
PixelPair	Pixel Pairs	N/A
DetachedQuad	N/A	Pixel Quadruplets
DetachedTriplet	Pixel Triplets	Pixel Triplets
Online Pixel Track reconstruction		
Pixel Tracks	Pixel Triplets	Pixel Quadruplets
iter0	Pixel Tracks	Pixel Tracks
iter1	Pixel Triplets	Pixel Quadruplets
iter2	Pixel Pairs	Pixel Triplets

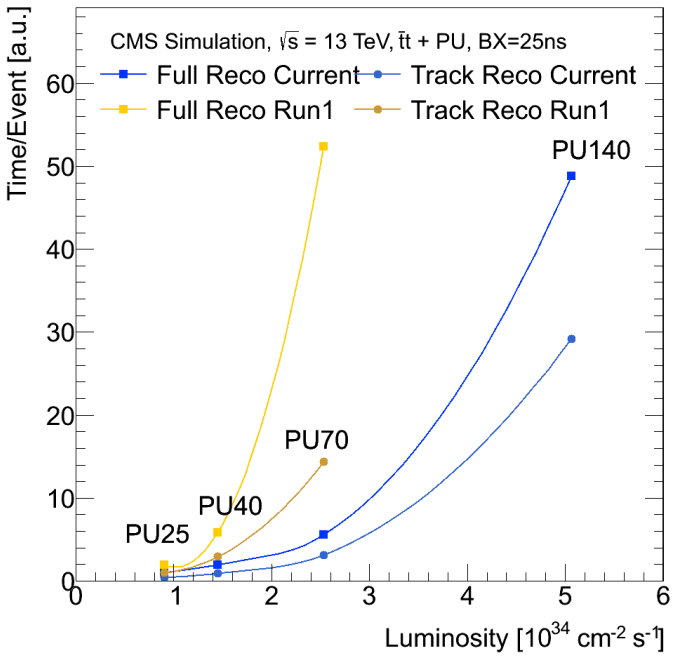


Figure 3.14: Total reconstruction time and track reconstruction time for Run 1 (orange) and for Run 2 (blue) with respect to instantaneous luminosity. Tracking execution time does not grow linearly because of increased event complexity and it is a major contribution to the entire event reconstruction time [130].

Chapter 4

Heterogeneous parallel computing

4.1 Introduction

In the next years CMS will collect a growing amount of data due to scheduled increase in instantaneous luminosity at the LHC.
The CMS online and offline computing infrastructure used to rely on the industry to deliver an exponential increase in processor performance per unit cost over time. An application could have been written using simple models and achieve performance improvements as new generations of processors came out. However, this trend stopped for technology problems connected to the power consumption, hence changing the evolution of processors. Today, scaling performance with processors' generations can be achieved only via an application-level parallelism.
CMS started a dedicated R&D program because failing to adapt would imply an exponential increase in costs for computing. The aim of this program is the identification of new technologies and software design techniques needed to cope with the increasing event complexity with an almost flat budget.

This chapter will describe trends in technology and the differences between serial and parallel computing. It will also describe a particular kind of accelerators, GPUs, in the past dedicated to rendering of 3D graphics. Today, it is possible to exploit massively parallel architecture of GPUs for *General Purpose GPU* (GPGPU) computing.

4.2 Serial and Parallel computing

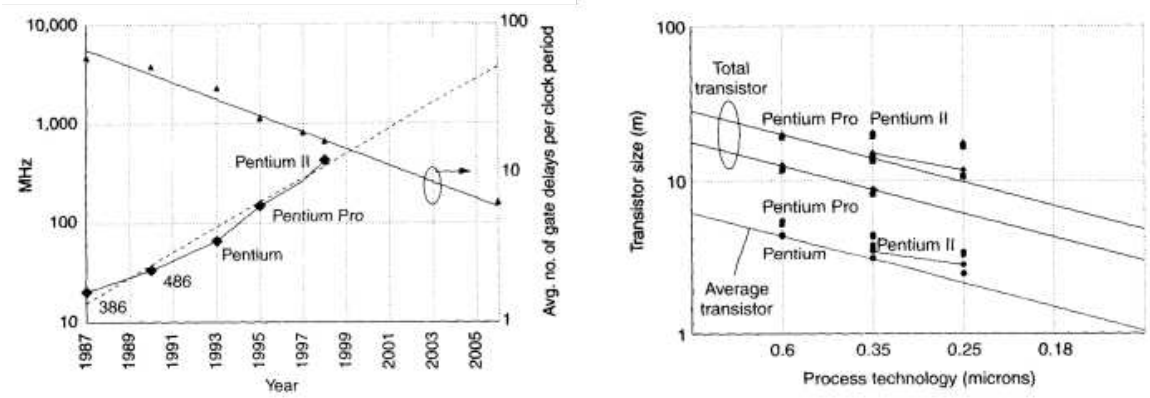


Figure 4.1: Scaling of Intel transistors' size and frequency vs time before hitting the Power Wall [131].

In 1965, Gordon Moore, one of the co-founders of Intel, affirmed that the performance of CPUs would have doubled every 12 months along with the number of their transistors [132]. This would have been called *Moore's Law* a few years later. Moore's Law was elaborated in its final form, stating that microprocessor performance doubles every 18 months (Figure 4.2). The main contribution to the gain in microprocessor performance at this stage came by increasing the clock frequency. Applications' performance doubled every 18 months without having to redesign the software or changing the source code.
The power dissipated by a processor scales as $P = QCV^2f + VI_{\text{leakage}}$, where Q is the number of transistors, C is the capacity, V the voltage across the gate and f the clock speed of the chip and I_{leakage} is the leakage current [134] [135]. To keep on the Moore's Law curve, the size of transistors had to be halved every 18 months [131]. However, in the early 2000s, the layer of silicon dioxide insulating the transistor's gate from the channels through which current flows was just five atoms thick and could not be shrunk anymore. This show-stopping problem is known as the *Power Wall* [136]: processors with higher frequencies also dissipate more power.
The transistors count kept growing exponentially by increasing the overall number of cores per chip (Figure 4.4). The *concurrency revolution* started.

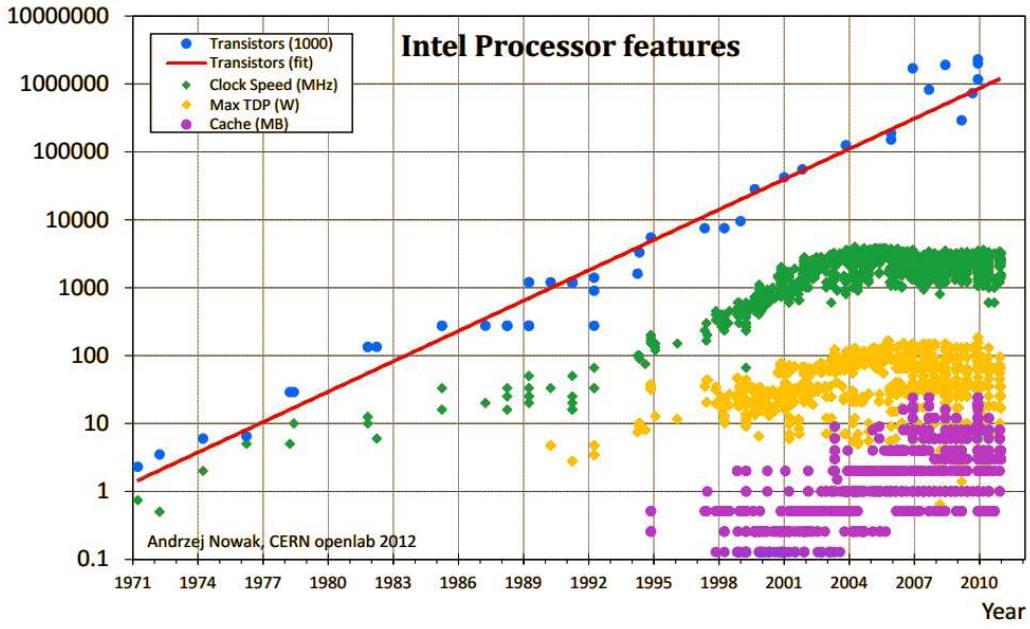


Figure 4.2: Moore's Law. After 2004 the power consumption (TDP) and the frequency remained uniform due to the Power Wall. Mainly because of heat dissipation issues, savings generated by improved process technologies do not translate to major frequency increases, but are typically re-invested in other areas. It should also be noted that technologies auxiliary to the CPU such as cache memory — a low-latency memory located on-chip — do not progress or scale at the same speed as raw compute power [133].

The basic operation that every Processing Unit (PU) has to process is called *instruction*. Whenever a PU is asked to perform an instruction, the address in memory containing the instruction is saved. A *Program Counter* holds the address of the next instruction. In general, an instruction can be modeled as a sequence of three operations:

- *fetch*: the content of the memory stored at the address pointed by the Program Counter is loaded in a *Instruction Register* and the Program Counter is increased to point to the next instruction's address.
- *decode*: the content of the Instruction Register is interpreted to determine the actions that need to be performed
- *execute*: an *Arithmetic Logic Unit* performs the decoded actions.

In serial computation, one PU executes the instructions in sequential order — one after

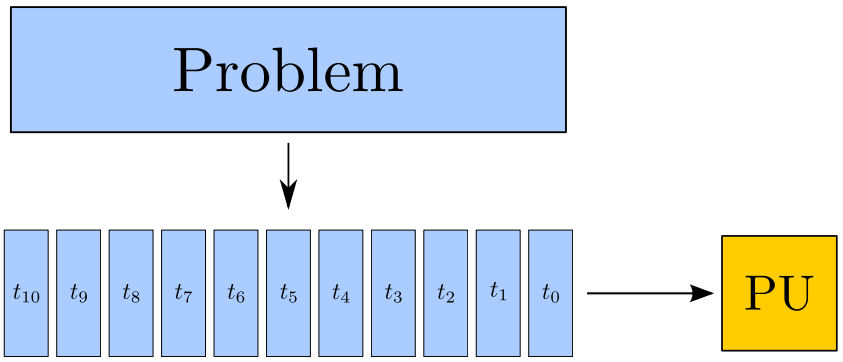


Figure 4.3: Serial computing: Every chunk of the problem is executed *serially* by a single *Processing Unit* (PU).

the other — and every instruction has to wait for the previous one to be completed before its execution can start (Figure 4.3).
In parallel computation, if two instructions have no data dependency, they can be executed in *parallel*, at the same time, by two PUs (Figure 4.4).

As the number of cores increases, problems of memory bus contention start to arise, which go under the name of *Memory Wall*. There are different techniques that can be used to mitigate the memory wall:

- Reuse data and instructions: data and instructions which are used often are stored in a on-chip memory called cache.
- Increase the memory transfer speed: this can be done by increasing frequency, which is limited by the power wall.
- Increase the amount of data to transfer: memory transfers have overheads, which can become negligible if more memory is transferred in one instruction.
- Improve the access pattern to memory: if more processing units are reading adjacent memory locations, they can all be fed by a single memory transfer.

4.2.1 Amdahl's Law

One could hope of getting a boost in performance by a factor equal to the number of processing units (CPUs). However, this is limited by the fraction of the code which cannot run in parallel, due to initialization, I/O or synchronization. Given a program,

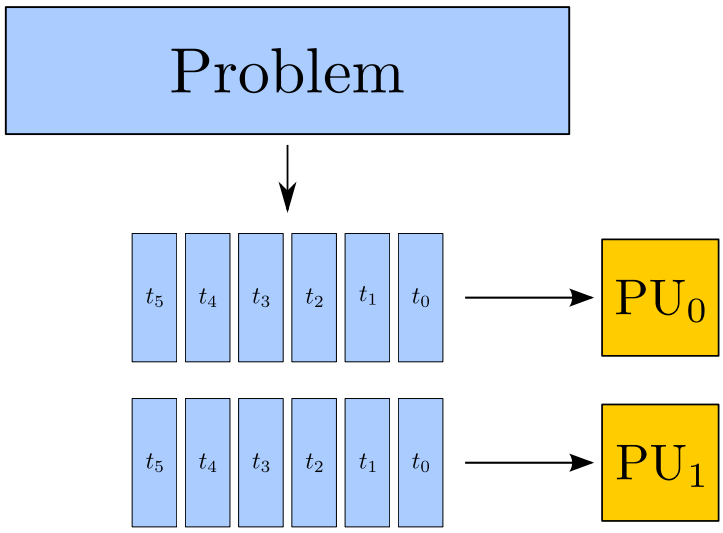


Figure 4.4: A problem that can be divided in independent chunks is solved by more than one Processing Unit *concurrently*.

Amdahl's law indicates the maximum speed up that can be achieved for a variable number of PUs [137].
Let T_s be the execution time of a serial program and T_p its execution time after parallelization when using a number p of processing units. The speedup obtained is then given as the ratio:

$$S(p) = \frac{T_s}{T_p}. \tag{4.1}$$

If we consider the fraction f of the program that has to run serially, and the time t obtained when executing the parallel section serially, the parallel execution time T_p will then be given by:

$$T_p = fT_s + (1-f)T_p = fT_s + \frac{(1-f)t}{p}. \tag{4.2}$$

The speed up for a program, with a serial fraction f, that is running on p PUs is given by:

$$S(p,f) = \frac{T_s}{fT_s + \frac{(1-f)t}{p}} \rightarrow \frac{1}{f} = S_{max}(f) \tag{4.3}$$

The trend of Amdahl's Law for different values of f and for a variable number of Processing Units is shown in Figure 4.5. If the parallel part of the program is 95% of the whole program, the maximum speedup one can obtain is $S_{max} = \frac{1}{1-0.95} = 20$.

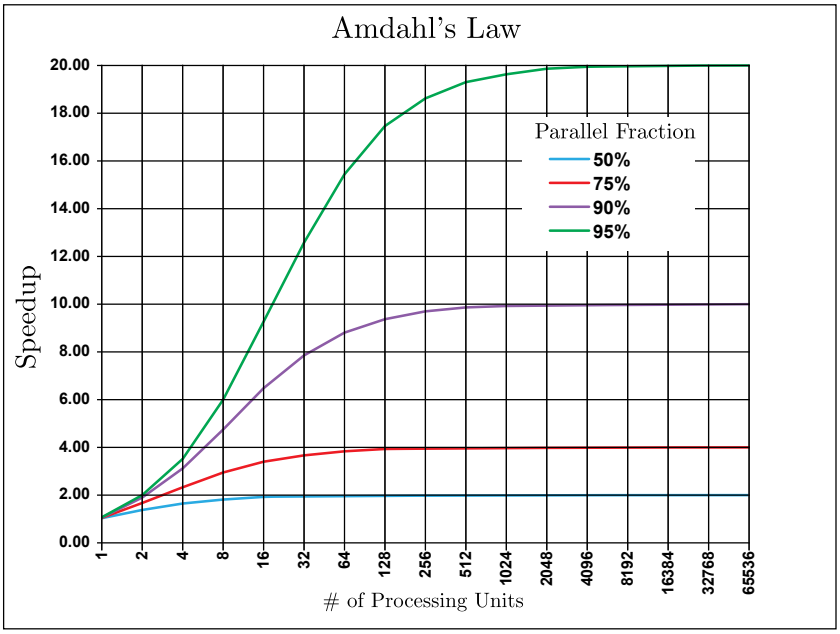


Figure 4.5: Amdahl's law for different fractions of parallel code [138]

.

4.2.2 Gustafson's Law

Amdahl's Law assumes that a problem can be split in a number of independent chunks n that can be processed in parallel and that this number is fixed. Often, increasing the size of the problem does not change the time spent executing the sequential part, and only affects the parallel portion. This is due to the fact that the number of independent chunks n increases and so do the opportunities of parallelization. Under this assumption, let $f(n)$ be the sequential code fraction of the program. Then, this fraction $f(n)$ decreases to zero as the size of the problem n approaches infinity.
The speed up is given by *Gustafson's Law* [139]:

$$S(n) = f(n) + p[1 - f(n)]. \tag{4.4}$$

The maximum speedup is then given by:

$$S_{max} \equiv \lim_{n \to \infty} S(n) = p. \tag{4.5}$$

Hence, one can increase the size of a problem and still solve it in the same amount of time by making more parallel equipment available.

4.3 Computer Architectures

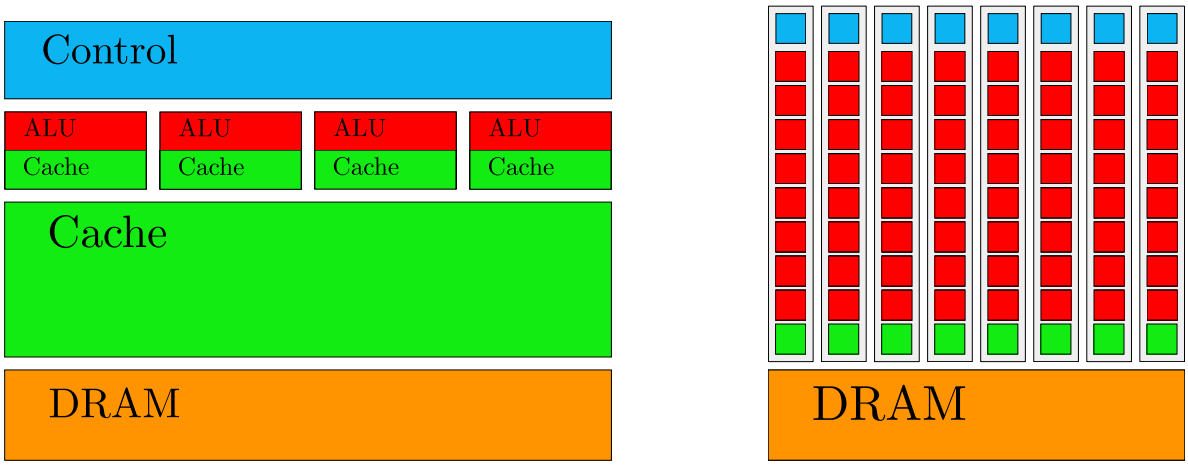
Today the computing industry facing the following challenges:

- Reducing power consumption.
- Reducing cooling costs.
- Improving performance.
- Managing ever-expanding volumes of data.

Industry is evolving to develop or employ heterogeneous parallel computing systems, i.e. systems that use a variety of different types of specialized computational units. Instead of increasing the number of the same kind of processing units, industry has opened a new line of development of parallel architectures such as multi-core CPUs and *accelerators*, that incorporate highly specialized processing capabilities to handle particular tasks. These heterogeneous platforms can achieve higher energy efficiency and improved performance by assigning the best architecture for a particular task. There exist accelerators dedicated to random number generation, compression and decompression, encryption and decryption, matching of regular expressions, decoding of video and audio streams [140].

Central Processing Units (CPUs) comprise a relative small amount of cores per chip (around 10), supporting a large instruction set. Instructions are not necessarily executed in the same order they are issued in the program. This level of complexity is hidden from the programmer thanks to the *instruction-set architecture* (ISA) and its hardware implementation.
Microprocessors use traditional sequential-programming models and their design principles pursue the goal of reducing instruction execution latency.

There are several important design choices for today's CPUs (Figure 4.6a):



(a) CPU (b) GPU

Figure 4.6: Simplified architectures of a traditional CPU and a GPU. GPUs tend to allocate more transistors to execution units, whereas in CPUs more space resources are dedicated to caches and control units.

- Large Arithmetic Logic Units (ALUs) can deliver the results of complex calculation in a small amount of clock cycles.
- Caches are small memories with very low access latency that are usually found on chip. The use of large caches can reduce slow memory access latency to memory for repeatedly requested data elements.
- Large control units implement complex control mechanism like *branch prediction*, which is used for anticipating the direction of a conditional statement in the code based on its previous history. Control units allow multiple instructions from a single thread to be executed at the same time or even out of their sequential order.

It is challenging for CPU designers to increase bandwidth to and from memory because CPUs have to satisfy requirements from legacy operating systems and applications.

Graphic Processing Units (GPUs) are an example of accelerators. The design of GPUs has been shaped in the years by the video game industry. GPUs have vector processing capabilities that enable them to perform parallel operations on very large sets of data and to do it with much lower power consumption relative to the serial processing of similar data sets on CPUs (Figure 4.6b). Their *throughput-oriented* design aims to maximize the throughput of instructions by executing as many of them at the same time as possible.

This is achieved by assigning way more transistors to execution and less to caches, control, sacrificing features like branch prediction.
In the High-Performance Computing sector, there are two major manufacturers of GPUs: AMD and NVIDIA. GPU parallelism, on Graphic cards produced by NVIDIA, is exposed for general-purpose computing thanks to an architecture called *Compute Unified Device Architecture* (CUDA) [141]. On GPUs produced by AMD, parallelism is exposed by OpenCL [142].
These GPUs are connected to the CPU via a high speed bus called PCI Express that in its generation 3.0 can achieve a memory transfer speed of 985 MB/s per lane and usually a motherboard can have up to 16 lanes per PCI Express slot.
In the following, only solutions based on NVIDIA architecture have been employed because they can be programmed at a lower level allowing better control of the hardware through the software. GPU designers can more easily achieve higher memory bandwidth with respect to CPUs. This is due to the simpler memory models and fewer legacy constraints. In fact, the more recent NVIDIA GPU architecture, GP100, can achieve a bandwidth of 732 GB/s with respect to few tens of GB/s that a CPU can achieve [143].

4.4 Parallel and heterogeneous computing in High Energy Physics

The simulation and the reconstruction of events in High Energy Physics are blessed with natural event-level parallelism. The Power Wall and the consequent stop in the increase of the CPUs' clock frequency happened during the period of preparation to the imminent data taking. Chip manufacturers could only provide growing performance by increasing the number of cores and experimental software frameworks had little time to adapt. The strategy that has been pursued consists in executing as many different instances of an application as the number of CPU cores, i.e. *multi-processing*.
Many times multi-processing implies that n independent copies of an application are executed by different cores. In order to be able to support an application running in multiprocessing, a machine needs an amount of memory which is at least n times the memory needed to run a single instance of the application.

Table 4.1: Speed and memory savings of multi-threaded task relative to single-threaded [144].

Configuration	Speed-up		Memory	
	Low PU	High PU	Low PU	High PU
RECO	1.05	1.04	4.6	3.7
RECO and Monitoring	0.83	0.94	3.3	2.7

In order to reduce the memory footprint, utilize idle CPU cores and improve hardware utilization, one needs to introduce thread-parallelism within the process, hence making the application *multi-threaded*. Some of the benefits of the transition from multi-processing to multi-threading in CMSSW are shown in Table 4.1. Two datasets were used, each containing $t\bar{t}$ events with 4 (low PU) and 40 (high PU) superimposed pile-up collisions. Two configurations were executed: one with reconstruction only, the other adding monitoring. The single-threaded performance was measured by running 64 processes, one per core on a 64-cores machine, while the multi-threaded performance was measured by executing 8 jobs with 8 threads each on the same machine. Both for low and high pile-up datasets, if running the reconstruction only, a slight speed-up of the multi-threaded application is measured with respect to the single-threaded one. This is motivated by the better utilization of the memory hierarchy and of the common resources. In this configuration, the multi-threaded application also achieves a factor 4 smaller memory footprint with respect to the single-threaded one.
After including monitoring, both the speed-up and the memory saving are reduced. This is due to the fact that monitoring contains many synchronization points the application runs sequentially most of the time. The drop in speed-up is smaller for high pile-up, as the fraction of time spent in reconstruction increases, mainly due to the track reconstruction as shown in Figure 3.14 [144].

There are four main ecosystems in which parallel and heterogeneous computing can be employed today:

- Trigger.
- Reconstruction.
- Simulation.
- Data analysis.

Trigger

As described in Section 2.2.5, the amount of raw data produced by HEP experiments is too high to be entirely saved and analyzed offline. A trigger system is used to record only those events that contain potentially interesting physics. Event filtering has been traditionally done in two levels:

- The Low-Level Trigger reduces the event rate of tens of MHz by three orders of magnitude and the processing time has a maximum latency which is of the order of ten microseconds.

- The High-Level Trigger reduces the event rate to the order of some kHz. The maximum latency of the order of a hundred milliseconds is more relaxed, if compared to the Low-Level trigger.

Traditionally, for their hard real-time requirements, Low-Level Triggers have been implemented with programmable electronics performing a coarse local reconstruction. Many FPGAs can be employed for solving small local problems in a deterministic number of clock cycles. However, if an experiment can afford to buffer multiple events and run the same algorithms on more events at the same time in parallel, event throughput can be sustained using software and accelerators. This allows to use complex and flexible algorithms and improve the physics potential of an experiment. Many experiments, like NA62 [145] and ALICE [146] [147] at CERN, are moving along this direction by using GPUs for accelerating the reconstruction of physics objects coming from one sub-detector. Other experiments are developing trigger-less solutions, in which an almost offline-like reconstruction is executed online. This is the case of LHCb that, during the shutdown between 2018 and 2019, will upgrade its data acquisition system in order to provide trigger-less readout system and the full software trigger [148].
The High-Level trigger farms are isolated and controlled environments. This feature makes the High-Level Trigger a fertile ground for software optimization and adoption of new technologies in order to retain better physics signal in an environment with latency constrains.

Simulation

A sustained effort in the optimization of simulation software is required in order to cope with the steadily increasing need for computing resources. This is required in order to take advantage of accelerators and modern CPU architectures. For this reason, CERN concluded in 2012 that Geant4 had to be redesigned. A parallel simulation framework, GeantV [149], is being developed at CERN with the long term objective of achieving an order of magnitude faster performance with respect to Geant4.
In order to maximize the exploitation of vector units and threads coming with modern processors, particles are collected in contiguous memory regions called *baskets*. Simulation of transport is applied to particles in the same basket by a worker thread. In order to maximize the occupancy of these baskets, particles belonging to different events can be collected into the same basket.
Different baskets are assigned to different worker threads, while accelerators can be assigned more than one basket.

Reconstruction

Reconstruction frameworks used by the LHC experiments have been developed in C++ [150] with portability and functionality in mind. At the time of the initial design of the software, optimization for the hardware using high-level languages was not possible yet. As the CPUs clock frequency stopped growing and parallelism became the best way of improving the performance of the software, event reconstruction did not keep the industry's pace (Section 4.2).
There are various reasons why HEP reconstruction was not able to benefit from modern platforms, which are connected to data-locality and fetching of instructions. By not using small kernels, opportunities for vectorization and parallelization of an algorithm become unachievable. In fact, long functions mixing control with computation become a problem of control-flow rather than data-flow. Control-flow is not something modern CPUs excel at, and accelerators even less. Furthermore, data is sparse in memory making inefficient the use of memory caches. In such a paradigm, by relying just on the number of cores for scaling, the usage of multiple cores by multi-processing seems to be more beneficial than exploiting the same number of cores for parallelization of the algorithms. In 2012, CERN openlab estimated that HEP reconstruction can utilize efficiently only less than

10% of what industry and High-Performance Computing communities can achieve [133]. Two general design criteria are being employed for designing new software:

- A good memory access pattern locates data which are frequently accessed (temporal data locality) or adjacent data elements accessed linearly (spatial data locality) close to each other, in order to improve the efficiency of caches in the memory hierarchy.
- Compute-intensive loops should be contained in small branch-free kernels, instead of long functions mixing computation and control.

Parallelization at algorithm level becomes beneficial in case algorithms are redesigned with parallelism and efficient memory access pattern in mind.

Data analysis

At the LHC experiments, data analysis software is developed individually by groups, many times on top of the common framework called ROOT [151]. The variety of the performance of analysis software is very spread. Sometimes, although opportunities for parallelism within algorithms exist, physicists prefer to rely only on processing different chunks of data on different cores or different machines [152]. Sometimes, time-to-solution is critical and packages like GooFit [153] are deemed reliable for parallelizing maximum likelihood fit on GPUs.

One clear direction for the next decade computing lies in the need to exploit multiple hardware architectures, in order to be able to utilize opportunistic resources of heterogeneous nature for the offline computing and data analysis, as well as many-core co-processors (e.g. GPUs) at the Trigger stage.

Chapter 5

Development of parallel track seeding algorithms and data structures

The possibility of offloading the complex and intrinsically parallel-friendly problem of the CMS track seeding to GPUs has been explored and described in details in the following sections.

Developing parallel software adds a level of complexity with respect to sequential applications:

- Parallelism has to be found and many times algorithms have to be redesigned.
- A software can occupy a high number of different states: different control flows, data structures and memory management. The creation of additional threads means not only a multiplication of the possible states by the number of threads, but also possible interactions between threads should be taken into account. This makes parallel programs much harder to design, debug and verify.
- Parallel programs have non-deterministic execution. In general, the scheduling order of parallel executions is not managed by the programmer. Furthermore, as the arithmetic between floating point operands is not necessarily associative, the result of operations depends on their order [154].
- Enforcing an order often implies synchronization, resources contention and serialization.

This section describes the design of parallel algorithms aimed at executing the entire HLT Pixel Tracks reconstruction from RAW data to tracks onto GPUs. A requirement for all the algorithms described in this section is that the results produced are equivalent to those obtained with a serial version of the algorithm executed on CPUs. The input data to be sent to the GPU is the RAW data as it is read from the detector.
The following steps have to be performed on the GPU:

- Digitization of RAW data for the simulation of the detector's electronics response.
- Clustering of adjacent firing pixels: *clusters* are formed.
- Estimation of cluster parameters: *hits* are created carrying the information about their position and errors in the detector.
- Pair generation using FKDTree, which will be described in Section 5.2: given two layers with their hits, for each hit on the outer layer, the pair generator searches for hits on the inner layer that are compatible with the beam-spot hypothesis and a momentum range.
- A *Hit-Chain Maker* based on the concepts of *Cellular Automata* will be described in Section 5.3: given all the doublets generated in the pair generation step, it looks for chains of four compatible hits.
- Final Fit: Found quadruplets are fitted with a helix and the final pixel tracks parameters are found.

By introducing massive parallelism within the event, hence offloading the combinatorics to an architecture that can execute thousands of threads in parallel, the effects of pile-up would be mitigated.

5.1 CUDA threading model and memory hierarchy

A *thread* is a flow of control that can execute in parallel with others in the same process. All the threads within a program can share the same address space, heap memory, signals but not their private data, registers and stack. The coordination of a large number of threads is of paramount importance in order to fully harness the potential of a GPU.
The design and flow of a parallel program usually follows several steps:

- Initialization: localized data structures and communication channels are established.
- Identification: each thread acquires a unique identifier, typically ranging from 0 to $N-1$, where N is the number of threads.
- Data decomposition: decompose global data into chunks and localize them.
- Mapping: chunks of data are distributed and associated with threads that run the kernel computation.
- Finalization: acquire results from threads hence reconciling the global data structure.

The CUDA architecture comprises a scalable array of multi-core Streaming Multiprocessors (SMs) which are designed to execute hundreds of threads concurrently. To maximize throughput, the GPU can switch context between threads very efficiently and instructions are heavily interlaced. In CUDA a *grid* of parallel threads executes the same parallel kernel. Threads in the grid are grouped in work-groups called *blocks* (Figure 5.1). Threads within the same block can cooperate, communicate and synchronize. When a kernel is launched, each thread acquires three indices `x,y,z` that uniquely identify it within the same block. Blocks inside a grid can be identified by three indices `x,y,z` as well. The possibility of creating a three-dimensional grid of three-dimensional blocks of threads is just a logical representation for easing the mapping of threads to the particular data structure used in a problem. For example, one could use an index variable like `blockIdx.x * blockDim.x + threadIdx.x` to uniquely map threads in a grid to a linear data structure. These identifiers allow all the threads to distinguish among themselves and to compute memory addresses and make control decisions.
When a CUDA program on the host CPU invokes a kernel grid, the blocks of the grid are numbered and distributed to multiprocessors with available execution capacity. The threads of a thread block execute concurrently on one multiprocessor, and multiple thread blocks can execute concurrently on one multiprocessor. As thread blocks terminate, new blocks are launched on the free multiprocessors.

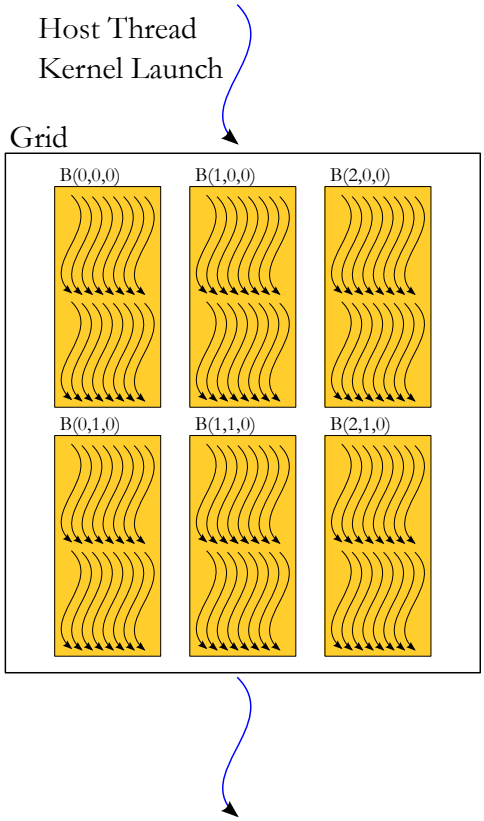


Figure 5.1: Representation of the CUDA execution model. A single host thread is launching a kernel which spawns a grid of blocks. Blocks can be arranged along three dimensions —x, y and z —, and so do threads inside blocks. In the example shown in figure a grid contains 6 blocks arranged in a configuration $(3, 2, 1)$. Each block contains 16 threads arranged in a configuration $(8, 2, 1)$.

5.2 FKDTree

In tracking or clustering in HEP, it is often the case that an algorithm needs to filter a large collection of hits, for certain conditions. For example, for calorimeter clustering, for each hit, algorithms have to search for other hits that lay within a certain distance around it. This is known as the *post office problem* [154], as it is very similar to the problem of assigning residences to a post office based on a properly defined distance. Another example is the doublet generation at the beginning of track seeding shown in Figure 5.2: consider two coaxial cylindrical layers in a solenoidal magnetic field. The seed generator is asked to produce pairs of hits that are compatible with the beam-spot width and length (in green), and p_{T}^{min} seeding criteria (Section 3.2.4). For each hit on

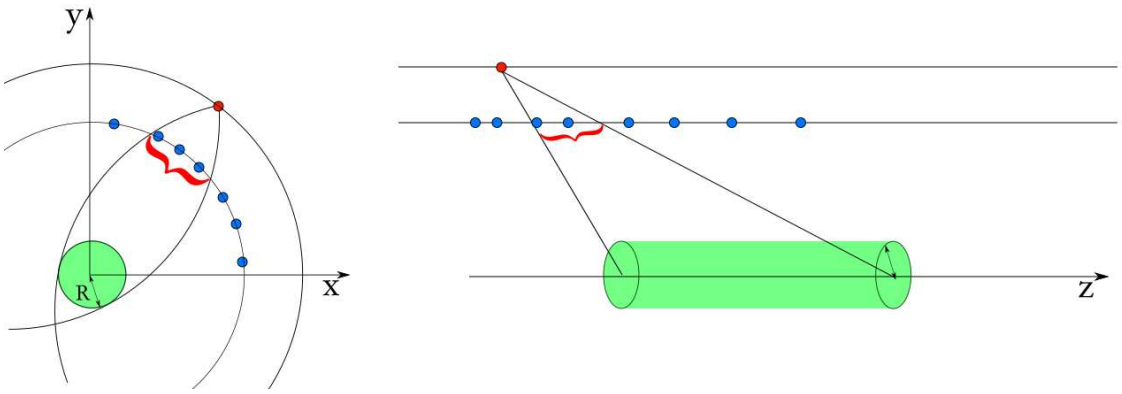


Figure 5.2: A hit on an outer layer creates a search window in $\eta - \phi$. The size of the window depends on the seeding criteria described in Section 3.2.4.

the outer layer, a $\eta - \phi$ two-dimensional interval is created on the inner layer. Hits on the inner layer are then filtered in the interval. This section describes the development of a data structure, called FKDTree, needed to search for hits in a k-dimensional interval in $\mathcal{O}(\log N)$ optimal complexity[1] and that can also be accessed efficiently in parallel. If the dataset is not sorted at all, the complexity is $\mathcal{O}(kN^2)$, because for every hit the algorithm executes k comparisons on all the other $N - 1$ hits. If the dataset is sorted along one dimension, the search complexity is $\mathcal{O}(kN \log N)$, because k comparisons are executed, but on one dimension the dataset can be traversed like a binary tree.

5.2.1 Original implementation of a k-d tree

A k-d tree is a space partitioning tree data structure used for storing multidimensional points in a way which is efficient to traverse. Given a set of N points $\{p_0, p_1, ..., p_{N-1}\}$, each having a number k of dimensions, the original implementation of a k-d tree present in CMSSW can be described as follows:

1. Elements are ordered along the first dimension d_0.
2. The median element is picked and elements are partitioned in two sub-spaces.
3. Recursively run from the beginning separately in the two sub-spaces on the next dimensions d_1, then d_2 and so on.

[1]The Big-$\mathcal{O}$ notation is used to indicate the dominant term, without coefficients, contributing to the execution time as the size N of the data structure grows to infinite.

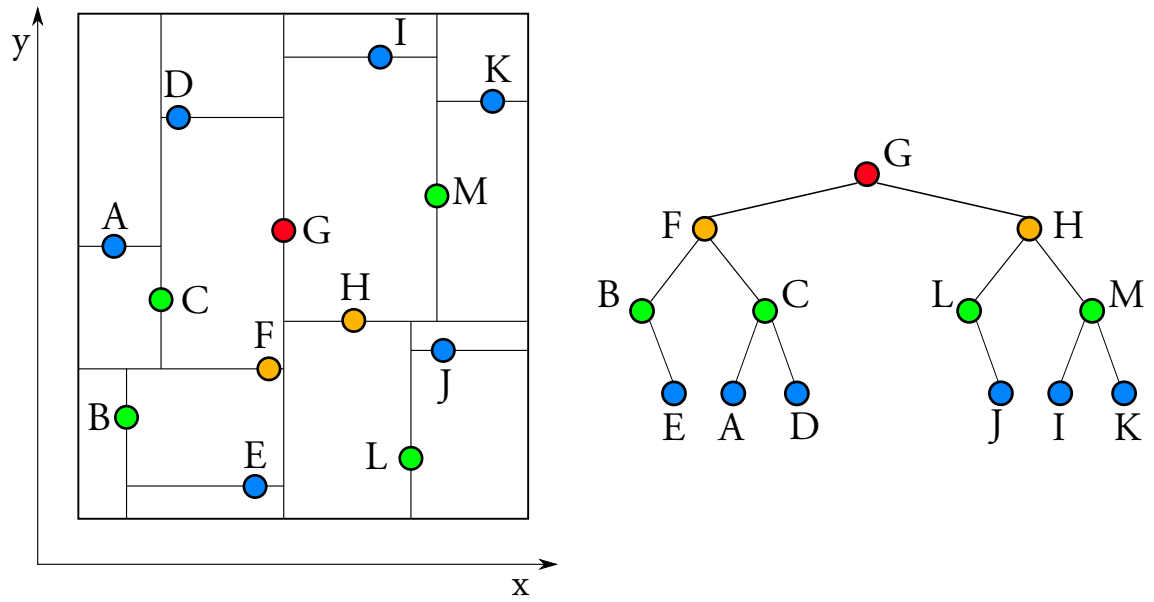


Figure 5.3: Original CMSSW implementation of a k-d tree for a set of two-dimensional points.

4. If the last dimension d_{k-1} is reached, the building process restarts from d_0.

Figure 5.3 shows a k-d tree built for a set of 13 points distributed on a plane (x, y): the point G is the median along the x-axis and it is identified as the root of the tree. The node G splits the space into two sub-spaces. All the nodes whose x coordinate is less than G will make the left sub-tree. The right sub-tree will comprise the other nodes. To identify G's children nodes, the median node along the y coordinate is chosen. This leads to the choice of nodes F and H, which in turn divide their sub-spaces into two sub-spaces. The loop over the two dimensions is now over and it starts again from the first, i.e. the x coordinate.

This implementation has the following properties:

- it allows for fast and efficient spatial queries of geometry data, e.g. a request for all points that lie on a specific axis aligned box.
  Searching for points within a box have a complexity which is $\mathcal{O}(\log N)$ in the best case, i.e. only one point falls within the box and $\mathcal{O}(N)$ in the worst case with all points located inside the box.

- No node carries the information of where it is located in the tree.

- The tree is always balanced[2].
- Nodes are stored in scattered memory locations:
  - Copying nodes to the GPU memory would require issuing many small copies and many small allocations.
  - Threads prefer a coalesced access to adjacent memory locations for maximizing throughput. This will not be possible in a scattered linked-tree implementation.
- A visit of the parent node is required for accessing a node.
- The memory needed for storing the k-d tree is given by $N \times$ sizeof(Node), which must include the memory needed for storing pointers to the left and right children.

The implementation described above has to be improved in order to make its porting to GPUs possible and advantageous. This improved implementation, developed completely in the context of the work described in this dissertation, goes under the name of *FKDTree* [155], and it was developed in CUDA, OpenCL, TBB and OpenMP.

5.2.2 Building

The first and most important improvement consists in changing the building process. In the FKDTree building process, the pivotal element does not need to be necessarily the median. The pivot can be the element that makes the k-d tree *complete* [154], i.e.:

- All its levels before the last are filled completely.
- The last level is filled from left to right.

The only information required to select this pivotal element is just the number of elements contained in the sub-space partition (Listing 5.1). Figure 5.4 shows a FKDTree built from the same points distribution shown in Figure 5.3.

Another important observation comes from the fact that, for selecting the i^{th} element among n elements, sorting all the n elements completely along one dimension is not required. In fact, in the following iteration, elements will be sorted along another

[2] A tree is balanced if, for every node, the number of nodes in its left sub-tree and in its right sub-tree differs at most by one.

Listing 5.1: The selection of the pivotal element that makes the sub-tree complete.

```
unsigned int partition_complete_kdtree(unsigned int length)
{
  if (length == 1)  return 0;
  unsigned int index = 1 << (FLOOR_LOG2(length));
  if ((index / 2) - 1 <= length - index)
    return index - 1;
  else
    return length - index / 2;
}
```

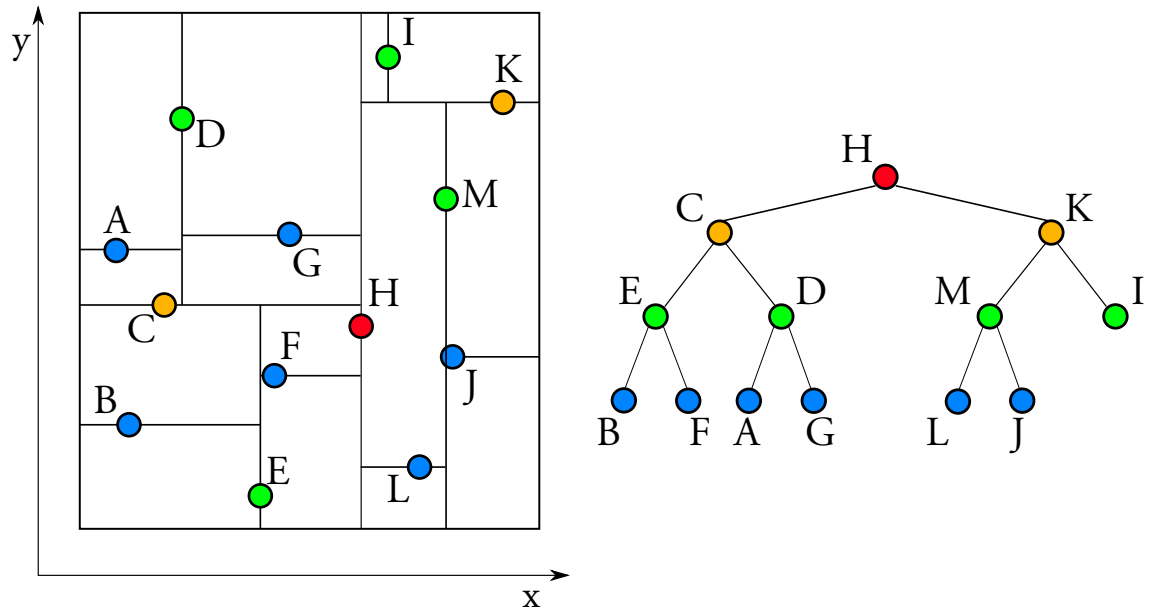


Figure 5.4: FKDTree built out of points distributed on a plane.

dimension, making no use of the previous order. Furthermore, sorting is a quite expensive operation. It has best-case complexity $\mathcal{O}(N \cdot \log N)$ [154]. It is not necessary to sort the elements completely, as it is sufficient to select all the elements smaller than the i^{th} element and place them, unordered, to its left, and those greater to its right. This selection has a time complexity of $\mathcal{O}(N)$ and it is achieved like follows (Figure 5.5):

- Choose a random pivotal element x.
- Create three partitions L for elements smaller, E for elements equal and G for elements greater than x.
- If the i^{th} element is contained in E stop the process, otherwise repeat recursively

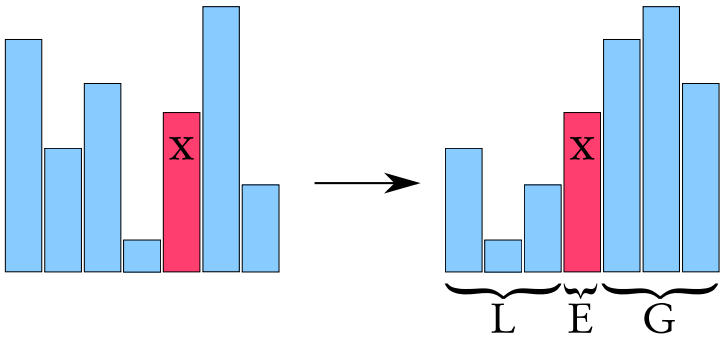


Figure 5.5: Elements in a partition of the space in a k-d tree don't need to be ordered. In the example in figure, the fourth element (indicated by x), can be found by using the quick-select algorithm, which partially sorts the array: the elements smaller than x make the L set, the elements greater than x are used to make the G set. The L and G sets don't need further sorting.

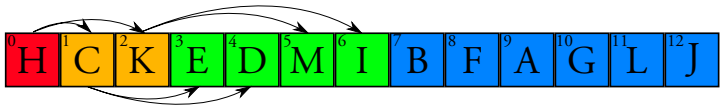


Figure 5.6: Array representation of a FKDtree. The information about the element's position in the tree can be deducted from the value of its index. The array is completely filled: there are no holes to be masked.

only in the partition.

Now that the tree is complete (Figure 5.4) it is possible to use the array representation based on integral indices:

- the left child of a parent, which is located at position j, is the node that is found in position $2j + 2$ in the array.
- The right child of the parent is at position $2j + 3$ in the array.
- To find the parent of any element in the tree, the integer division can be used. Given an element at the position m, its parent can be found at position floor($\frac{m-1}{2}$).

Figure 5.6 shows the array representation of a FKDTree. The array representation allows an efficient traversal of the FKDTree using only simple and fast mathematical operations. The array representation would have been possible even without building a complete tree. However, elements would have been scattered in memory requiring a mask of non-valid memory locations.

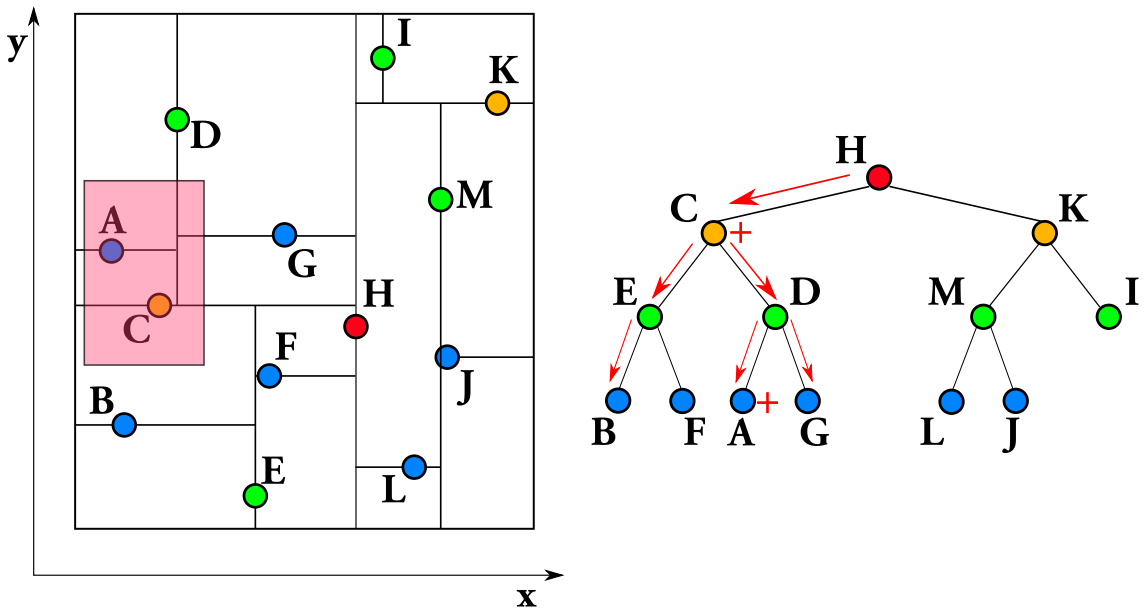


Figure 5.7: Searching for hits within a 2-dimensional box with a FKDTree.

5.2.3 Searching

In CMS event reconstruction, tree building would be executed a small amount of times per event.
On the other hand, it is not unusual to execute searching on the same pool of hits at least once for every hit in the event.

Given a node of the tree and its splitting hyper-plane of the sub-space, the search algorithm traverses either one or both the children. The decision of which child to traverse depends on the position of the box with respect to the splitting hyper-plane (Figure 5.7):

- if the box intersects the splitting hyper-plane, the node coordinates are checked for compatibility with the box and both its children will be visited. In case the node falls within the box boundaries, the node is added to the result.
- If the box is entirely located in one of the two sub-spaces defined by the splitting hyper-plane, then only the child node situated in that sub-space will be visited.

Hence, a precise search using the FKDtree always reaches the leaves of the tree. However, it could be possible to stop the search at a certain depth if one requires approximate

results.

Parallel implementation

In the case of the pair generation in the CMS track seeding or in the case of the nearest-neighbors search for clustering, parallelism comes naturally when the algorithm performs a search query on the same tree for every hit. Parallelism can be achieved by mapping one search query to one CUDA thread. Therefore, multiple queries can be executed in parallel.
The first obvious implementation of the search algorithm would be recursive. However, when conditional statements have different behavior for different threads, a parallel implementation of a recursive search algorithm would certainly suffer from the presence of control branches.
The new building implementation comes to aid: by making the FKDTree complete, it is possible to evaluate the depth of the tree by just knowing the number of elements: depth = floor($\log_2(N)$).

Consider the code in Listing 5.2:

- lines 3-4: A box is represented by two points `FKDPoints`: `minPoint` represents the corner of the box with minimum coordinates, while `maxPoint` points to the corner with maximum coordinates. The vector `foundPoints` will store the result.

- lines 8-9: `indicesToVisit` is a queue. It is basically a ring buffer, in adjacent memory locations, that supports insertion and extraction of elements at both its ends. The first element to be pushed into the queue is the root node, which index is 0. This queue is required in order to make the traversal of the tree branch-less: unlike the recursive implementation that would have required visiting a new node immediately, using a queue allows the storage of indices of the elements to be visited at the end of the processing of one level of the tree. This kind of tree traversal is also known in graph theory as *Breadth-first search* (BFS) [154].

- line 16:the depth of the tree is known at building time. This allows conversion of the recursive approach to an iterative one, processing one level of the tree at a time in a for-loop.

Listing 5.2: Branchless search using BFS.

```
theDepth = FLOOR_LOG2(nPoints);

search(const FKDPoint minPoint,
     const FKDPoint maxPoint, vector foundPoints)
{
// a queue containing the ids of the elements to visit at each level of the tree
// a queue is a FIFO supporting push_back and pop_front operations
  queue indicesToVisit;
  indicesToVisit.push_back(0);
  int index;
  bool intersection;
  int dimension, maxNumberOfSonsToVisitNext, numberOfSonsToVisitNext;
  int numberOfindicesToVisitThisDepth;
  int firstSonToVisitNext;
// iteratively loop over the levels of the tree.
  for (int depth = 0; depth < theDepth + 1; ++depth)
  {
    dimension = depth % numberOfDimensions;
    numberOfindicesToVisitThisDepth = indicesToVisit.size();
    for (int visitedindicesThisDepth = 0;
        visitedindicesThisDepth < numberOfindicesToVisitThisDepth;
        visitedindicesThisDepth++)
    {
      index = indicesToVisit[visitedindicesThisDepth];
      intersection = intersects(index, minPoint, maxPoint, dimension);
      firstSonToVisitNext = leftSonIndex(index);
// if the point intersects the box at this coordinate check if it is in the box,
// otherwise save both its children (if available) for the next tree level
      if (intersection)
      {
        if (is_in_the_box(index, minPoint, maxPoint))
        {
          foundPoints.emplace_back(theIds[index]);
        }
        numberOfSonsToVisitNext = (firstSonToVisitNext < theNumberOfPoints)
                                + ((firstSonToVisitNext + 1) < theNumberOfPoints);
      }
      else
      {
// decide which son to visit at the next level
        firstSonToVisitNext += (theDimensions[dimension][index]
                                < minPoint[dimension]);
        numberOfSonsToVisitNext = min((firstSonToVisitNext < theNumberOfPoints)
                                      + ((firstSonToVisitNext + 1) < ↩
                                          theNumberOfPoints), 1);
      }
// push the marked children in the queue of elements to visit at the next level
      for (int whichSon = 0; whichSon < numberOfSonsToVisitNext; ++whichSon)
      {
        indicesToVisit.push_back(firstSonToVisitNext + whichSon);
      }
    }
// extract elements to be check for compatibility with the search box
    indicesToVisit.pop_front(numberOfindicesToVisitThisDepth);
  }
}
```

- line 18: the dimension that will be considered at this iteration cycles through the possible dimensions.
- lines 20-26: the loop over the indices in the queue starts here. For each element, its position relatively to the box is pre-computed.
- lines 27-37: if the box intersects the splitting hyper-plane, the maximum number of children will be pushed into the queue. If the node is contained in the box, its index is stored in the result.
- lines 40-45: if there is no intersection, only one of the children node will be pushed into the queue.
- lines 47-49: the marked indices are pushed into the queue.
- line 53: the processing of one element has completed and the element is removed from the queue.

By using a BFS and processing one level of the tree at a time, one ensures that multiple threads are working together, not waiting for each other's branches to complete.

The implementation is not specific to the number of dimensions, allowing for the analysis of any k-dimensional point-cloud: for mitigating the ever-increasing pile-up in CMS tracking and clustering one could create search queries that comprise not only the traditional three spatial dimensions but also information like:

- The cluster shape for locating the vertex, or for having an initial identification of the particle: tracks that are more inclined with respect to a layer produce clusters whose shape is stretched (Figure 3.2).
- Timing information for reducing both the in-time and the out-of-time pile-up.

5.2.4 Tests and results

Sometimes there are not enough resources to execute a parallel algorithm with more than one CPU thread. Even a good parallel algorithm could achieve slightly worse performance with respect to a good sequential implementation. This is caused by overhead needed to make data structures thread-safe and for parallel book-keeping. For this reason, the

Table 5.1: Comparison between different implementations of the search algorithm running on the same CPU Intel Core i7-4771 running with a clock frequency of 3.50 GHz. The search has been used in a nearest-neighbor algorithm: the algorithm searches for neighboring hits in a volume around each of the 50,000 three-dimensional points.

Algorithm	Time	speed-up wrt naive search
naive search	23795 ms	1×
recursive FKDTree	145 ms	164×
BFS FKDTree (1 thread)	112 ms	212×
BFS FKDTree (4 threads)	30 ms	793×

sequential execution of FKDTree needs to be compared to the performance achieved by other good sequential-only implementations.

Table 5.1 shows the result of the comparison for 50,000 points when performing a nearest-neighbors search. The sequential BFS implementation in FKDTree does not only show no regression, but it's actually 30% faster than the recursive one, due to the smaller number of branches.
Note also that the parallel version launched using 4 threads does not guarantee a perfect factor 4 improvement in computational performance, due to Amdahl's law, as described in Section 4.2.1.

Now that it has been proven that there is an advantage even when running sequentially by adapting the algorithm for parallel computation, a measurement of how well the parallel implementation performs on different architectures has been carried out.
The result of these measurements is summarized in Table 5.2: for measuring the performance of the search algorithm on CPUs, the Intel TBB implementation has been used assigning on search query to a different CPU thread. For GPUs instead, the CUDA implementation was used, mapping one CUDA thread to a different search query. In this benchmark, the measured throughput is the average number of searches in the unit time. The throughput was measured by executing the same set of search queries on different architectures, and the results were verified to be the same. Parallel implementations running on traditional CPU architectures can achieve at most half of the throughput achieved by GPUs. However, the performance of CPUs and GPUs should not be considered as exclusive. The real benefit of heterogeneous systems is that one or more GPUs can execute high-throughput algorithms while the CPU can run something else or execute

Table 5.2: Comparison between different architectures when running FKDTree BFS search algorithm.

Architecture	Frequency	Physical cores	Threads	searches/ms
CPU: Intel i7 6700K	4.00 GHz	1×4	8	1820
CPU: IBM Power8 Minsky	2.9 GHz	2×10	160	2820
CPU: Intel KNL 7210	1.3 GHz	1×64	256	1360
GPU: NVIDIA K40	745 MHz	2880	—	4400
GPU: NVIDIA P100	1.3 GHz	3584	—	9910

the same workload. In the case of FKDTree, the best performance can be achieved by combining together on the same machine one or more IBM Power8 Minsky CPUs and up to eight NVIDIA P100 GPUs.

5.3 Hit-Chain Maker using a Cellular Automaton

In this section, a solution to the problem of creating seeds with three or four hits, namely *triplets* and *quadruplets*, in the upgraded Phase-1 pixel detector is discussed, which includes four barrel layers and three end-cap disks per direction. This geometry provides a four-hits coverage against the three-hits coverage of the 2016 Pixel Detector (Section 2.5). In the event reconstruction software, a list of seeding layers is passed via a configuration file. For the geometry shown in Figure 5.8, the layer list for the HLT Pixel Tracks step would be:

1. BPix1+BPix2+BPix3+BPix4
2. BPix1+BPix2+BPix3+FPix1$^+$
3. BPix1+BPix2+FPix1$^+$+FPix2$^+$
4. BPix1+FPix1$^+$+FPix2$^+$+FPix3$^+$
5. BPix1+BPix2+BPix3+FPix1$^-$
6. BPix1+BPix2+FPix1$^-$+FPix2$^-$
7. BPix1+FPix1$^-$+FPix2$^-$+FPix3$^-$

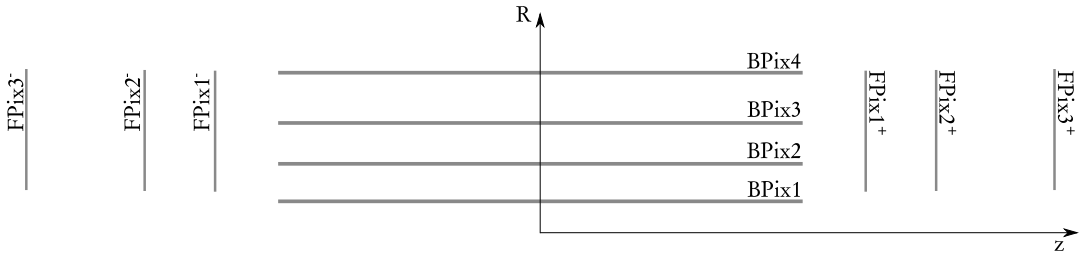


Figure 5.8: Simplified Phase-1 Pixel detector geometry with layer names.

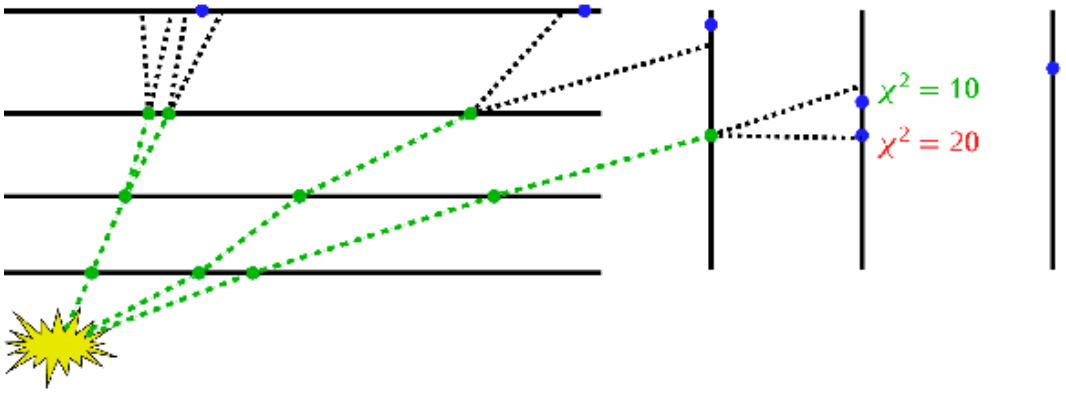


Figure 5.9: The Triplet Propagation algorithm is the natural continuation of the seeding approach used up to 2016: propagate a triplet found with 2016 seeding to the fourth layer and search for compatible hits.

where B is used to indicate a Barrel layer and F to indicate a forward one.

Triplet Propagation

The classical approach of track seeding in the Phase-1 detector would be to extend the existing algorithms, that have been used until 2016, to propagate the triplets to a fourth layer, hence creating quadruplets. However, this approach is not suitable for parallel architectures and it is therefore used for comparison only. This method is called *Triplet Propagation* (Figure 5.9).

The Triplet Propagation algorithm can be summarized as follows:

- One combination of four layers is picked from the layer list (e.g. combination number 1).

- Pairs of hits compatible with the region configuration are created between the first two layers in the combination (e.g. between BPix1 and BPix2).
- Every created pair propagates to the third layer (e.g. BPix3), creating a two-dimensional box where hits are searched. These hits, together with the already formed hit doublet, form triplets.
- Every created triplet propagates to the fourth layer (e.g. BPix4), creating a two-dimensional box where hits are searched. For each triplet, only the hit on the fourth layer that produces the best fit is used for forming a quadruplet.

The entire process is repeated for every combination in the layer list.
There are some disadvantages with this approach:

- As the 2016 triplet seeding is a subset of the triplet propagation approach, the latter will certainly be slower.
- Propagating a triplet to form a quadruplet does not make the algorithm easier to parallelize, as it adds another iteration that has dependencies on the completing previous one. Coarse-grained parallelism is still possible by associating one thread to a triplet, but it will not be enough to exploit a massively parallel architecture like a GPU, for which a fine-grained parallelism is desirable (Gustafson's law described in Section 4.2).
- Doublets and triplets from the same sub-set of layers in different combinations are evaluated multiple times. As an example: triplets from BPix1+BPix2+BPix3 are calculated in combinations 1, 2 and 5.

An implementation of quadruplet seeding that takes longer CPU time, would have to be executed on a smaller number of events at the HLT due to limited computing resources. This would lead to degraded physics performance. Furthermore, in offline reconstruction this will create a higher demand of resources which is not sustainable.

5.3.1 Cellular Automata

The upgrade of the pixel detector is a good opportunity to restructure the mathematical formulation of the problem and to introduce innovations at the algorithmic level.

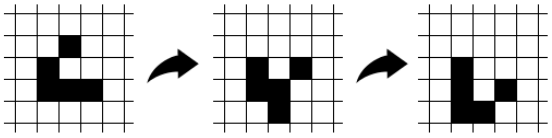


Figure 5.10: Conway's Game of Life in consecutive steps of its evolution [157].

The class of mathematical models that worked as inspiration for a local and parallelizable algorithm used to restructure the CMS track seeding goes under the name of *Cellular Automata* (CA) [156]. One example of a Cellular Automaton is the Conway's *Game of Life* [157]. It is a game without any players, and its evolution is determined solely by its initial state (Figure 5.10):

- A grid of square cells represents the world. Every cell has eight *neighbors*, its adjacent cells.
- Each cell can be found in two states: dead or alive.
- The state of each cell evolves in discrete time intervals, called *generations*. There can be four different states:
  1. Under-population: a living cell dies if less than two neighbors are alive.
  2. Overcrowding: a living cell dies if more than three neighbors are alive.
  3. Stability: a living cell survives if two or three neighbors are alive.
  4. Expansion: a dead cell becomes a living cell if exactly three neighbors are alive.
- It is important to update the state of all the cells simultaneously going from one generation to the next one.

What makes a Cellular Automaton easy to parallelize is the fact that a cell does not need to have any knowledge of the world outside the neighboring cells. The interaction between a cell and its neighbors is based on simple and local calculations.

5.3.2 Graph of Seeding Layers

The Hit-Chain Maker based on the CA could have reused the infrastructure of the Triplet Propagation, running once for each combination of four layers. This would have caused the repetition of the evaluation of some parts of the algorithm, such as the pair generation or the neighbor finding between cells belonging to adjacent layers. It is possible to avoid all repetitions by parsing the entire layer list with all the desired combinations and evaluate hit pairs exactly once.
Two graphs are created: one containing layer pairs (GLP), and one containing layers (GL). When parsing a new combination of four layers, the algorithm proceeds as follows:

- The presence of each layer in GL is checked. If the layer is already contained in GL, its index is considered, otherwise a new index is created.
- If this layer is not the first one in the combination of seeding layers, a layer pair formed using the indices of this layer and the preceding one is considered. If this layer pair is not contained in GLP, pairs of hits compatible with the seeding region are evaluated and the layer pair is added to GLP.
- A layer which is the first one in the combination of the seeding layers is called *Root layer*.

Connections between nodes of the GL and GLP graphs are created as follows:

- A layer is the *outer layer* of a layer pair, if it appears later in the layer combination.
- A layer is the *inner layer* of a layer pair, if it appears before in the layer combination.
- A layer pair is the *outer layer pair* of a layer, if the layer is one of the layer pair's inner layers.
- A layer pair is the *inner layer pair* of a layer, if the layer is one of the layer pair's outer layers.

As an example, connections and nodes of the two graphs for the specific case of HLT Pixel Tracks are shown in Figure 5.11.

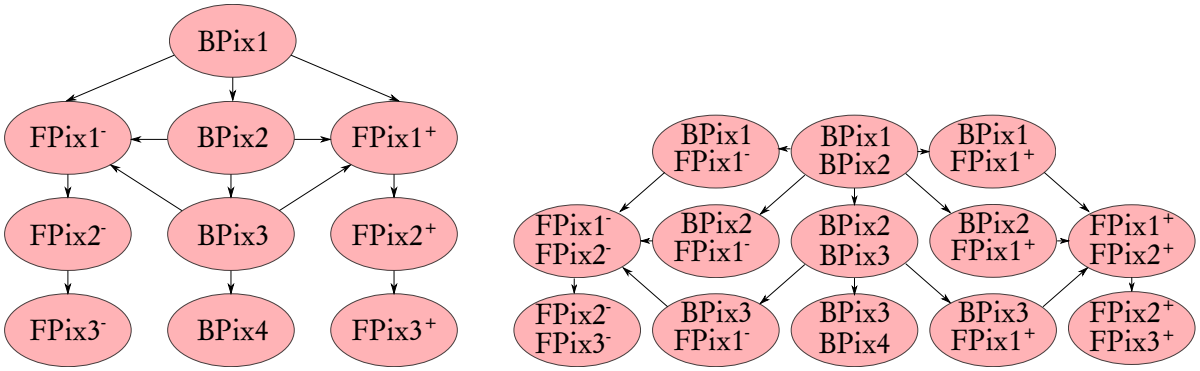


Figure 5.11: CAGraph for layers and layer pairs.

Implementation

First, an instance of a container class called `CAGraph` is created (Listing 5.3). `CAGraph` contains all the unique layers and the unique pairings of layers found in the configuration. By looping over all the combinations and through all the layers in each combination, parsing the whole layer list proceeds like follows:

- if a layer has never been found it is pushed to the list of layers in `CAGraph.theLayers`;
- if the layer is the first one in the combination and has never been found as a root layer, its index is pushed into `CAGraph.theRootLayers`;
- whenever a new `CALayer` is created, the indices of adjacent layers and layer pairs are stored in its containers `theInnerLayers`, `theOuterLayers`, `theInnerLayerPairs` and `theOuterLayerPairs`.
- if two consecutive layers in a combination have never been found together, they form a new `CALayerPair` and they are pushed into `CAGraph.theLayerPairs`;

5.3.3 Cell Creation

A *Cell* is the building block of the Hit-Chain Maker using the CA. It can be constructed starting from two hits that are compatible with the track seeding criteria. Two cells are said to be *neighbors* if:

- They have one hit in common.

Listing 5.3: Simplified code of the `CAGraph` container. `vector` and `concurrent_vector` are containers developed in the context of this work.

```
1  struct CALayer
2  {
3    vector<int> theOuterLayerPairs;
4    vector<int> theInnerLayerPairs;
5    vector<int> theOuterLayers;
6    vector<int> theInnerLayers;
7    vector< concurrent_vector<CACell*> >  isOuterHitOfCell;
8  };
9
10 struct CALayerPair
11 {
12   int theLayers[2];
13   vector<CACell> theFoundCells;
14 };
15
16 struct CAGraph
17 {
18   vector<CALayer> theLayers;
19   vector<CALayerPair> theLayerPairs;
20   vector<int> theRootLayers;
21 };
```

- They are compatible.

Compatibility between two cells will be discussed in Section 5.3.4. In addition to the information of the two hits, it also contains:

- the state of the Cell, `CAState`, which is initially set to 0;
- the pointers to the inner and outer neighboring Cells.

The state of the cells and their neighbors will be used when the CA evolves. The creation of a cell from a doublet is independent from the creation of another cell and allows to map each cell to the hits that it connects.
For reducing the complexity of later parts of the algorithm, it is a useful choice to keep track of the opposite connection as well, i.e. from hits to Cells.

Implementation

For this reason a container `isOuterHitOfCell` is introduced(line 7 of Listing 5.3). This container stores the pointers to all the cells that have a hit as an outer hit in the doublet. Filling this container is not straightforward in a concurrent environment. In fact, two

Listing 5.4: A thread-safe `push_back` method in CUDA.

```
1  __device__
2  int push_back_ts(const T& element)
3  {
4    auto previousSize = atomicAdd(&m_size, 1);
5    if(previousSize<maxSize)
6    {
7      m_data[previousSize] = element;
8      return previousSize;
9    }
10   else
11   {
12     atomicSub(&m_size, 1);
13     return -1;
14   }
15 };
```

different cells can share the same outer hit and a *race condition* could arise. A race condition could occur when two threads access a shared variable at the same time and they are not synchronized. In order to avoid that, the `isOuterHitOfCell` data structure must become *thread-safe*, concurrent-friendly. Thread-safety is achieved by means of atomic functions, which are capable of reading, modifying, and writing a value back to memory without the interference of any other threads, hence guaranteeing that a race condition will not occur. Listing 5.4 shows how the `push_back` method becomes thread-safe:

- line 1: `__device__` is a CUDA function attribute. It indicates that a function will be called by the GPU code to be executed on the GPU itself;
- line 4: the size of the container is increased atomically by 1. However, the size is checked to be smaller than the container maximum size, before any push operation can be done;
- line 7: the element is pushed into the container;
- lines 10-13: The insertion fails in the case the insertion goes beyond the allocated size for the container. In this case, the size is reset back to its original value.

All the data structures involved are now thread-safe and the creation kernel can be safely parallelized.
In the HLT Pixel Tracks step, there are 13 unique pairs of layers (on the right in Figure 5.11). These layer pairs can be treated independently in parallel. A two dimensional

Listing 5.5: A simplified version of the creation kernel.

```
1  __global__
2  void kernel_create(const GPULayerDoublets* gpuDoublets,
3      const GPULayerHits* gpuHitsOnLayers, CACell** cells,
4      vector<CACell*> ** isOuterHitOfCell, int numberOfLayerPairs)
5  {
6
7    unsigned int layerPairIndex = blockIdx.y;
8    unsigned int cellIndexInLayerPair = threadIdx.x + blockIdx.x * blockDim.x;
9    if (layerPairIndex < numberOfLayerPairs)
10   {
11     int outerLayerId = gpuDoublets[layerPairIndex].outerLayerId;
12
13       auto& thisCell = cells[layerPairIndex][cellIndexInLayerPair];
14       thisCell.init(&gpuDoublets[layerPairIndex], gpuHitsOnLayers,
15             layerPairIndex, cellIndexInLayerPair);
16       isOuterHitOfCell[outerLayerId][thisCell.get_outer_hit_id()].push_back_ts(&(↩
             thisCell));
17   }
18 }
```

grid of one dimensional blocks of threads that work in parallel on all the graph of layers is built as follows:

- blocks of threads are distributed in two dimensions: x and y.
- The y coordinate of a block is associated to the identifier of a unique layer pair in the `CAGraph`. In the case of Pixel Tracks, its maximum value is 13.
- Each block contains threads distributed only along the x dimension.
- The number of threads in each block should be fixed and should be a multiple of the warp size.
- A thread is uniquely associated with a Cell.
- The number of blocks used in a row depends on the number of cells in a layer pair.

Listing 5.5 shows a simplified version of the creation kernel:

- line 1: `__global__` is a CUDA function attribute. It indicates that this function is a kernel, i.e. it is called from the CPU code and its content will be executed on the device.
- lines 7-8: there is no for-cycle. This kernel is executed by the entire grid. Each thread in each block, when reading built-in variables like `blockIdx.x`, sees a

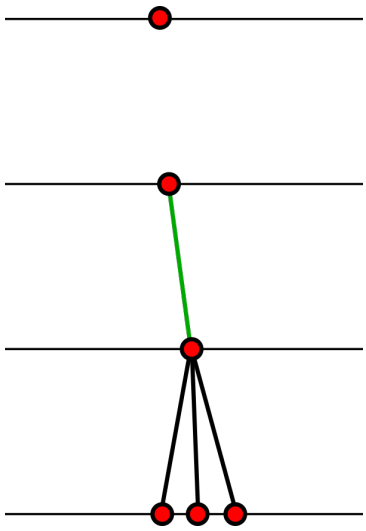


Figure 5.12: A cell (in green), looping over the three cells (in black) sharing its inner hit.

different version of it. As an example, thread 27 belonging to the block $(4, 7)$ will assign 7 to the variable `layerPairIndex`. The value of `cellIndexInLayerPair` depends on the number of threads per block. In the case of blocks containing 128 threads, `cellIndexInLayerPair` will be assigned a value of $27 + 128 * 4 = 539$.

- line 14: every thread initializes the cell that it is assigned to. The state, `CAState`, of the cell is initialized to 0.
- line 16: the pointer to the cell is stored into the `isOuterHitOfCell` in the location which corresponds to the outer hit index.

5.3.4 Cells Connections

The second step consists in finding neighbors for each Cell. Two cells are said to be *Neighboring Cells* if all of the following conditions are satisfied:

1. They belong to different layer pairs.
2. They have one hit in common, i.e. the inner hit for one cell is the outer hit for another cell.
3. They are aligned along the $R - z$ plane.

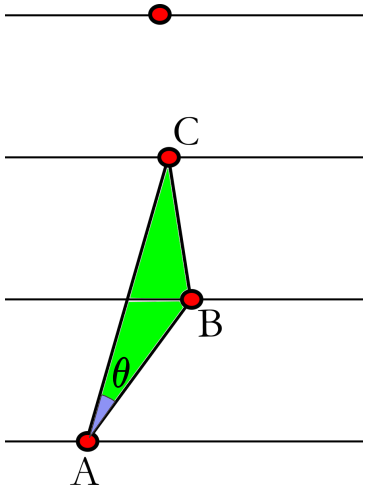


Figure 5.13: The longitudinal plane compatibility between two cells depends on the angle θ.

4. They are compatible with a circular trajectory coming from the beam-spot in the $x - y$ plane.

Longitudinal plane compatibility

In the $R-z$ plane, the alignment between two Cells, $\overline{AB}$ and $\overline{BC}$, is enforced by requiring the following conditions (Figure 5.13). The area (in green) between two cells can be written as:

$$\mathcal{A} = z_A(R_B - R_C) + z_B(R_C - R_A) + z_C(R_A - R_B) \tag{5.1}$$

The angle θ is related to $\mathcal{A}$ by the relation:

$$\tan(\theta) = \frac{2\mathcal{A}}{\text{dist}^2(A, C)} \approx \theta \tag{5.2}$$

for small values of θ. For multiple Coulomb scattering the following relation holds [158]:

$$\theta_{RMS} \propto \frac{1}{p} \tag{5.3}$$

It is possible to find a threshold value for θ_{RMS}, for a given value of p, which is passed via external configuration together with the seeding criteria:

$$2 \cdot \mathcal{A} \cdot p_{\text{min}} < \text{dist}^2(A, C) \cdot \kappa_{\text{long}} \tag{5.4}$$

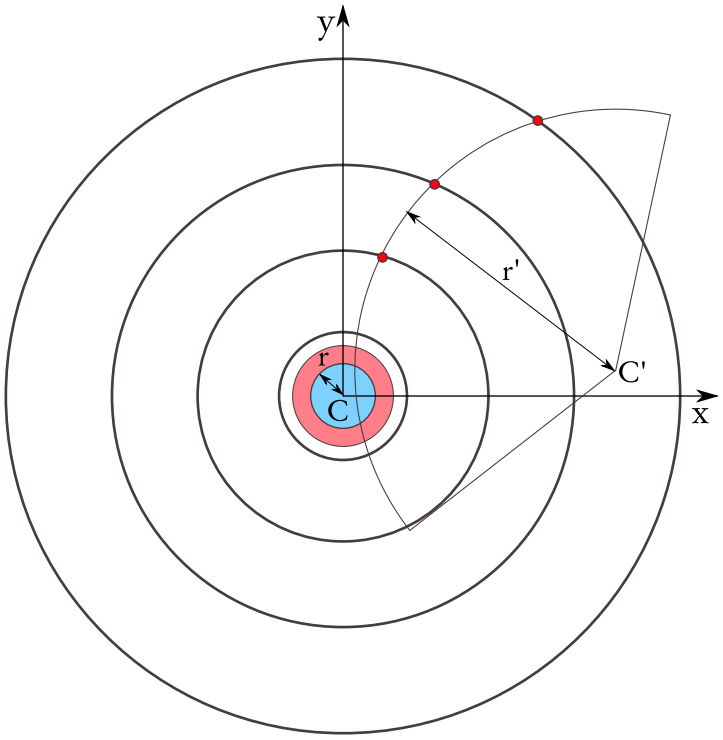


Figure 5.14: Transverse plane compatibility.

where κ_{long} is a constant that can be tuned as an external parameter, and depends on the detector structure and region hypothesis.

Transverse plane compatibility

In the $x - y$ plane, compatibility is calculated using the intersection between the circumference (C, r) given by the beam-spot from external configuration and the circumference (C', r') passing through the three hits belonging to the two cells (Figure 5.14). Calculating the intersection points between two circles can be computationally expensive and for this reason, only the coordinates of the center C' and the radius r' are computed.
Two circles intersect if the following condition holds:

$$r' - r \leq \text{dist}(C, C') \leq r' + r. \tag{5.5}$$

Multiple scattering and displaced tracks are taken into account by adding a tolerance ρ_{trans} to the radius of the beam-spot r (shown in red in Figure 5.14).

However, if the two cells are aligned, floating point precision could be insufficient to evaluate correctly a very large radius. Therefore, the straight line between the innermost point and the outermost point is required to intersect the beam-spot circumference.

Implementation

The grid of threads is still the same as for the creation kernel, where each thread is mapped to a cell and the y-coordinate of a block indicates the layer pair. In the connection kernel each cell loops over the cells that share its inner hit. The pointers to these cells for each hit are stored in the `isOuterHitOfCell` container, which had been filled in the creation phase. Figure 5.12 shows a cell (in green), looping over the three cells (in black) sharing its inner hit.
Now that the set of cells to loop over has been restricted to those that satisfy the condition number 1, it is possible to check the compatibility between those cells both in the longitudinal and in the transverse planes.
Since the compatibility check is called many times per Cell, it is important to find relations that do not involve slow transcendental function calls, square roots, divisions that take a high number of clock cycles.

5.3.5 Evolution

The previous steps have created a graph of interconnected Cells.

The purpose of the evolution step is to propagate the information of the length of a neighboring chain of cells to the innermost cells, in the least computationally expensive way possible.
Initially, during the creation step, every Cells' state has been set to 0. Consider two Cells, A and B, where A is the inner neighbor of B. Then, *the evolution rule* states that if $\texttt{CAState}_A$=$\texttt{CAState}_B$, then the state of the inner neighbor will increase by 1 in the next generation: $\texttt{CAState}_A = \texttt{CAState}_A + 1$.
This evolution rule is repeated for a number G of generations. After G generations, a cell with $\texttt{CAState} = G$, will start a chain of G more interconnected Cells. Therefore the *n-tuplet* originating from that cell will contain $G+2$ hits. As an example, if one is interested in quadruplets, the Cellular Automaton has to evolve at least twice (Figure 5.15).

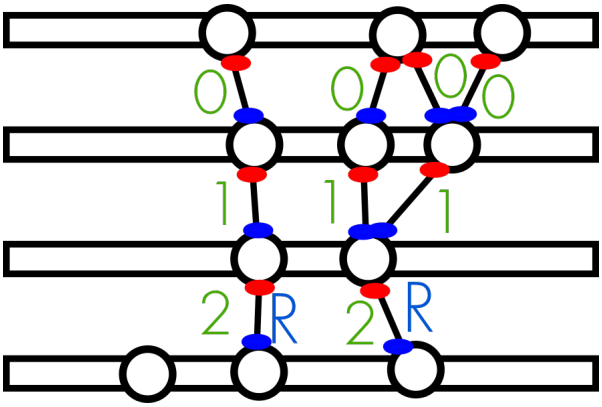


Figure 5.15: A Cellular Automaton that has evolved for two generations. Root cells are indicated with R.

Parallelism is evident and it is achieved in two steps per generation, in order to avoid race conditions. In the first step, for each cell satisfying the evolution rule, a flag `hasSameStateNeighbor` is set from 0 to 1. Then, a synchronization marks the beginning of the next generation. Finally, the `CAState` of each cell is increased by the value of its `hasSameStateNeighbor` flag.

5.3.6 Quadruplet Finder

If a Cell's `CAState` $= 2$ and its inner hit is on a root layer (BPix1, in the case of HLT Pixel Tracks), it is certainly the origin of a quadruplet and it is therefore marked as a Root cell (indicated by a blue letter R in Figure 5.15). A Depth-First Search (DFS) starts from each Root cell and looks for quadruplets [154]. A DFS is used because, in future implementations of the Hit-Chain Maker using the CA, one could prefer searching for all the n-tuplets up to n hits. As an example, if the CA approach is extended to the strip detector, it could be possible to fill different n-tuplet buckets: triplets, quadruplets, pentuplets, et cetera. The advantage of this approach is that these different buckets are disjoint sets: an element of the triplet bucket could not have been extended more and become a quadruplet.
In the final implementation, searches starting from two different Root cells are independent from each other and, therefore, they are performed by different threads.
After quadruplets have been found, they are fitted with a line in the longitudinal plane and with a circle in the transverse plane. If the resulting χ^2 does not exceed a threshold

set by external configuration, then the quadruplet goes to the final track fit used to get an estimate of the track parameters.

The pair of two hits is the building block of the algorithm, while a Graph is used for keeping track of the connections between layer pairs. This structure allows the implemented CA-based Hit-Chain Maker to be very flexible and it can be potentially extended to any geometry with an arbitrary number of layers, and also to reduce the number of duplicate seeds in case of overlapping modules within the same layer.

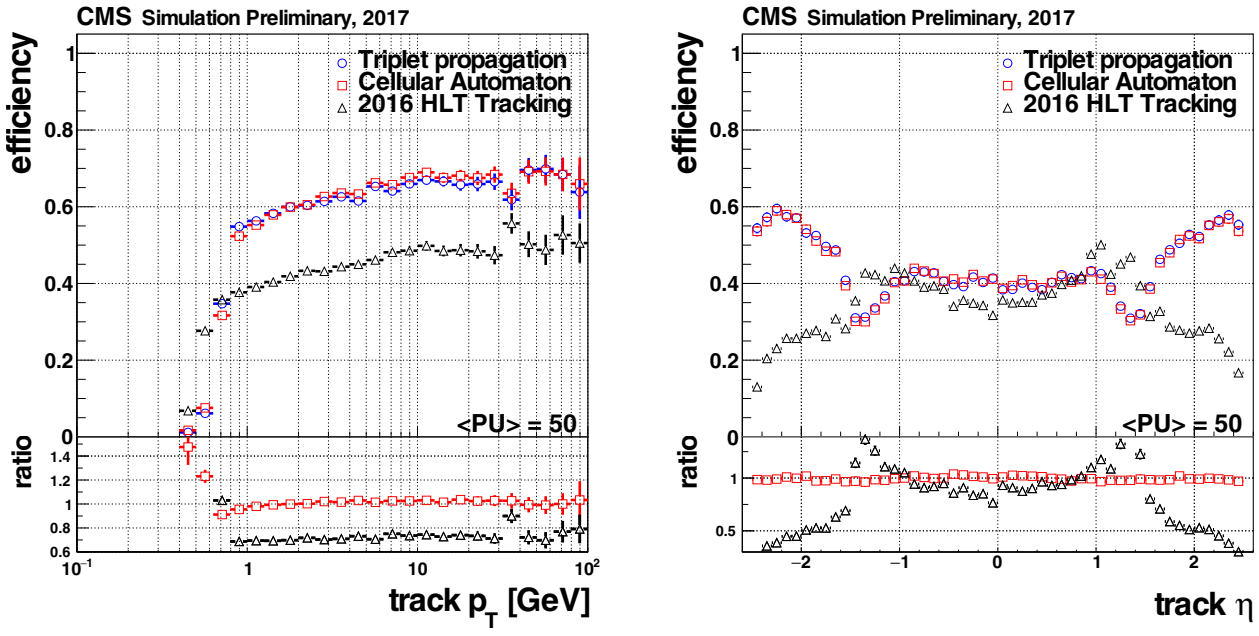
Chapter 6

Performance and Results

In this chapter, the performance of the CMS Phase-1 Tracking using the Hit-Chain Maker based on the CA track seeding algorithm described in Section 5.3 is evaluated.

6.1 Pixel Tracks using the CA-based Hit-Chain Maker

Referring to the definition used in Section 3.2.3, a reconstructed track contributes to the increase of the efficiency if more than 75% of the hits it contains have been produced by the same simulated particle. Therefore, in a quadruplet, the presence of one hit belonging to another simulated track would cause the quadruplet to be labeled as fake. The Pixel Tracks step at HLT is focused at reconstructing medium-high momentum prompt tracks, with $R^{\max} = 2$ mm, $z^{\max} = 24$ cm, and $p_{\mathrm{T}} > 0.9$ GeV/c. The tuning of the parameters of the Hit-Chain Maker based on the CA has been done with the objective of keeping similar efficiency, while rejecting more fake tracks if compared to the Triplet Propagation. The resulting tuned values for the HLT Pixel Tracks are $\kappa_{\mathrm{long}} = 0.0012$ and $\rho_{\mathrm{trans}} = 2$ mm.
The performance of the Cellular Automaton-based Hit-Chain Maker running at the HLT Pixel Tracks was evaluated with respect to the 2016 Pixel Tracking and the Triplet Propagation algorithm, described in Section 5.3. The sample used to evaluate the performance comprises 1000 simulated $\mathrm{t\bar{t}}$ events at an energy of $\sqrt{s} = 13$ TeV, with a flat pile-up distribution with an average of 50 superimposed pile-up collisions. This sample was reconstructed using CMSSW version 8.1.0.
Figure 6.1 shows the track reconstruction efficiency at the Pixel Tracks step for the



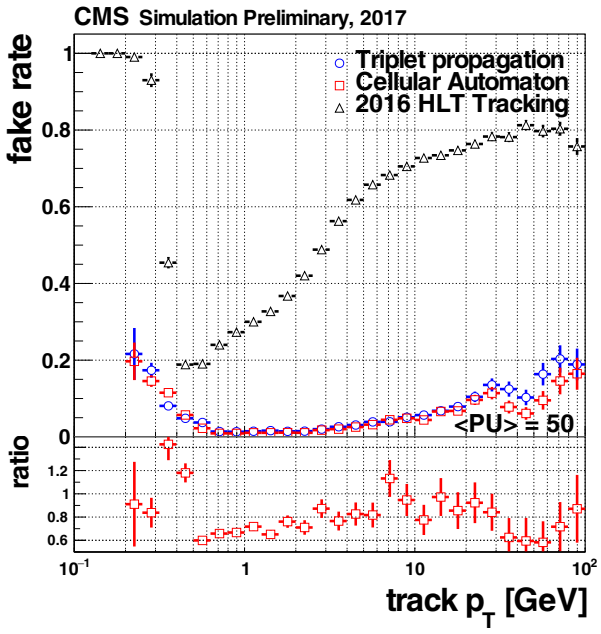
(a) Efficiency vs p_T

(b) Efficiency vs η

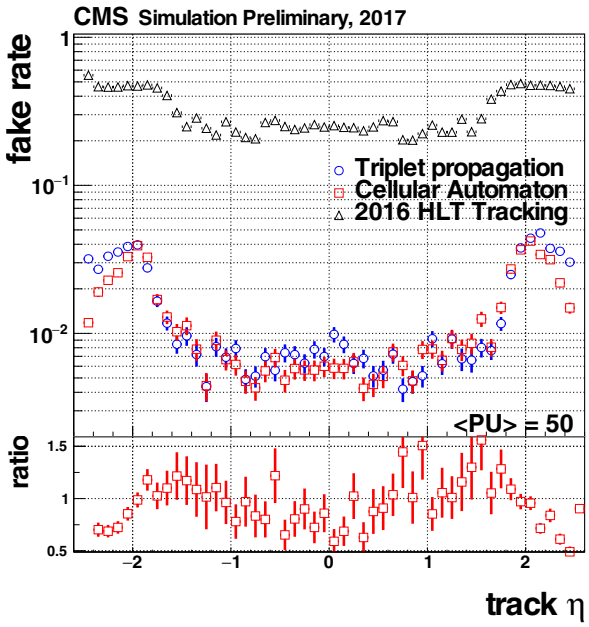
Figure 6.1: Pixel Tracks reconstruction efficiency for simulated $t\bar{t}$ events with a flat pile-up distribution with an average of 50 superimposed pile-up collisions, showing in blue the Cellular Automaton-based Hit-Chain Maker algorithm, in red Triplet Propagation and in black the 2016 Pixel Tracking. The ratio pad underneath shows the performance of each algorithm with respect to the CA-based Hit-Chain Maker.

Cellular Automaton algorithm compared to the Triplet Propagation and the 2016 Pixel Tracking. The drop in efficiency for quadruplets algorithms in the pseudo-rapidity interval $1 < |\eta| < 1.5$ is due to the passage from the barrel region to the forward region. Here, the four-hits coverage is not complete. Possible later iterations that make use of triplets could recover part of this inefficiency. Furthermore, the 2016 Pixel Tracking's efficiency is about 30% lower than the CA approach. This is mainly caused by the different detector geometry.

The fake-rate is shown in Figure 6.2. The reduction of the fake-rate by almost two orders of magnitude is an advantage coming from the upgrade of the Pixel Detector. This is mainly due to the much higher purity of quadruplets with respect to triplets. The main reason why the fake-rate grows with decreasing transverse momentum below $p_{\mathrm{T}} = 0.3$ GeV/c is due to the minimum value of the transverse momentum configured in the seeding region. For Pixel Tracks at HLT, this value is $p_{\mathrm{T}}^{\mathrm{min}} = 0.9$ GeV/c and



(a) Fake-rate vs p_T



(b) Fake-rate vs η

Figure 6.2: Pixel Tracks reconstruction fake-rate for simulated $t\bar{t}$ events with 50 superimposed pileup collisions. In blue the Cellular Automaton algorithm, in red Triplet Propagation and in black the 2016 Pixel Tracking. The ratio pad underneath shows the performance of each algorithm with respect to the CA.

compatible hits are filtered following this hypothesis when computing seeds. This makes the correct reconstruction of very low transverse momentum tracks difficult. If compared to the Triplet Propagation algorithm, the CA Pixel Tracking shows a fake-rate which is lower by up to 40%.

6.1.1 Latency measurements, configuration and results

A machine with the following configuration has been employed to carry out measurements of the execution time:

- Intel Core i7-4771 CPU @ 3.50GHz, 4 cores, 8 hw-threads,
- 16 GByte main memory,
- Scientific Linux CERN release 6.7,

Table 6.1: Timing performance of Pixel Tracks for different algorithms.

Algorithm	Timing
GPU Cellular Automaton	(1.2 ± 0.9) ms
CPU Cellular Automaton	(14.0 ± 6.2) ms
Triplet Propagation	(72.1 ± 25.7) ms
2016 Pixel Tracks	(29.3 ± 13.1) ms

- NVIDIA GTX 1080,
- CUDA v8.0,
- CMSSW 8.1.0.

The sample used to measure timing is the same sample used for evaluating the physics performance shown earlier in this section. In all the three cases, the time interval measured are delimited by:

- The beginning of the seeding function.
- The found quadruplets or triplets are returned and are ready for the final fit.

The results of the measurements are shown in Table 6.1. The CA-based Hit-Chain Maker executing on a GPU is about 60× faster than the Triplet Propagation and about 24× faster than 2016 Pixel Tracking running on CPUs.

6.1.2 Integration and Demonstrators

Demonstrators of the GPU-based Cellular Automaton at the HLT Pixel Track step will be created during 2017, once all the steps from RAW data to Pixel Tracks have been developed to run on a GPU. Three approaches of integration have been identified, each with its assets and liabilities.

GPU-enhanced Builder Units

Figure 6.3 shows the approach in which Builder Units are GPU-accelerated. This method requires the evaluation of Pixel Tracks on GPUs to be executed on every event, with

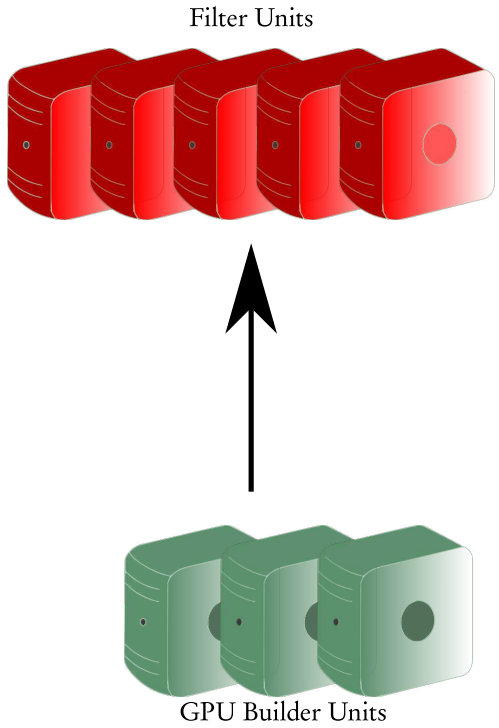


Figure 6.3: Simplified architecture of an HLT farm with GPU-enhanced Builder Units.

a number of GPUs that is constrained by the number of Builder Units. However, this method would have the performance of standalone applications without requiring to restructure the architecture of the farm.

A demonstrator for the GPU-enhanced Builder Units was created with the following test configuration:

- 2 sockets x Intel(R) Xeon(R) CPU E5-2650v4 2.20GHz (12 physical cores).
- 256 GByte main memory.
- Scientific Linux CERN release 7.2.
- 8× GPUs NVIDIA GTX 1080Ti.
- CUDA v8.0.

A test was performed for measuring sustained event rate and energy efficiency. The sample used to evaluate the performance comprises 100 simulated $t\bar{t}$ events at an energy

of $\sqrt{s} = 13$ TeV, with a flat pile-up distribution with an average of 50 superimposed pile-up collisions. Before the test the following actions were performed:

- Hit doublets for 100 events were preloaded in the host memory.
- Each CPU core was assigned a host thread.
- Each GPU was assigned a host thread.
- Memory needed to process 8 events concurrently on each GPU is preallocated.
- A concurrent queue is filled with 25 million indices of events to process.

During the test, when a host thread becomes idle, it tries to pop a new event index from the queue:

- If the host thread is associated to a GPU, doublets are copied to the GPU, the GPU version of the Hit-Chain Maker is executed and quadruplets are copied back to the host memory.
- If the host thread is associated to a CPU core, the CPU version of the Hit-Chain Maker is executed.

The test goes one until the event queue is empty. During the test the number of events processed by each host thread in the unit time was monitored, as well as the electric power used by the processor using the tools *turbostat* for the CPU, *nvidia-smi* for the GPU. The same test was also performed for a CPU-only configuration, with no host thread associated to a GPU.
Results are shown in Table 6.2 and Table 6.3. The results of the test show that it is possible to execute the Hit-Chain Maker on all the events with a low number of GPU-enhanced Builder Units. The hybrid configuration, which uses the GPUs together with the installed CPU cores can achieve a higher energy efficiency with respect to a traditional CPU-only configuration. The amount of power needed to run 14 hybrid nodes is about 17.2 kW, while the CPU-only configuration would need 24.7 kW to produce the same results using 128 nodes.

Table 6.2: Sustained event rate of the Hit-Chain Maker running in the GPU-enhanced Builder Units demonstrator. The number of nodes required to be able to sustain an event input rate of 100 kHz was also extrapolated.

Configuration	GPU Rate (Hz)	CPU Rate (Hz)	Nodes to sustain 100 kHz
Hybrid	6527	613	14
CPU-only	N/A	777	128

Table 6.3: Energy efficiency performance of the Hit-Chain Maker running in the GPU-enhanced Builder Units demonstrator. The average power consumption was measured using internal registers. The energy efficiency was calculated as the ratio between the average event rate and the average power consumption.

Configuration	GPU Power (W)	CPU Power (W)	Evts/J
Hybrid	1037	191	GPU: 6.3, CPU: 3.2
CPU-only	N/A	191	4.1

Heterogeneous HLT farm

In the heterogeneous HLT farm approach, the Builder Units send packed events to the Filter Units where the processing and filtering happens. However, the RAW data coming from the Pixel Detector is sent to a dedicated GPU node, where the evaluation of the Pixel Tracks is executed on GPUs. Then, the results are transmitted to the correct Filter Unit for that event (Figure 6.4).

The communication protocol is be based on *Message Passing Interface* (MPI) [159], with the addition of GPU Direct [160]. GPU Direct allows network packets to be read directly from the network interface by bypassing the Linux kernel in a zero-copy fashion. In the context of this thesis work a framework called *PATATRACK* has been developed [161]. A circular buffer is allocated by the GPU Direct driver which contains pointers to the incoming packets in the GPU memory. The trigger framework can read packets by simply dereferencing those pointers and updating the index to the next slot that will host the next incoming packets. Communication with the network adapter happen in *Direct Memory Access* (DMA). This means that the network adapter can access system memory independently of the CPU, generating only one interrupt per transferred block (instead of one interrupt per transferred byte).

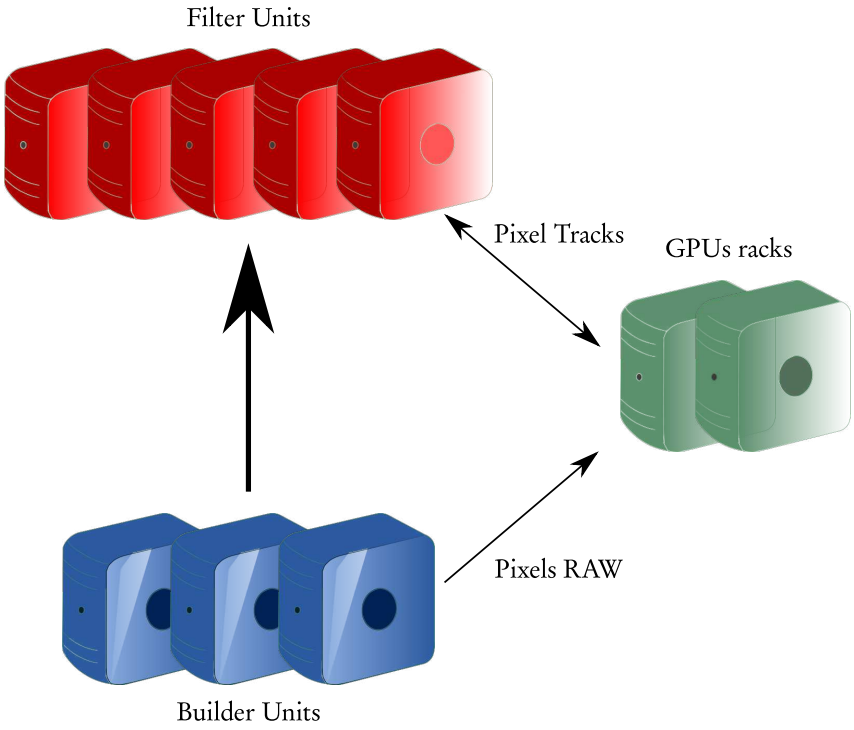


Figure 6.4: Simplified architecture of a heterogeneous HLT farm.

The strengths of this approach are:

- The scale of the system is flexible, and can be adjusted based on needs independently from the rest of the farm.
- The system can achieve better timing performance as it can run in a standalone application and more events can be processed at the same time by the same kernel increasing the overall throughput.
- The GPU machines can be used during the periods of time in which no data is taken to train deep neural networks.
- The intermediate products (e.g. Pixel Clusters) never leave the GPU memory.
- It's future-ready: when events will become more complex during LHC Run 4, it would be possible to execute tracking or clustering for one event on different nodes for different portions of the detector.

On the other side the weaknesses are the following:

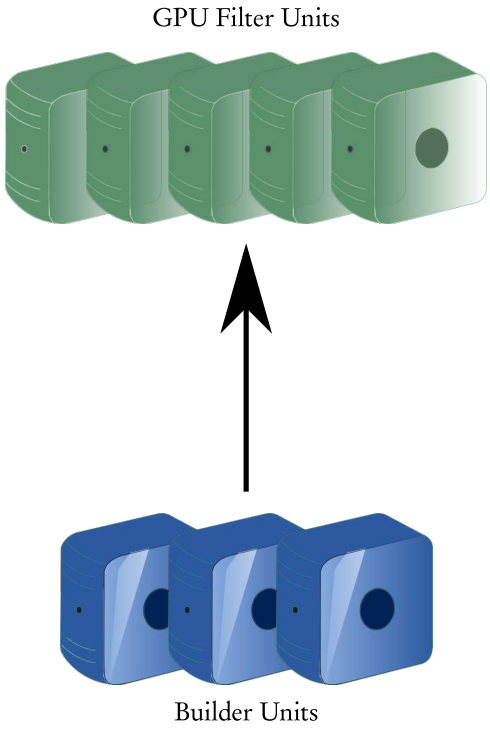


Figure 6.5: Simplified architecture of an HLT farm with GPU-enhanced Filter Units.

- Builder Units will have to send the RAW data along with an identifier of the Filter Units that needs that results.
- Filter Units need to be able to offload computation to another node and compute some other task while waiting for the processed Pixel Tracks.
- Detector conditions have to be received and interpreted by the standalone application.

GPU-enhanced Filter Units

Another approach consists in equipping each Filter Unit with one or more GPUs (Figure 6.5). This method requires GPU code to be integrated in CMSSW with its limitations:

- Process one event at a time per CPU thread.
- Memory traffic between CPU and GPU.

- It won't be possible to run one event on many nodes/GPUs without MPI support.

However, a strong asset is that there would not be any need of redesign the architecture of the farm.

The GPU-enhanced Builder Unit approach is the main candidate to be implemented at the High-Level trigger farm during the LHC Run 3 and 4 [162].

6.2 Sequential CA-based Hit-Chain Maker in the online tracking

The improved physics performance of the CA-based Hit-Chain Maker with respect to the Triplet Propagation implementation and the possibility to enable fine-grained parallelism within one event, lead to the introduction of a sequential version of the Cellular Automaton during the LHC Run-2. This earlier introduction required the porting of the CUDA implementation of the CA-based Hit-Chain Maker to C++.
Including the track seeding based on the Cellular Automaton during LHC Run-2 has two main advantages. First, it would make the transition to the CA-based Hit-Chain Maker running on GPUs during the LHC Run 3 in 2020 much easier. This is motivated by the fact that the CUDA and the sequential implementations produce the same results. Secondly, code designed to be executed in parallel can perform better than purely sequential implementations, when executed on a CPU using one thread. This is due to the attention paid to the development of parallel-friendly data structures and optimized memory access pattern, which is required when developing a parallel program. The same data structures used in the CUDA implementation have been reused and replaced by their C++ Standard Template Library [163] equivalent. By running this version on the same data samples reconstructed in Section 6.1, the same physics results were obtained.

An overview of the physics performance showing the result of a first tuning of the Cellular Automaton at the Pixel Tracks step is shown in Figure 6.6. The Pixel Tracks seeding is executed at Iter0 with the seeding region $p_{\mathrm{T}}^{\mathrm{min}} = 0.9$ GeV. Iter1 recovers efficiency in the low momentum region below 1 GeV. By adding a triplet iteration, Iter2, efficiency can reach a plateau around 80%.

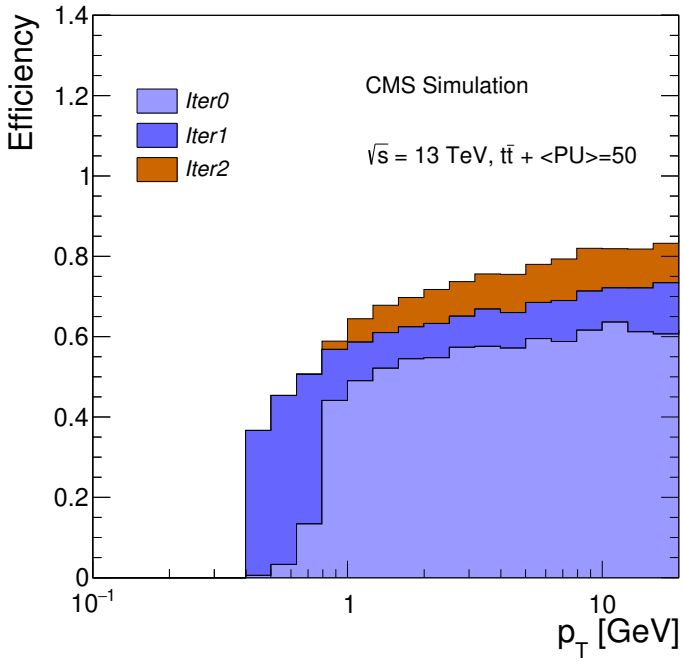


Figure 6.6: Stacked efficiency plot for the Online Iterative Tracking with the Cellular Automaton Pixel Tracks.

The CPU timing, measured using the same methodology, is (14.0 ± 6.2) ms. This makes the sequential version about 12 times slower than the GPU version, but respectively 15 and 2 times faster than the Triplet Propagation and 2016 Pixel Tracking.

6.3 Sequential CA-based Hit-Chain Maker in the offline track seeding

The physics performance of the 2017 track reconstruction based on the Cellular Automaton track seeding has been assessed and compared to the track reconstruction based on legacy triplets and their propagation. All the iterations of the offline iterative tracking that make use of pixel tracking (Table 6.4), have been ported to use the CA-based Hit-Chain Maker algorithm.
Tuning the CA parameters for the offline track reconstruction is more complex than for HLT Pixel Tracks. Not only the number of iterations is higher, but there are dependencies between iterations, as the output of an iteration is the input of the following one. The

resulting tuned parameters are shown in Table 6.4. The tuning of the parameters has been achieved by tightening the cuts of one iteration until efficiency starts to be affected. If two configurations give the same physics result, the one with the smallest execution time has been chosen. Once the parameters of one iteration have been tuned, they are fixed and the tuning of the next iteration can start. The performance of the offline

Table 6.4: Iterations of the 2017 offline iterative tracking ported to the CA-based Hit-Chain Maker seeding.

Iteration	Seeds	κ_{long}	ρ_{trans} [mm]
InitialStep[1]	pixel quadruplets	0.0012	2
LowPtQuad	pixel quadruplets	0.0017	3
HighPtTriplet	pixel triplets	0.004	0.7
LowPtTriplet	pixel triplets	0.002	0.5
DetachedQuad	pixel quadruplets	0.0011	0
DetachedTriplet	pixel triplets	0.001	0

track reconstruction based on the Cellular Automaton has been compared to the 2016 track reconstruction (Figure 6.7). The reconstruction efficiency is increased, especially in forward region. The fake-rate is significantly reduced in the entire pseudo-rapidity range. The resolution of the determination of the transverse momentum does not benefit much on the upgraded detector (Figure 6.8). On the other side, Figure 6.9 shows that the impact parameters resolution is improved over all the η spectrum.

The physics performance of the track reconstruction which uses the CA-based Hit-Chain Maker seeding has been compared to the best alternative, the Triplet Propagation seeding, described in Section 5.3. The results of this comparison are shown in Figure 6.10. The dataset used for the comparison comprises 10000 simulated $t\bar{t}$ events with 35 superimposed pileup collisions. The overall track reconstruction efficiencies are very similar while the fake-rate when seeding uses the Cellular Automaton becomes lower by up to about 40% when transitioning from the barrel to the forward region with $|\eta| > 1.5$.

It is important to study how the track reconstruction performance changes at different pile-up conditions. The results of this study for the Triplet Propagation seeding are shown in Figure 6.11 and Figure 6.12, while Figure 6.13 and Figure 6.14 show the

[1] In the original implementation, the Initial Step is based on triplet extension seeding: a triplet is found and then a Kalman filter is used to find the fourth hit.

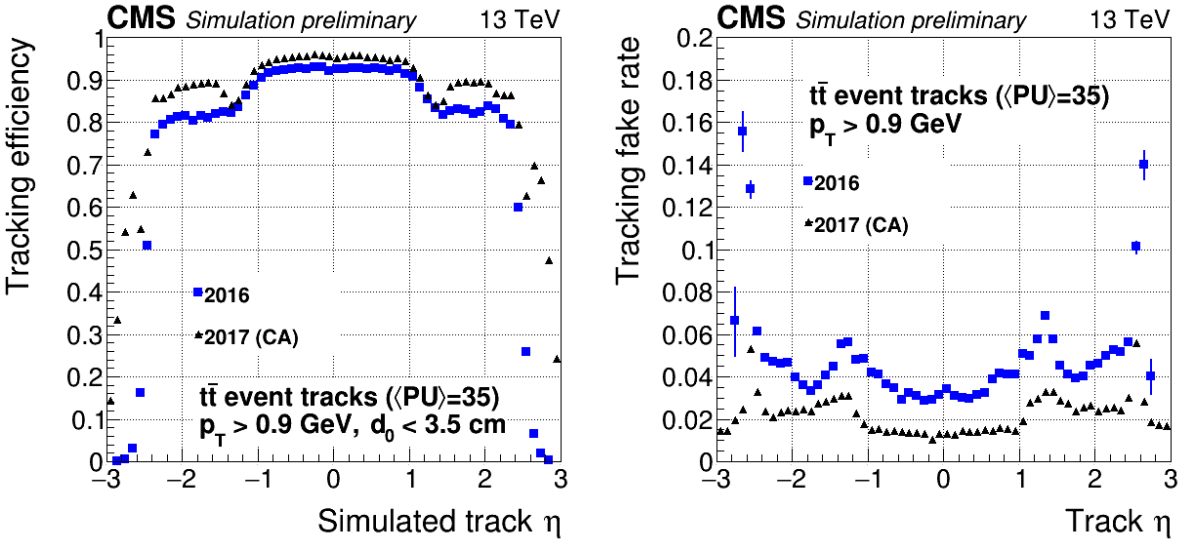
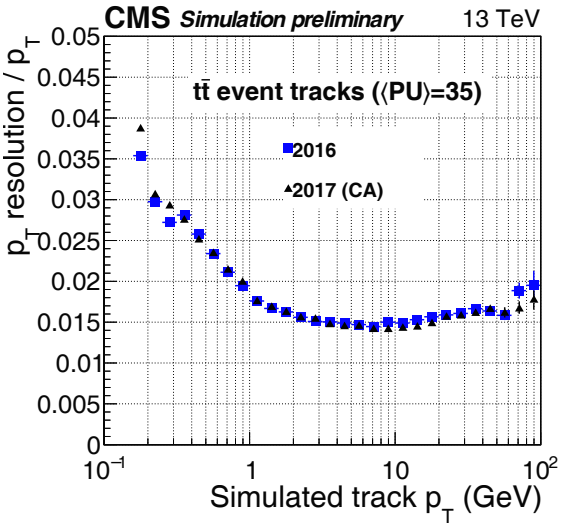
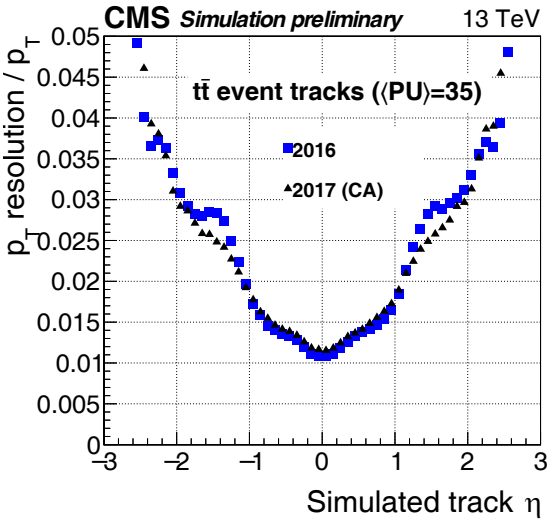


Figure 6.7: Performance of the track reconstruction using the CA-based Hit-Chain Maker seeding, compared to the 2016 track reconstruction. The performance has been measured for 10000 simulated $t\bar{t}$ events with 35 superimposed pileup collisions. Only simulated tracks with transverse momentum greater than 0.9 GeV and transverse impact parameter lower than 3.5 cm contribute to the efficiency. Only reconstructed tracks with transverse momentum greater than 0.9 GeV contribute to the fake-rate [164].

physics performance computed after replacing the Triplet Propagation algorithm with the CA-based Hit-Chain Maker on the same input dataset. The efficiency is slightly affected by the increasing pile-up. This is caused by the fact that more fake tracks pass the track selection and their hits get removed from the pool of available hits, making the reconstruction of other simulated tracks more difficult. The efficiency of the two seeding algorithms is substantially the same. The CA-based Hit-Chain Maker shows a fake rejection which is better than Triplet Propagation by few percent, especially in the endcap region $|\eta| > 1$.

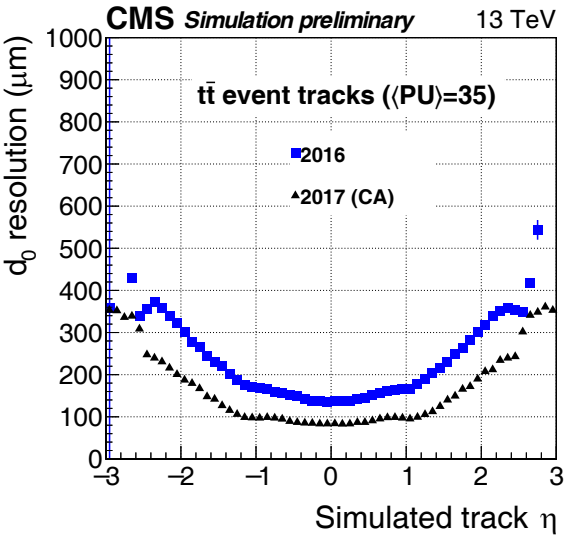


(a) p_{T} resolution vs p_T

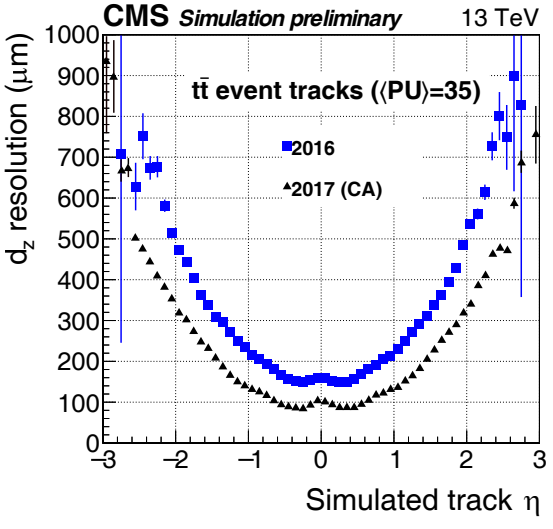


(b) p_{T} resolution vs η

Figure 6.8: Track p_{T} resolution as a function of the simulated track p_{T} (left) and η (right) for 2016 and 2017 tracking, this latter using the CA-based Hit-Chain Maker. The p_{T} resolution improves in $1.2 < |\eta| < 1.6$, because the fourth pixel layer yields better precision on the track extrapolation to the strip tracker in the pixel barrel-forward transition.



(a) Efficiency vs p_T



(b) Efficiency vs η

Figure 6.9: Resolutions of the track transverse impact distance d_0 (left) and longitudinal impact point d_z (right) as a function of the simulated track η for 2016 and 2017 detectors. The 2017 detector shows better performance than 2016 over all the η spectrum.

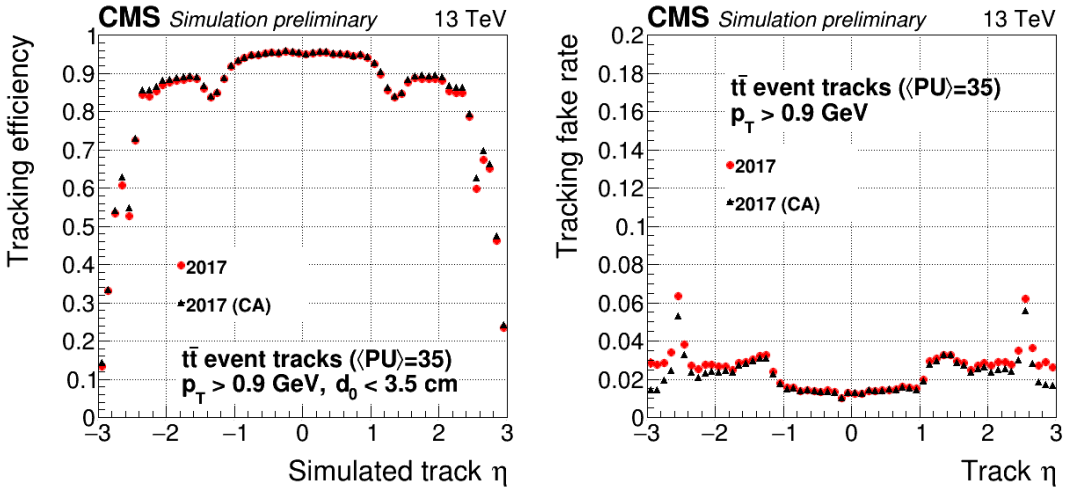


Figure 6.10: Performance of the track reconstruction using the CA-based Hit-Chain Maker seeding, compared to the Triplet Propagation tracking. The performance has been measured for 10000 simulated $t\bar{t}$ events with 35 superimposed pileup collisions. Only simulated tracks with transverse momentum greater than 0.9 GeV and transverse impact parameter lower than 3.5 cm contribute to the efficiency. Only reconstructed tracks with transverse momentum greater than 0.9 GeV contribute to the fake-rate [164].

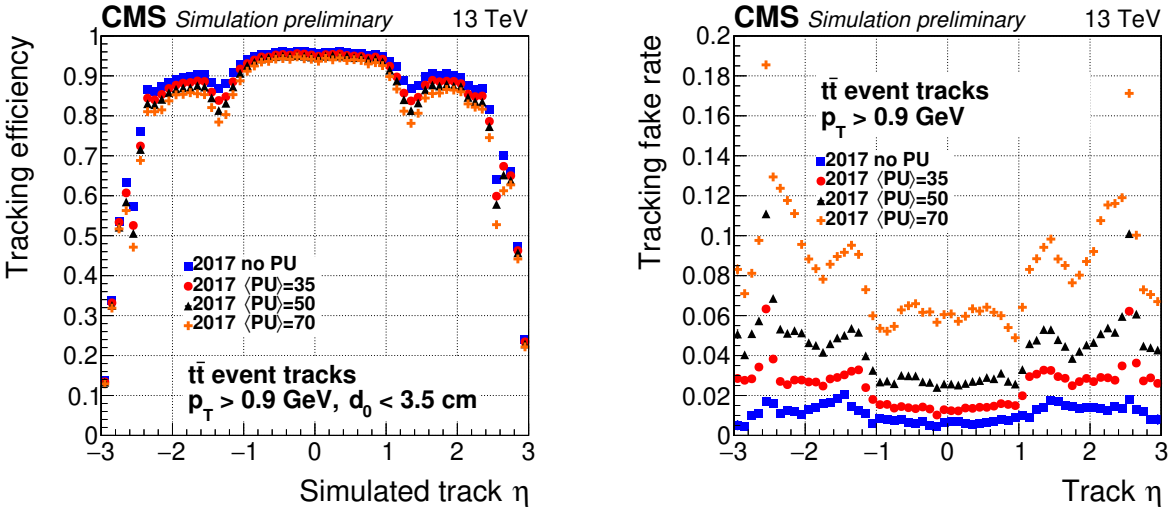


Figure 6.11: Performance of the track reconstruction using the Triplet Propagation seeding as a function of η at different pile-up conditions. The performance has been measured for 10000 simulated $t\bar{t}$ events. Only simulated tracks with transverse momentum greater than 0.9 GeV and transverse impact parameter lower than 3.5 cm contribute to the efficiency. Only reconstructed tracks with transverse momentum greater than 0.9 GeV contribute to the fake-rate [164].

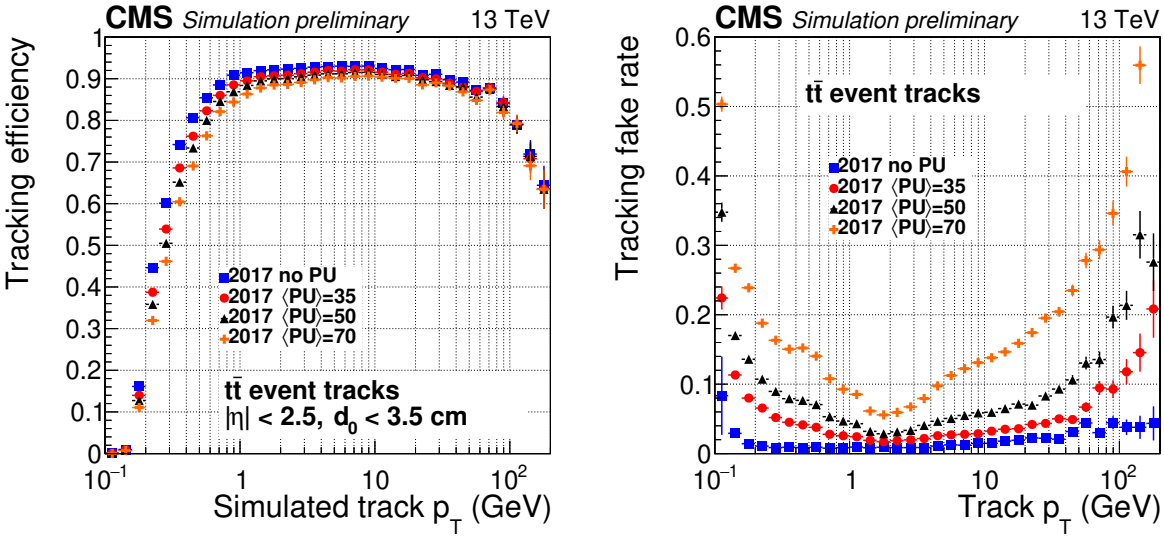


Figure 6.12: Performance of the track reconstruction using the Triplet Propagation seeding as a function of p_{T} at different pile-up conditions. The performance has been measured for 10000 simulated $\mathrm{t\bar{t}}$ events. Only simulated tracks with $|\eta| < 2.5$ and transverse impact parameter lower than 3.5 cm contribute to the efficiency [164].

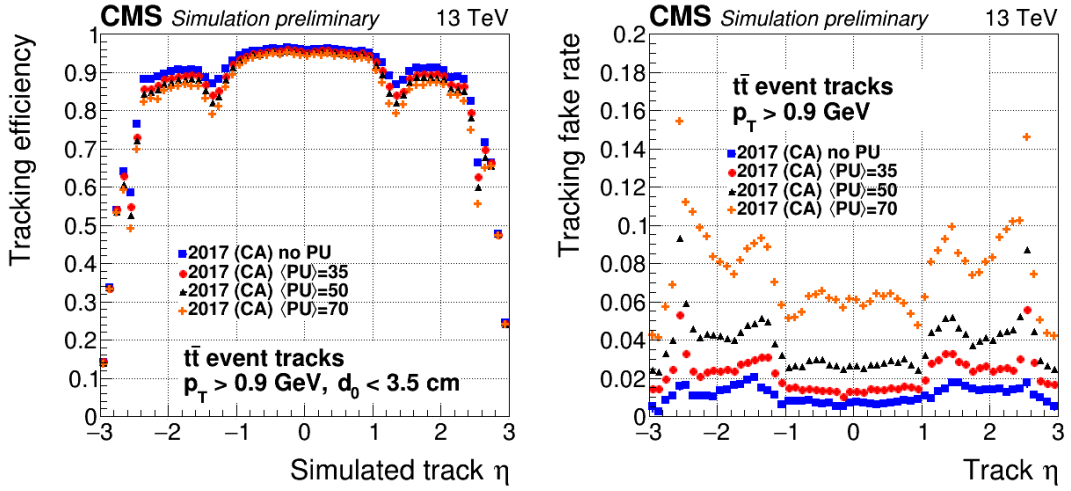


Figure 6.13: Performance of the track reconstruction using the CA-based Hit-Chain Maker seeding as a function of η at different pile-up conditions. The performance has been measured for 10000 simulated $\mathrm{t\bar{t}}$ events. Only simulated tracks with transverse momentum greater than 0.9 GeV and transverse impact parameter lower than 3.5 cm contribute to the efficiency. Only reconstructed tracks with transverse momentum greater than 0.9 GeV contribute to the fake-rate [164].

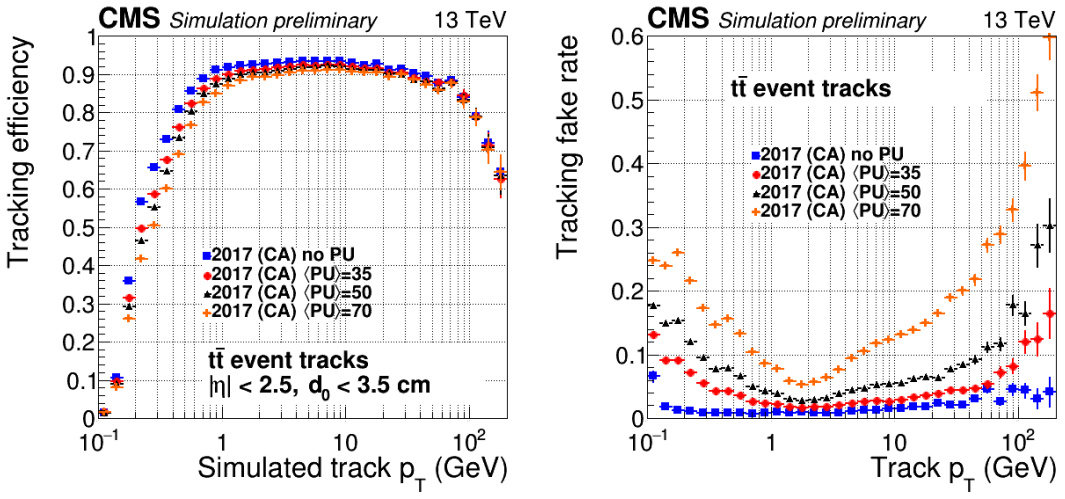


Figure 6.14: Performance of the track reconstruction using the CA-based Hit-Chain Maker seeding as a function of p_{T} at different pile-up conditions. The performance has been measured for 10000 simulated $\mathrm{t\bar{t}}$ events. Only simulated tracks with $|\eta| < 2.5$ and transverse impact parameter lower than 3.5 cm contribute to the efficiency [164].

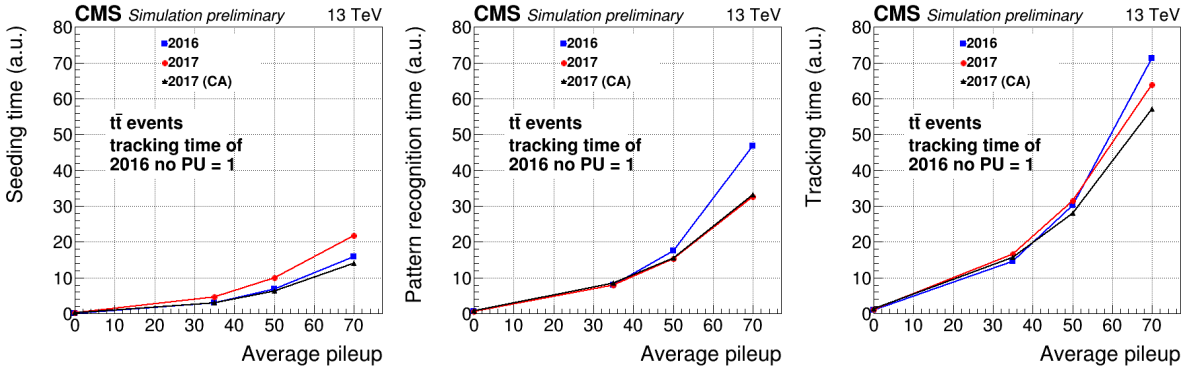


Figure 6.15: Track seeding (left), track building (center) and total track reconstruction (center) execution time as a function of average pile-up. The 2016 tracking (in blue), the 2017 tracking with Triplet Propagation seeding (in red) and the 2017 tracking with the CA-based Hit-Chain Maker seeding (in black) are compared. All time measurements are normalized such that 1 indicates the tracking time of 2016 tracking without pileup [164].

The execution time of the full track reconstruction improves when using the CA-based Hit-Chain Maker seeding (Figure 6.15). The execution time of the CA-based Hit-Chain Maker seeding is at the same level of the 2016 seeding and increases slightly slower with pile-up than 2016 track seeding despite the increased number of layer combinations involved in the seeding phase with respect to the 2016 seeding. As expected, in pattern recognition, the CA-based Hit-Chain Maker seeding brings no additional gain. The slower increase of the Cellular Automaton's execution time as a function of the pile-up with respect to Triplet Propagation seeding, contributes to make the overall execution time of the sequential track reconstruction about 20% faster than 2016 tracking for an average pile-up of 70. If offline track seeding was running onto GPUs, its timing would become negligible with respect to the rest of the track reconstruction, making the track reconstruction up to two times faster with respect to 2016 tracking for an average pile-up of 70.
An evaluation of the performance of the CA-based Hit-Chain Maker in the upgraded Phase-2 CMS Tracker is being carried out during 2017, in conditions of pile-up going from 140 and 200 simultaneous proton-proton collisions.

Conclusion and Outlooks

The future runs of the Large Hadron Collider (LHC) at CERN will impose significant challenge on the software performance, due to the increasing complexity of events. In fact, many event reconstruction algorithms have to explore many combinations of possible paths, making the performance not scaling linearly with the number of simultaneous proton collisions.

The exploitation of massively parallel computer architectures and innovative software design techniques can successfully be accomplished in order to cope with the increasing event complexity.

This thesis work presents new ways to solve the compute intensive problem of track seeding in the CMS Pixel Detector: a parallel k-d tree data structure, FKDTree, and a Hit-Chain Maker algorithm based on the concepts of Cellular Automata.
A careful assessment of its physics and timing performance demonstrated that the Hit-Chain Maker improves physics performance while being significantly faster than the existing sequential implementation. For these reasons it has replaced the existing track seeding algorithm starting from the 2017 data-taking.
This new algorithm has the potential of running on heterogeneous computing systems exploiting massively parallel GPU architectures both in online trigger systems and offline reconstruction software.

Bibliography

[1] CMS Collaboration. "The CMS experiment at the CERN LHC". In: *JInst* 3.08 (2008), S08004.

[2] ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08003 (2008).

[3] LHC Study Group. *The large hadron collider, conceptual design*. Tech. rep. CERN/AC/95-05 (LHC) Geneva, 1995.

[4] CMS Collaboration. "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.

[5] 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 (2012), pp. 1–29.

[6] CMS Collaboration. *Technical proposal for the upgrade of the CMS detector through 2020*. Tech. rep. 2011.

[7] CMS Collaboration. "Technical proposal for the phase-II upgrade of the CMS detector". In: *CERN, CERN-LHCC-2015-010. LHCC-P-008* (2015).

[8] Thomas Speer et al. "Track reconstruction in the CMS tracker". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 559.1 (2006), pp. 143–147.

[9] CMS Collaboration. "The CMS high level trigger". In: *arXiv preprint hep-ex/0512077* (2005).

[10] David B Kirk and W Hwu Wen-Mei. *Programming massively parallel processors: a hands-on approach*. Morgan Kaufmann, 2016.

[11] William David Ross et al. *Aristotle's metaphysics*. Vol. 2. Clarendon Press, 1924.

[12] Robert Brown. "XXVII. A brief account of microscopical observations made in the months of June, July and August 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies". In: *Philosophical Magazine Series 2* 4.21 (1828), pp. 161–173.

[13] Joseph-Louis Proust. "Researches on copper". In: *Ann. chim* 32 (1799), pp. 26–54.

[14] Joseph John Thomson. "XL. Cathode rays". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 44.269 (1897), pp. 293–316.

[15] Robert A Millikan. "XXII. A new modification of the cloud method of determining the elementary electrical charge and the most probable value of that charge". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 19.110 (1910), pp. 209–228.

[16] Ernest Rutherford. "LXXIX. The scattering of α and β particles by matter and the structure of the atom". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 21.125 (1911), pp. 669–688.

[17] Niels Bohr. "XXXVII. On the constitution of atoms and molecules". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 26.153 (1913), pp. 476–502.

[18] James Chadwick. "Possible existence of a neutron". In: *Nature* 129.3252 (1932), p. 312.

[19] Walther Gerlach and Otto Stern. "Der experimentelle nachweis der richtungsquantelung im magnetfeld". In: *Zeitschrift für Physik A Hadrons and Nuclei* 9.1 (1922), pp. 349–352.

[20] Max Karl Ernst Ludwig Planck. "Über eine Verbesserung der Wienschen Spectralgleichung". In: *Verhandl. Dtsc. Phys. Ges.* 2 (1900), p. 202.

[21] Albert Einstein. "Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt". In: *Annalen der physik* 322.6 (1905), pp. 132–148.

[22] Albert Einstein. "Zur elektrodynamik bewegter körper". In: *Annalen der physik* 322.10 (1905), pp. 891–921.

[23] Hendrik Antoon Lorentz et al. *The principle of relativity: a collection of original memoirs on the special and general theory of relativity*. Courier Corporation, 1952.

[24] Paul AM Dirac. "The quantum theory of the electron". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. Vol. 117. 778. The Royal Society. 1928, pp. 610–624.

[25] Carl D Anderson. "The apparent existence of easily deflectable positives". In: *Science* 76 (1932), pp. 238–239.

[26] Sin-itiro Tomonaga. "On a relativistically invariant formulation of the quantum theory of wave fields." In: *Progress of Theoretical Physics* 1.2 (1946), pp. 27–42.

[27] Julian Schwinger. "On quantum-electrodynamics and the magnetic moment of the electron". In: *Physical Review* 73.4 (1948), p. 416.

[28] Julian Schwinger. "Quantum electrodynamics. I. A covariant formulation". In: *Physical Review* 74.10 (1948), p. 1439.

[29] Richard Phillips Feynman. "Space-time approach to quantum electrodynamics". In: *Physical Review* 76.6 (1949), p. 769.

[30] Wolfgang Ernst Friederich Pauli. "Open letter to radioactive persons". In: (1930).

[31] Enrico Fermi. "An attempt of a theory of beta radiation. 1". In: *Z. phys* 88.161 (1934), p. 19.

[32] Bruno Pontecorvo. "Inverse Beta Process". In: *Chalk River Laboratory Report PD-205* (1948).

[33] Marcello Conversi, Ettore Pancini, and Oreste Piccioni. "On the disintegration of negative mesons". In: *Physical Review* 71.3 (1947), p. 209.

[34] Enrico Fermi, Edward Teller, and Victor Weisskopf. "The decay of negative mesotrons in matter". In: *Physical Review* 71.5 (1947), p. 314.

[35] Cesare Mansueto Giulio Lattes, GPS Occhialini, and Cecil Frank Powell. "Observations on the tracks of slow mesons in photographic emulsions". In: *Nature* 160.4066 (1947), pp. 453–456.

[36] Bruno Pontecorvo. "Nuclear capture of mesons and the meson decay". In: *Physical Review* 72.3 (1947), p. 246.

[37] LM Lederman, M Schwartz, and J Steinberger. "Observation of high-energy neutrino reactions and the existence of two kinds of neutrinos". In: *Physical Review Letters* 9.1 (1962), p. 36.

[38] Martin L. et al Perl. “Evidence for anomalous lepton production in e+- e- annihilation”. In: *Physical Review Letters* 35.22 (1975), p. 1489.

[39] K. et al. Kodama. “Observation of tau neutrino interactions”. In: *Physics Letters B* 504.3 (2001), pp. 218–224.

[40] Murray Gell-Mann. “A schematic model of baryons and mesons”. In: *Physics Letters* 8.3 (1964), pp. 214–215.

[41] Moo-Young Han and Yoichiro Nambu. “Three-triplet model with double SU (3) symmetry”. In: *Physical Review* 139.4B (1965), B1006.

[42] Jean-Jacques Aubert et al. “Experimental observation of a heavy particle J”. In: *Physical Review Letters* 33.23 (1974), p. 1404.

[43] J-E Augustin et al. “Discovery of a Narrow Resonance in e+ e- Annihilation”. In: *Physical Review Letters* 33.23 (1974), p. 1406.

[44] Leon M Lederman. “Upsilon particle”. In: *Scientific American* 239.4 (1978), pp. 72–80.

[45] CDF Collaboration. “Observation of top quark production in p p collisions with the Collider Detector at Fermilab”. In: *Physical review letters* 74.14 (1995), p. 2626.

[46] S Abachi et al. “Search for High Mass Top Quark Production in p p^- Collisions at s= 1.8 TeV”. In: *Physical Review Letters* 74.13 (1995), p. 2422.

[47] R Brandelik et al. “Evidence for planar events in e+ e- annihilation at high energies”. In: *Physics Letters B* 86.2 (1979), pp. 243–249.

[48] FJ et al. Hasert. “Observation of neutrino-like interactions without muon or electron in the Gargamelle neutrino experiment”. In: *Nuclear Physics B* 73.1 (1974), pp. 1–22.

[49] G et al. Arnison. “Experimental observation of lepton pairs of invariant mass around 95 GeV/c2 at the CERN SPS collider”. In: *Physics Letters B* 126.5 (1983), pp. 398–410.

[50] G Barbiellini, Burton Richter, and JL Siegrrist. “Radiative Z0 production: a method for neutrino counting in e+ e- collisions”. In: *Physics Letters B* 106.5 (1981), pp. 414–418.

[51] François Englert and Robert Brout. “Broken symmetry and the mass of gauge vector mesons”. In: *Physical Review Letters* 13.9 (1964), p. 321.

[52] Gerald S Guralnik, Carl R Hagen, and Thomas WB Kibble. "Global conservation laws and massless particles". In: *Physical Review Letters* 13.20 (1964), p. 585.

[53] Peter W Higgs. "Broken symmetries and the masses of gauge bosons". In: *Physical Review Letters* 13.16 (1964), p. 508.

[54] Michael E Peskin, Daniel V Schroeder, and Emil Martinec. *An introduction to quantum field theory*. 1996.

[55] C Patrignani, Particle Data Group, et al. "Review of particle physics". In: 40.10 (2016).

[56] Y et al. Fukuda. "Evidence for oscillation of atmospheric neutrinos". In: *Physical Review Letters* 81.8 (1998), p. 1562.

[57] Kenneth G Wilson. "Confinement of quarks". In: *Physical Review D* 10.8 (1974), p. 2445.

[58] David JE Callaway and Aneesur Rahman. "Microcanonical ensemble formulation of lattice gauge theory". In: *Physical Review Letters* 49.9 (1982), p. 613.

[59] Jeffrey Goldstone, Abdus Salam, and Steven Weinberg. "Broken symmetries". In: *Physical Review* 127.3 (1962), p. 965.

[60] C Grojean. "Higgs Physics". In: *CERN Yellow Reports* 5 (2016), p. 143.

[61] *Measurements of properties of the Higgs boson decaying into four leptons in pp collisions at sqrts = 13 TeV*. Tech. rep. CMS-PAS-HIG-16-041. Geneva: CERN, 2017. URL: https://cds.cern.ch/record/2256357.

[62] *Measurement of differential fiducial cross sections for Higgs boson production in the diphoton decay channel in pp collisions at* $\sqrt{s} = 13$ TeV. Tech. rep. CMS-PAS-HIG-17-015. Geneva: CERN, 2017. URL: https://cds.cern.ch/record/2257530.

[63] Serguei Chatrchyan et al. "Measurement of the properties of a Higgs boson in the four-lepton final state". In: *Physical Review D* 89.9 (2014), p. 092007.

[64] Serguei Chatrchyan et al. "Study of the mass and spin-parity of the Higgs boson candidate via its decays to Z boson pairs". In: *Physical Review Letters* 110.8 (2013), p. 081803.

[65] Howard Georgi and A Pais. "Calculability and naturalness in gauge theories". In: *Physical Review D* 10.2 (1974), p. 539.

[66] Douglas Clowe et al. “A direct empirical proof of the existence of dark matter”. In: *The Astrophysical Journal Letters* 648.2 (2006), p. L109.

[67] Mark Peplow. “Planck telescope peers into primordial Universe”. In: *Nature* (2013).

[68] G Castelo Branco and L Lavoura. “On the addition of vector-like quarks to the standard model”. In: *Nuclear Physics B* 278.3 (1986), pp. 738–754.

[69] Lisa Randall and Raman Sundrum. “Large mass hierarchy from a small extra dimension”. In: *Physical Review Letters* 83.17 (1999), p. 3370.

[70] JG Bransoni et al. “High transverse momentum physics at the large hadron collider”. In: *EPJ direct* 4.1 (2002), pp. 1–61.

[71] CERN. “The CERN accelerator complex. Complexe des accélérateurs du CERN”. In: (July 2016). General Photo. URL: https://cds.cern.ch/record/2197559.

[72] Jean-Luc Caron. “Cross section of LHC dipole. Dipole LHC: coupe transversale.” AC Collection. Legacy of AC. Pictures from 1992 to 2002. May 1998. URL: https://cds.cern.ch/record/841539.

[73] Rossi L. “The LHC main dipoles and quadrupoles toward series production”. In: *IEEE transactions on applied superconductivity* 13.2 (2003), pp. 1221–1228.

[74] M Mathieu et al. “293 K - 1.9 K supporting systems for the Large Hadron Collider (LHC) cryo-magnets”. In: *Adv. Cryog. Eng., A* 43.LHC-Project-Report-163. CERN-LHC-Project-Report-163 (Jan. 1998), 427–434. 9 p. URL: https://cds.cern.ch/record/347116.

[75] CMS Collaboration. “CMS Web Based Monitoring”. In: (Oct. 2016). URL: https://cmswbm.web.cern.ch/cmswbm/cmsdb/servlet/FillReport?FILL=5423.

[76] *Measurement of the inelastic proton-proton cross section at* $\sqrt{s} = 13$ TeV. Tech. rep. CMS-PAS-FSQ-15-005. Geneva: CERN, 2016. URL: https://cds.cern.ch/record/2145896.

[77] CMS Collaboration. “CMS Luminosity - Public Results”. In: (Nov. 2016). URL: https://twiki.cern.ch/twiki/bin/view/CMSPublic/LumiPublicResults#2016_Proton_Proton_13_TeV_Collis.

[78] Fabienne CERN Graphic Design service Marcastel. "CMS Tent Point 5 Cessy . Tente CMS point 5 Cessy". Sept. 2013. URL: http://cds.cern.ch/record/1622198.

[79] CMS collaboration. "Particle-flow reconstruction and global event description with the CMS detector". In: *Journal of Instrumentation* (submitted 2017).

[80] CMS Collaboration. *CMS, the magnet project: Technical design report*. Tech. rep. CERN-LHCC-97-10, 1997.

[81] CMS Collaboration. "Description and performance of track and primary-vertex reconstruction with the CMS tracker". In: *Journal of Instrumentation* 9.10 (2014), P10009.

[82] CMS Collaboration. "CMS Tracker Technical Design Report". In: *CERN/LHCC* 6 (1998), p. 1998.

[83] CMS Collaboration. "Addendum to the CMS Tracker TDR". In: *CERN/LHCC* 16 (2000), p. 2000.

[84] Gunnar Lindström. "Radiation damage in silicon detectors". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 512.1 (2003), pp. 30–43.

[85] MJ et al French. "Design and results from the APV25, a deep sub-micron CMOS front-end chip for the CMS tracker". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 466.2 (2001), pp. 359–365.

[86] CMS Collaboration. *The CMS electromagnetic calorimeter project: Technical Design Report*. Geneva: CERN, 1997. URL: https://cds.cern.ch/record/349375.

[87] CMS Collaboration. "Energy calibration and resolution of the CMS electromagnetic calorimeter in pp collisions at sqrt (s)= 7 TeV". In: *arXiv preprint arXiv:1306.2016* (2013).

[88] CMS Collaboration. *The CMS hadron calorimeter project: Technical Design Report*. Geneva: CERN, 1997. URL: https://cds.cern.ch/record/357153.

[89] CMS Collaboration. "The CMS barrel calorimeter response to particle beams from 2 to 350 GeV/c". In: *Journal of Physics: Conference Series*. Vol. 160. 1. IOP Publishing. 2009, p. 012056.

[90] CMS Collaboration. *The Muon Project, CMS Technical Design Report, CERN/LHCC 97-32*. Tech. rep. CMS-TDR-003.

[91] CMS Collaboration. "The performance of the CMS muon detector in proton-proton collisions at $\sqrt{s} = 7$ TeV at the LHC". In: *Journal of Instrumentation* 8.11 (2013), P11002.

[92] CMS Collaboration. "CMS: The TriDAS project. Technical design report, Vol. 2: Data acquisition and high-level trigger". In: *Rapport technique* (2002).

[93] CMS Collaboration. "CMS technical design report for the level-1 trigger upgrade". In: *CMS Technical Design Report CERN-LHCC-2013-011, CMS-TDR-12* (2013).

[94] Perrotta A. "Performance of the CMS High Level Trigger". In: *Journal of Physics: Conference Series*. Vol. 664. 8. IOP Publishing. 2015, p. 082044.

[95] CMS Collaboration. *CMS Computing Model: The "CMS Computing Model RTAG"*. Tech. rep. CMS-NOTE-2004-031. CERN-LHCC-2004-035. LHCC-G-083. Geneva: CERN, Dec. 2004. URL: http://cds.cern.ch/record/814248.

[96] WLCG. "WLCG REBUS: REsource, Balance, USage". Feb. 2017. URL: https://wlcg-rebus.cern.ch/apps/topology/.

[97] Amazon Elastic Compute Cloud. "Amazon web services". In: *Retrieved November* 9 (2011), p. 2011.

[98] *TOP500 Supercomputer Site*. URL: http://www.top500.org.

[99] Facebook. "Scaling the Facebook data warehouse to 300 PB". Apr. 2014. URL: https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/.

[100] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. "PYTHIA 6.4 physics and manual". In: *Journal of High Energy Physics* 2006.05 (2006), p. 026.

[101] Gennaro Corcella et al. "HERWIG 6: an event generator for hadron emission reactions with interfering gluons (including supersymmetric processes)". In: *Journal of High Energy Physics* 2001.01 (2001), p. 010.

[102] Johan Alwall et al. "MadGraph 5: going beyond". In: *Journal of High Energy Physics* 2011.6 (2011), pp. 1–40.

[103] Tanju Gleisberg et al. "Event generation with SHERPA 1.1". In: *Journal of High Energy Physics* 2009.02 (2009), p. 007.

[104] Michelangelo L Mangano et al. "ALPGEN, a generator for hard multiparton processes in hadronic collisions". In: *Journal of High Energy Physics* 2003.07 (2003), p. 001.

[105] S. et al Agostinelli. "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.

[106] CMS Collaboration. *Particle-flow event reconstruction in CMS and performance for jets, taus and MET*. Tech. rep. CMS-PAS-PFT-09-001, 2009.

[107] CMS collaboration. "Performance of electron reconstruction and selection with the CMS detector in proton-proton collisions at sqrt (s)= 8 TeV". In: *arXiv preprint arXiv:1502.02701* (2015).

[108] Wolfgang 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 (2005), N9.

[109] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "The anti-k_t jet clustering algorithm". In: *JHEP* 04 (2008), p. 063. DOI: 10.1088/1126-6708/2008/04/063. arXiv: 0802.1189 [hep-ex].

[110] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "FastJet User Manual". In: *Eur. Phys. J.* C72 (2012), p. 1896. DOI: 10.1140/epjc/s10052-012-1896-2. arXiv: 1111.6097 [hep-ph].

[111] CMS Collaboration. "Jet energy scale and resolution in the CMS experiment in pp collisions at 8 TeV". In: *JINST* 12 (2017), P02014. DOI: 10.1088/1748-0221/12/02/P02014. arXiv: 1607.03663 [hep-ex].

[112] *Performance of missing energy reconstruction in 13 TeV pp collision data using the CMS detector*. Tech. rep. CMS-PAS-JME-16-004. Geneva: CERN, 2016. URL: http://cds.cern.ch/record/2205284.

[113] G. Petrucciani, A. Rizzi, and C. Vuosalo. "Mini-AOD: A New Analysis Data Format for CMS". In: *J. Phys. Conf. Ser.* 664.7 (2015), p. 072052. DOI: 10.1088/1742-6596/664/7/072052.

[114] "The European Strategy for Particle Physics Update 2013. La stratégie européenne pour la physique des particules Mise à jour 2013. 16th Session of European Strategy Council". In: (May 2013). URL: https://cds.cern.ch/record/1567258.

[115] L Arnaudon et al. *Linac4 technical design report*. Tech. rep. 2006.

[116] A. Tricomi. "Upgrade of the CMS tracker". In: *Journal of Instrumentation* 9.03 (2014), p. C03041.

[117] A-M. Magnan. "HGCAL: a High-Granularity Calorimeter for the endcaps of CMS at HL-LHC". In: *Journal of Instrumentation* 12.01 (2017), p. C01042.

[118] CMS collaboration. "CMS technical design report for the pixel detector upgrade". In: *CMS Technical Design Report CERN-LHCC-2012-016, CMS-TDR-11* (2012).

[119] CMS Collaboration. "Replacement of the heart of the CMS experiment - the Pixel detector." In: (Feb. 2017). General Photo. URL: https://cds.cern.ch/record/2253519.

[120] Gavril A Giurgiu et al. "Pixel Hit Reconstruction with the CMS Detector". In: *arXiv preprint arXiv:0808.3804* (2008).

[121] M Swartz et al. "A new technique for the reconstruction, validation, and simulation of hits in the CMS Pixel Detector". In: *PoS* Vertex 2007.CMS-NOTE-2007-033 (July 2007), 035. 37 p. URL: https://cds.cern.ch/record/1073691.

[122] CMS Collaboration. *Tracker DPG public results*. https://twiki.cern.ch/twiki/bin/view/CMSPublic/DPGResultsTRK. 2016.

[123] Ting Miao et al. *Beam position determination using tracks*. Tech. rep. CERN-CMS-NOTE-2007-021, 2007.

[124] Are Strandlie and W Wittek. *Propagation of covariance matrices of track parameters in homogeneous magnetic fields in CMS*. Tech. rep. CERN-CMS-NOTE-2006-001, 2006.

[125] Rudolf 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* 262.2-3 (1987), pp. 444–450.

[126] Rudolf Frühwirth and Meinhard Regler. *Data analysis techniques for high-energy physics*. Vol. 11. Cambridge University Press, 2000.

[127] "High pt jets tracking". In: (Aug. 2015). URL: https://twiki.cern.ch/twiki/bin/view/CMSPublic/HighPtTrackingDP.

[128] CMS Collaboration. *HLT Timing plots*. https://twiki.cern.ch/twiki/bin/view/CMSPublic/HLTplotsCHEP2016. 2016.

[129] Mia Tosi. "Performance of Tracking, b-tagging and Jet/MET reconstruction at the CMS High Level Trigger". In: *Journal of Physics: Conference Series*. Vol. 664. 8. IOP Publishing. 2015, p. 082055.

[130] M Rovere. "CMS reconstruction improvements for the tracking in large pileup events". In: *J. Phys.: Conf. Ser.* 664.7 (2015), 072040. 8 p. URL: http://cds.cern.ch/record/2134627.

[131] Shekhar Borkar. "Design challenges of technology scaling". In: *IEEE micro* 19.4 (1999), pp. 23–29.

[132] I PRESENT. "Cramming more components onto integrated circuits". In: *Readings in computer architecture* 56 (2000).

[133] Sverre Jarp, Alfio Lazzaro, and Andrzej Nowak. "The future of commodity computing and many-core versus the interests of HEP software". In: *Journal of Physics: Conference Series*. Vol. 396. 5. IOP Publishing. 2012, p. 052058.

[134] Robert H Dennard et al. "Design of ion-implanted MOSFET's with very small physical dimensions". In: *IEEE Journal of Solid-State Circuits* 9.5 (1974), pp. 256–268.

[135] Mark Bohr. "A 30 year retrospective on Dennard's MOSFET scaling paper". In: *IEEE Solid-State Circuits Society Newsletter* 12.1 (2007), pp. 11–13.

[136] Anantha P Chandrakasan et al. "Optimizing power using transformations". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 14.1 (1995), pp. 12–31.

[137] Gene M Amdahl. "Validity of the single processor approach to achieving large scale computing capabilities". In: *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM. 1967, pp. 483–485.

[138] Wikipedia. *Amdahl's law*. 2017. URL: https://en.wikipedia.org/wiki/Amdahl%5C%27s_law.

[139] John L Gustafson. "Reevaluating Amdahl's law". In: *Communications of the ACM* 31.5 (1988), pp. 532–533.

[140] Michael Gschwind. "I/O virtualization and system acceleration in POWER8". In: *Hot Chips 27 Symposium (HCS), 2015 IEEE*. IEEE. 2015, pp. 1–26.

[141] CUDA Nvidia. "Compute unified device architecture programming guide". In: (2007).

[142] John E Stone, David Gohara, and Guochun Shi. "OpenCL: A parallel programming standard for heterogeneous computing systems". In: *Computing in science & engineering* 12.1-3 (2010), pp. 66–73.

[143] "GP100 Pascal Whitepaper - Nvidia". In: (2016). URL: https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf.

[144] CD Jones et al. "Using the CMS threaded framework in a production environment". In: *Journal of Physics: Conference Series*. Vol. 664. 7. IOP Publishing. 2015, p. 072026.

[145] Felice Pantaleo and Marco Sozzi. "Development of a parallel trigger framework for rare decay searches". Presented 2013. PhD thesis. Pisa U., 2013. URL: https://cds.cern.ch/record/1636892.

[146] P Buncic, M Krzewicki, and P Vande Vyvre. *Technical Design Report for the Upgrade of the Online-Offline Computing System*. Tech. rep. CERN-LHCC-2015-006. ALICE-TDR-019. Apr. 2015. URL: http://cds.cern.ch/record/2011297.

[147] Sergey Gorbunov et al. "ALICE HLT high speed tracking on GPU". In: *IEEE Transactions on Nuclear Science* 58.4 (2011), pp. 1845–1851.

[148] D. H. Campora Perez et al. "The 40 MHz trigger-less DAQ for the LHCb Upgrade". In: *Nucl. Instrum. Meth.* A824 (2016), pp. 280–283. DOI: 10.1016/j.nima.2015.10.047.

[149] G Amadio et al. "The GeantV project: preparing the future of simulation". In: *Journal of Physics: Conference Series*. Vol. 664. 7. IOP Publishing. 2015, p. 072006.

[150] Bjarne Stroustrup. *The C++ programming language*. Pearson Education India, 1995.

[151] Rene Brun and Fons Rademakers. "ROOT—an object oriented data analysis framework". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 389.1-2 (1997), pp. 81–86.

[152] Maarten Ballintijn et al. "Parallel interactive data analysis with PROOF". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 559.1 (2006), pp. 13–16.

[153] R Andreassen et al. "GooFit: A library for massively parallelising maximum-likelihood fits". In: *Journal of Physics: Conference Series*. Vol. 513. 5. IOP Publishing. 2014, p. 052003.

[154] Donald Ervin Knuth. *The art of computer programming*. Pearson Education, 1998.

[155] Felice Pantaleo. *FKDTree*. https://github.com/felicepantaleo/FKDTree. 2016.

[156] John Von Neumann. *The Theory of Self-reproducing Automata*. 1966.

[157] John Conway. "The game of life". In: *Scientific American* 223.4 (1970), p. 4.

[158] Virgil L Highland. "Some practical remarks on multiple scattering". In: *Nuclear Instruments and Methods* 129.2 (1975), pp. 497–499.

[159] Message Passing Interface Forum. *MPI2: A message passing interface standard*. 1991.

[160] NVIDIA. *Developing a Linux Kernel Module using GPUDirect RDMA*. http://docs.nvidia.com/cuda/gpudirect-rdma/index.html. 2016.

[161] F Pantaleo et al. "Development of a phase-II track trigger based on GPUs for the CMS experiment". In: *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2015 IEEE*. IEEE. 2015, pp. 1–6.

[162] CMS Collaboration. *DAQ interim document for the Phase-2 upgrade of the CMS detector*. 2017.

[163] P.J. Plauger et al. *C++ Standard Template Library*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000. ISBN: 0134376331.

[164] CMS Collaboration. *CMS Tracking POG Performance Plots For 2017 with Phase-1 pixel detector*. https://twiki.cern.ch/twiki/bin/view/CMSPublic/TrackingPOGPerformance2017MC. 2017.

Acknowledgements

The work described in this thesis was sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research.

I would like to express my sincere gratitude to my supervisor Dr. Alexander Schmidt, who, with passion and indefatigable diligence, has always set the bar higher and higher.

I would like to express my special appreciation to Dr. Vincenzo Innocente for being always my Pole star, always there to guide my education and research with his deep knowledge of the subjects and with the kindness of a mentor.

I would like to show my greatest appreciation to Dr. Andreas Pfeiffer. I feel motivated and encouraged every time I explain my doubts and problems to him.

Completing this work would have been much more difficult were it not for the support and friendship provided by Dr. Benedikt Hegner.

I would like to warmly thank Dr. Andreas Meyer for the guidance and supervision.

I would also like to thank Dr. Marco Rovere and Dr. Andrea Bocci for always supporting my ideas and for the experience they have helped me gain by working with them at CERN.

I would like to extend my heartfelt thanks to Dr. John Harvey. His door has been always open to guide and support my ideas.

I am very grateful to Dr. Matti Kortelainen for his scientific advice, knowledge and many discussions.

I would like to express my gratitude to Dr. Frans Meijers and Dr. Emilio Meschi for their advices and fruitful discussions.

I gratefully thank Dr. Tiziano Camporesi, Dr. Luca Malgeri, Prof. Roberto Carlin, Dr. Tommaso Boccali, Dr. Gianluca Cerminara, Dr. Giovanni Franzoni, Dr. Simone Gennai, Dr. Arabella Martelli, Dr. Giovanni Petrucciani, Dr. Maurizio Pierini, Dr. Lucia Silvestris and Dr. Mia Tosi for making CERN feel like home.

I will forever be thankful to Dr. Alfio Lazzaro for always being a good friend and for teaching me how to follow The Code in the first place.

A very special thanks goes out to Dr. Domenico Giordano, without whose encouragement and motivation my academic career would have probably followed a different path.

I wish to express my love and gratitude to my beloved family for their infinite understanding and endless love, protection and support throughout the duration of my life.

A special acknowledgement goes to Isabel, who loved and supported me during these years.

I would like to express my deepest gratitude to Nicola for being like a brother to me during all the time spent in Geneva.

I am very grateful to my friend Silvio, although he did not thank anyone.

I wish to thank my friends in Geneva: Noemi, Francesco, Mirko, Maddalena, Ruggero, Alessandro, Alessandra, Nike. You have been like a second family to me. Thank you, this work would have been much harder to complete without you.

I wish to express my gratitude to my best friends Anna, Fabio, Felice, Giuseppe, Marco, Mariele, Mary and Vito. Although we meet only once a year, our friendship strengthens day by day.

I would like to thank the students I have supervised during the last three years: Adriano, Alessandro, Ann-Christine, Antonio, Dominik, Jean-Loup, Konstantinos, Kunal, Luca, Panos, Roberto, Romina, Simone, Somesh. Thank you for teaching me so much.

I would also like to thank Emanuele, Ivan and Mattia for making the time spent in Hamburg so wonderful.

Finally, I would like to thank Cesare, Emilio, Giacomo, Luca, and Tommaso for almost fifteen years of conspiracy and many secrets encoded in this thesis text.