Calibrating and Recommissioning the ALICE Transition Radiation Detector

Jason Barrella

University of Cape Town

supervised by
A. Prof. Thomas Dietel

2023

Submitted as part of the requirements of a Masters degree in Physics.

Abstract

Starting from the year 2022, the LHC produces a factor 6 increase in the interaction rate of Pb-Pb collisions from 8 to 50 kHz, offering exciting new avenues of research in the field of heavy-ion physics. Changes to the ALICE experiment include the introduction of an entirely new software framework as well as an approximate 100 fold increase in statistics and significant improvements in tracking capabilities which will further expand the realm of potential research. The work presented here consists of three primary parts. Firstly, in preparation for new detector control systems at CERN, core software for the ALICE Transition Radiation Detector was updated and migrated to a new version control system. Pipelines were also developed to automatically deploy packages to remote repositories. Secondly, drift velocity and $E \times B$ calibration software was developed and tested on Run 2 data. A mean angular resolution of 2.6° was achieved. This was found to be within a range of a previous result that could be plausibly explained by a number of factors. Finally, components were written in the new ALICE O^2 software framework with applications in calibration, TRD geometry transformations, and a new TRD event display. All were tested successfully and the event display was deployed to the new TRD web page.

Contents

List of Figures

Chapter 1

Introduction

1.1 Particle Physics and the Standard Model

Particle physics involves the study of the properties of elementary particles. It concerns itself with elucidating the nature of interactions within the Standard Model and continuously probing deeper for truly fundamental physics. Originally a pursuit of philosophy, ancient thinkers like Democritus (ca. 460 BC) first postulated the existence of tiny fundamental particles which formed the building blocks of all visible matter. The name given to this theoretical particle was "atom" from the Greek *atomos* meaning "indivisible". Over the millennia, atomic theory has been developed by many great thinkers and physicists each contributing crucial insights and discoveries to the field. Considered to be one of the founders of modern chemistry, Robert Boyle (1627 - 1691), first argued that the chemical properties of a distinct material object were determined by the configuration of its atomic constituents [1]. John Dalton (1766 - 1844) further expanded on Boyle's work and was the first to formally incorporate atomic theory into chemistry. During this period, it was still believed that atoms were fundamental until eventually in 1897, when JJ Thompson (1856 - 1940) discovered the electron and revolutionized over two millennia of understanding concerning the nature of fundamental physics. Realizing that electrons were negatively charged and atoms neutral, he proposed the famous "plum-pudding" atomic model in which negatively charged electrons (plums) wander within the confines of a positively charged volume (pudding) [1]. At this time, it was also discovered that electrons could be liberated from the atom by a significantly high input of energy. Later, after a realization that the positive charge as well as the mass of the atom were heavily concentrated towards its centre via the famous "gold foil" experiment of 1908, Ernest Rutherford (1871 - 1937) discovered the atomic nucleus. With contributions from Eugen Goldstein (1850 - 1930), the proton was also discovered in 1917 and named by Rutherford. Based on these findings, he went on to propose a new atomic model in which electrons follow a planar orbit about the nucleus similar to the way planets orbit the Sun. Neutrons were later discovered in 1932 by James Chadwick (1891 - 1974) [1].

Since the advent of quantum theory, the atom and subatomic particles had to be re-understood in the light of a quantized rather than continuous distribution of energy (Planck, Bohr) and an inexorable limit to the accuracy of any measurement of position or momentum (Heisenberg). Furthermore, electrons should be thought of as occupying fuzzy, uncertain three-dimensional shells rather than circular orbits around the nucleus. A vast array of new fundamental particles have been discovered and studied since the top-level structure of the atom was gradually uncovered during the 19th century. An overview can be seen in Figure 1.1. By modern understanding, any particle should be properly understood as a collapsed wave function described by the appropriate quantum field theory. The discovery of the

quantum-mechanical wave function by Erwin Schrödinger in 1926 was one of the principal achievements of early quantum theory. Named after its discoverer, the Schrödinger equation was further generalized for relativistic effects by Paul Dirac in 1928 to obtain, in natural units,

$$i\gamma^{\mu}\partial_{\mu}\psi - m\psi = 0. \tag{1.1}$$

The Dirac equation describes the evolution of a particle's temporal-spatial four vector and is valid for all fermions (see Figure 1.1). According to the Copenhagen interpretation, the wave function represents a probabilistic description of a particle's position over time and collapses into a particular configuration immediately upon measurement. This final configuration cannot be determined before-hand up to an arbitrary precision. Physicists and philosophers alike have debated how exactly the wave function should be interpreted and why it is that the observer plays such an important role in the process even though the wave equation itself does not explicitly encode it.

The Standard Model is a broad theory describing the properties and interactions of the elementary particles. The particles of the Standard Model can be separated into two primary categories; fermions and gauge bosons. Fermions are characterized by 3 attributes: They all have spin 1/2 and obey the Pauli exclusion principle which forbids any two fermions from occupying identical quantum states in any quantum system. Additionally, they each have a respective anti-particle partner and annihilate, releasing mass energy, upon coming into contact with one-another [2]. Fermions can be further subdivided into quarks and leptons. According to the Standard Model, quarks are the most fundamental constituents of matter. Baryons include hadrons, which are particles consisting of 3 quarks, for example protons and neutrons, and mesons, which consist of 2 quarks. Quarks come in three generations and six flavours varying in rest-mass from 2.2 MeV/c^2 for the up quark to 173.1 GeV/c^2 for the massive top quark. Among fermions, they are uniquely characterized by their possession of colour charge and consequent susceptibility to the strong force. Additionally, they also posses both electrical and weak charge. On the other hand, leptons, excluding neutrinos, poses only electrical charge and, therefore, experience the electroweak forces [2].

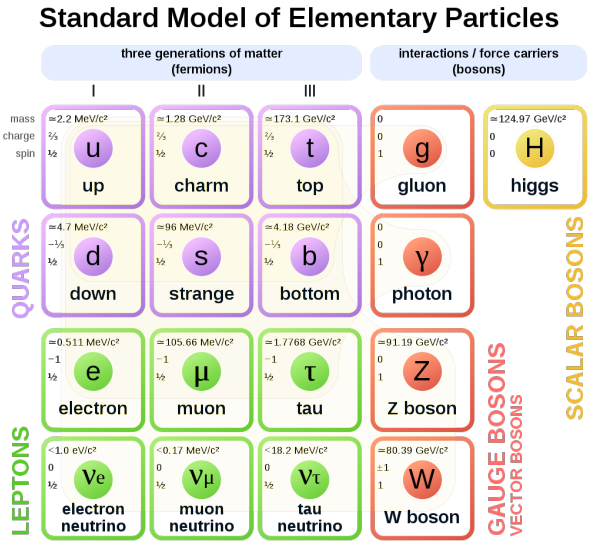

Figure 1.1: The Standard Model of particle physics [3].

Bosons are particles with integer spin and, with the exception of the Higgs boson, mediate forces between other elementary particles. They consist of the gluon, which is responsible for the strong force between quarks, and the photon and Z^0, $W^{\pm}$ bosons which are responsible for the electroweak forces. The Higgs boson was introduced to the Standard Model after its discovery in 2012 and is the mediator of the Higgs field which gives elementary particles their mass [4]. The Standard Model is yet to incorporate gravity and this is currently described by the theory of general relativity. Ongoing efforts in quantum gravity seek to reconcile general relativity with quantum mechanics and thereby introduce gravity to the Standard Model.

1.1.1 Quantum Chromo-Dynamics

In addition to the interactions of individual elementary particles, one may also study the properties of high-energy nuclear interactions, a field of study which has only become viable in the modern age of particle accelerators. One particular area of interest is the study of the Quark-Gluon Plasma (QGP). The QGP is an extremely energy dense phase of matter characterized by the free movement of quarks and gluons within a very small volume of space. It is understood to have filled the early universe a few microseconds after the big bang before cooling and forming hadronic matter. The strong force is extremely powerful compared to the other four fundamental forces and can only be overcome by the input of considerable amounts of energy on the TeV scale. Take for example, the electromagnetic force which has a coupling constant of $\alpha = 1/137$ and compare this to the strong force with $\alpha_s \approx 1$. The quantum field theory that describes the behaviour of quarks and gluons (collectively referred to as partons) is called Quantum Chromo-Dynamics (QCD). Two phenomena of QCD are of particular importance to the field of experimental heavy-ion physics. The first is called confinement and refers to the principle according to which quarks remain bound inside colour-neutral hadrons [5]. Baryons consist of one quark of each possible colour that is red, green, and blue (rgb) or anti-red, anti-green, and anti-blue ($\bar{r}\bar{g}\bar{b}$), while mesons will consist of a quark anti-quark pair for example $b\bar{b}$. In all three cases, the net colour charge is 0 or "white". This phenomenon may more specifically be called colour-confinement. A common potential used to describe quark systems is the Cornell potential [5]. It takes the form

$$V(r) = \frac{-A(r)}{r} + Kr. \tag{1.2}$$

Here, the first term arises from one-gluon interactions between quark pairs and takes a Coulombic form with a $1/r$ dependence. The second term is referred to as the confinement term and relates to gluon-gluon interactions. It increases linearly with respect to r. The constant K has been measured to be $K \approx 880$ MeV/fm [5]. This can be seen to be remarkably large when considering that the nucleic scale is on the order of 1 fm. This immense confinement energy is also the reason that the rest-mass of a baryon far exceeds the rest-mass of its constituent quarks. It can be shown that the source of 99% of nuclear rest-mass is due to the confinement energy [6]. One may take for example the proton which has a rest-mass of nearly 1 GeV although it is composed of 2 up quarks and a down quark whose combined rest-mass is only approximately 10 MeV. What this means is that the amount of energy required to move a quark a distance of 1 fm away from its confinement volume is far greater than the pair production energy of a new $q\bar{q}$ pair. Recall that the up, down, and strange quarks all have rest-masses of < 100 MeV/c^2. Therefore, any attempt to extract a quark from its confined volume does not result in a free quark, but rather the production of a new particle.

However, QCD also describes the experimentally confirmed phenomenon called asymptotic freedom which refers to the decrease in the coupling strength of the strong interaction as a function of momentum transfer.

1.2 Heavy-Ion Physics

As a result of asymptotic freedom, given an environment of sufficient temperature in the realm of 2 terakelvin, it is possible to separate a quark system [6] enough such that scatterings appear as the interactions with free quarks even though the quarks themselves are still strongly bound to one-another. It is according to this property that quarks may be deconfined from hadrons in high-energy collisions.

During a heavy-ion collision, bunches of nuclei are accelerated up to relativistic energies at which point they take the form of Lorentz contracted discs. The discs approach each other head-on from opposite directions. In the case of lead nuclei, the diameter of each disc is approximately 14 fm with a γ factor in the region of 2500 for collisions at the LHC [7]. Largely as a consequence of the extreme Lorentz contraction in beam direction, the participating region where the two discs overlap upon collision reaches an extremely high energy density of over 12 GeV/fm^3 [7]. This is roughly 20 times the energy density of a hadron [7] and is enough to take advantage of QCD's property of asymptotic freedom. About 1 fm/c after the collision, a significant number of new partons have been produced via strong interactions as well as hard collisions between the participating matter. These partons remain coupled to each other and begin expanding as a hydrodynamic fluid. This is the QGP. Typical temperatures of the QGP at the LHC are on the order of 10^{12} K [8]. The fluid continues to expand under its own immense pressure until about 10 fm/c at which point it has cooled enough such that the local energy density begins to drop below that of a typical hadron [7]. At this point, hadronization occurs as the partons recombine into colourless pairs and triplets. Still moving at relativistic velocities, the resulting particles fly out in all directions. Because of this, direct observations of the particles comprising the QGP are not possible. Instead, properties of the QGP must be inferred from the resulting hadrons and other radiated particles. These particles themselves are often unstable and will decay further before reaching any detectors. They will then have to be reconstructed during the analysis phase based on knowledge about the final state particles. Other sources of information about the QGP include probes such as jets which are created by particles with high p_T that are produced during the initial collision [7]. The jets are attenuated by the QGP as they propagate through it by a process known as jet quenching and therefore contain information about properties like energy density.

1.3 CERN and ALICE

The European Organization for Nuclear Research (CERN[1]) is located on the outskirts of Geneva, Switzerland and houses the largest particle physics laboratory in the world. Funded by 23 member states and working with over 10 000 scientists and engineers across the globe, CERN has been on the forefront of high-energy physics research for the past several decades since it was founded in 1954. The facility operates a total of 7 particle accelerators. The most widely known is the Large Hadron Collider (LHC). The LHC is a 27.2 km tunnel buried 100 m below the surface that straddles the border between Switzerland and France. It is

[1]The acronym "CERN" is derived from the French *Conseil Européen pour la Recherche Nucléaire* which may be translated in English as the European Organization for Nuclear Research.

the largest and highest-energy particle accelerator in the world and can presently produce a maximum combined collision energy of 13.5 TeV. Inside the LHC, protons can achieve velocities of up to 99.99999% the speed of light at which point they circle the LHC ring approximately 11 000 times a second. At these energies, collisions offer a window into new physics and are able to produce massive unstable particles such as the Higgs boson whose existence was predicted by Peter Higgs in 1964 and experimentally confirmed by the ATLAS and CMS collaborations at CERN in 2012 [9]. These exotic particles cannot be observed easily anywhere else in nature and therefore accelerators like the LHC offer a unique opportunity for research in fundamental physics.

Before arriving at the LHC, particles are initially injected into a linear accelerator called Linac4. For a proton beam, a hydrogen gas sample is used and for a lead beam, the particles are taken from a 500 mg sample of ^{208}Pb [10]. The bunches of atoms are gradually stripped of all electrons so as to gain a net positive charge and are then accelerated by a powerful electric field. After Linac4, the particles are consecutively passed through three synchrotrons ranging from 157 m to 7 km in circumference [11]. In each synchrotron, the bunches gradually gain energy as they are further accelerated via timed electric fields and curved by magnets situated along the circumference of the beam pipes. Eventually, they are fed into the LHC where they achieve their maximum velocity inside two separate pipes of opposing beam directions. The bunches may then be collided in any of the four detectors caverns known as Point 1 (P1), P2, P5, and P8 respectively where each cavern is named after the LHC octant that it is located in (see Figure 1.2).

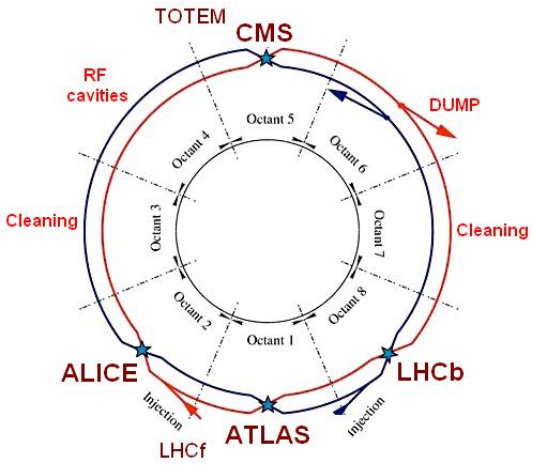

Figure 1.2: Schematic of the LHC showing all 8 octants. Octants are labelled in a clockwise direction starting from the southernmost segment. Injection points for both beam pipes are shown toward the south in octants 2 and 8. The four collision points are labelled with stars and the names of the respective experiments are shown [12].

A Large Ion Collider Experiment (ALICE) is one of the 4 primary experiments situated on the LHC ring. It is located at P2 just across the border in the French commune of Saint-Genis-Pouilly. The chief objective of ALICE is to study the physics of strongly interacting matter, particularly the QGP via collisions of heavy ions (Pb-Pb, p-Pb) at ultra-relativistic energies. Inside super-colliders like the LHC is the only place in the universe where free quarks are currently known to be observable [6].

Within the ALICE reference frame, coordinate systems in Cartesian and cylindrical coordi-

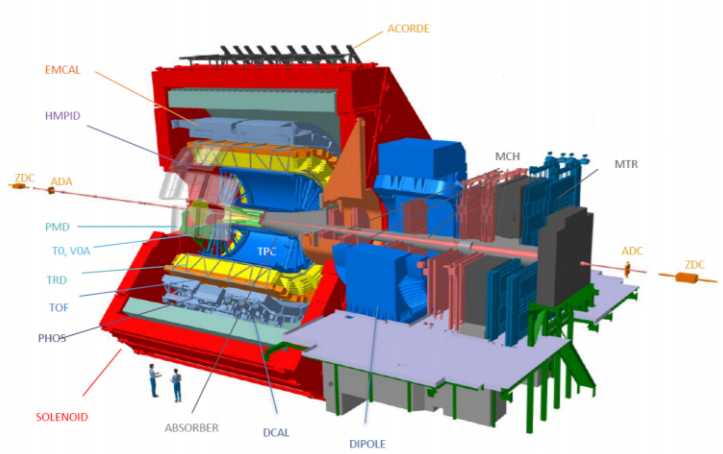

Figure 1.3: Schematic of the ALICE experiment during Run 3. Each component and detector is labelled. The beam axis passing through the centre of the detector is shown in red.

nates are commonly used. In both systems, the origin is located at the beam's interaction point which is defined as the point of collision [13]. The degrees of freedom in each coordinate system are described as follows:

Cartesian

- x - Aligned with the local horizontal in the plane of the LHC ring and pointing towards the centre of the LHC.
- y - Points vertically upwards towards the surface of the ground.
- z - Parallel to the beam axis and pointing away from the muon tracker (shown in blue on the right-hand side of Figure 1.3.

Spherical

- azimuthal ϕ - In the x-y plane. Increases counter-clockwise in the direction from the x-axis ($\phi = 0$) to the y-axis ($\phi = \pi/2$) from the perspective of an observer standing at positive z and looking toward the muon tracker.
- polar θ - Measured with respect to the z-axis. Increases in the direction from the z-axis ($\theta = 0$) to either the y-axis ($\theta = \pi/2$) or x-axis (also $\theta = \pi/2$).
- r - Increases proportionally to the component perpendicular to the beam axis. Explicitly, $r = \sqrt{x^2 + y^2 + z^2}$.

ALICE is sometimes referred to as a detector although in reality, it is better understood as a combined whole of many detectors that each serve a specific function. In total, ALICE is 26 m long, 16 m high, and 16 m wide[2], and weighs approximately 10 000 tonnes making it one of the largest detector systems ever built. It comprises 13 detectors which are, in general, arranged as concentric barrels although not all of them occupy the full azimuthal range. Beam particles arrive along the central beam axis shown in red in Figure 1.3. From the interaction point, particles produced in the collision event move outwards with a radial component

[2]Length is measured in z, height in y, and width in x.

and pass through some number of detectors sequentially. The actual distance that the particle travels will depend on its type and various physical properties like momentum and charge.

In order to study the QGP in a meaningful way, ALICE attempts to accurately describe each collision event to the level of the individual particle. The entire ALICE detector system is immersed in an intense 0.5 T magnetic field produced by the solenoid shown in bright red in Figure 1.3. Via the Lorentz Force, the magnetic field results in a deflection of charged particles in ϕ, the plane that is perpendicular to the beam axis, where the direction depends on the combination of the sign of the charge and the magnetic field itself which is a variable parameter. The degree of deflection is determined by the momentum of the particle. Particles with a higher momentum will curve less, while particles with lower momenta will curve more. This result can be obtained by equating the centrifugal force of a particle on a circular orbit with the Lorentz force. After dividing by the velocity which appears on both sides of the equation, it can be shown that the momentum is proportional to the radius of curvature. For very high momenta, particles will travel in almost straight lines and are sometimes approximated as such during analysis. Once momentum is derived from the trajectory of the particle in the magnetic field, velocity can be determined by measurements from the tracking detectors like the Inner Tracking System (ITS) or Time Projection Chamber (TPC) and then mass can be calculated from both momentum and velocity. Knowing mass and charge helps to identify many of the particles found in the detector and the Time Of Flight (TOF) detector is specialized in identifying charged particles. The process of particle identification (PID) is one of the primary functions of the detector system. The tracking detectors also aid in determining where each particle was produced in order to distinguish between particles originating from the initial collision and secondary particles which result from the decay of others after initial production. The calorimeters in the outer layer of ALICE absorb particles in order to determine their energy.

1.4 The Transition Radiation Detector

The Transition Radiation Detector (TRD), shown in yellow in Figure 1.3 is primarily concerned with the identification of electrons and pions as well as functioning as a trigger[3] for high p_T processes and electrons [14]. The TRD also assists with the calibration of the TPC which is adjacent. In the lower momentum range $p_T < 1$ GeV, the TPC is able to identify electrons by measuring their energy loss. However, at higher p_T it becomes increasingly less effective. The TRD is able to provide accurate electron identification in these higher momenta ranges by exploiting the phenomenon of transition radiation. This is described in more detail later.

The TRD consists of 522 detector chambers[4] and covers the full azimuthal range of 2π as well as a pseudorapidity[5] range of $|\eta| \leq 0.9$. The barrel of TRD chambers is segmented along ϕ into 18 sectors (also referred to as super-modules) which each consist of 30 chambers or 24 in the case of the three sectors which had chambers removed to reduce scattering of particles moving toward the PHOS detector which is positioned behind the TRD. Sector number increases in positive ϕ direction (see Figure 1.4). An individual sector consists of 5 stacks

[3] Triggering is a rapid event-selection system based on simple criteria by which only particular events are read out in order to reduce data load.

[4] "Chamber" and "detector" are often used synonymously with one-another although the technical term for each module is Read Out Chamber (ROC). Formerly, having 540 chambers, and now only 522.

[5] Pseudorapidity is a measure of angle relative to the beam axis. It is defined as $\eta \equiv -\ln\left[\tan\left(\frac{\theta}{2}\right)\right]$. Therefore $\eta = 0.9$ is 44.25°

oriented in z direction. Stack number increases in negative z direction. A stack consists of 6 chambers arranged as layers increasing in r. The radius of the starting edge of the layers with respect to the interaction point varies between 2945 mm for a layer 0 detector and 3575 mm for a layer 5 detector. The width of the layers (in ϕ) becomes larger as one moves radially outwards to account for the increasing circumference of the TRD barrel. Widths vary between 956 mm and 1178 mm and lengths (in z) vary between 1100 mm and 1605 mm [14]. Each TRD chamber is comprised of either 12 or 16 rows of 144 pads. The pad is the read out unit of the TRD (see section 2.1 for more information on detector read out). The precise number of rows depends on the type of chamber. C1 chambers have 16 rows and make up stacks 0, 1, 3, and 4. Stack 2 consists exclusively of C0 chambers which have 12 rows [15]. The rows are oriented along the z axis while the columns are in ϕ. The higher density of pad columns as opposed to pad rows gives the TRD significantly better resolution in ϕ which is the direction along which charged particles bend in the magnetic field. Pads are tilted about their centres (along z) by 2° with respect to the z axis. This effect creates a correlation between the pad column and pad row positions and leads to better resolution in z. The sign of the tilt alternates between layers.

Positions inside a given TRD chamber may also be spoken about in terms of local chamber coordinates. In this coordinate system, the pad rows are along the local z axis and the pad columns along y. z points along the beam line. The x axis points from the radiator toward the pad plane. The origin of the coordinate system is at the centre of the chamber in y and z. This is equivalent to the plane between pads 71 and 72 in y and the plane between pad rows 6 and 7 for C0 chambers or pad rows 8 and 9 for C1 chambers in z. Regarding the origin of x, different conventions within the chamber are used depending on the context, for example at the boundary between the radiator and the drift volume. The local coordinate axes would coincide with the global axes for an imaginary chamber in sector -0.5 as can be seen in Figure 1.4. Another coordinate system that is often used is sector coordinates, or tracking coordinates as it is also known. In this coordinate system, x and y have the same direction and origin as in local chamber coordinates, however the z axis has its origin at the centre of the sector being the centre of stack 2.

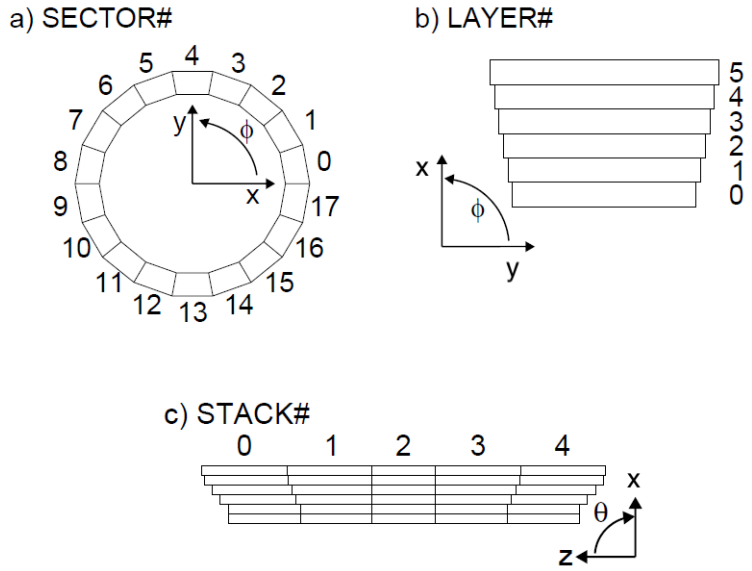

Figure 1.4: Diagram and numbering guide for TRD layers, stacks, and sectors. Coordinates are given in the ALICE global coordinate system. Local chamber coordinate axes can be seen to coincide with the global coordinate system for a chamber in the hypothetical sector -0.5. It is also noted that the TRD is not strictly circular, but rather composed of 18 planes arranged in a circular fashion [16].

Among other components, each chamber includes a radiator and a drift region (see Figure 1.5) filled with Xe + CO_2 gas in a ratio of 85:15 for optimal absorption of transition radiation [14]. Particles originating from the interaction point will pass through the radiator before the drift volume. A charged particle with sufficient velocity will produce transition radiation as it traverses the boundary between two media of different dielectric constants as in the alternating layers of foam and air of the radiator [14]. The emitted radiation is the direct result of energy conservation laws acting on the sudden change in the electromagnetic field of the incident charged particle. The charged particle exhibits a different electromagnetic field depending on the dielectric constant of the medium and the transition radiation is merely the excess energy arising from the change in this field across the collection of radiator boundaries. The emitted energy is proportional to the Lorentz γ factor within a certain range of energies.

As the charged particle exits the radiator and continues its trajectory through the drift region, it ionizes the Xe + CO_2 gas, creating clusters of roughly 275 electrons per cm within its 3 cm high volume [14]. Clusters are also produced in the 0.7 cm amplification region. An electric field is applied across the drift region via an electrode located directly on top of the radiator and kept at a high potential of around -1900 V. The TRD contains a multi-wire proportional drift chamber and therefore uses a plane of cathode wires and a plane of anode wires 7 and 3.5 mm below the top pad plane respectively as shown in Figure 1.6. The pad plane and cathode wires are kept at ground while the anode wires are supplied with a high positive voltage in the region of +1500 V. Because the cathode plane is kept at 0 potential, the drift region is compartmentalized from the strong electric field in the region of space surrounding the anode wires. This causes the electron clusters as a whole to drift at a predictable and constant velocity through the drift chamber. However, once the ionized electrons cross the cathode wire plane, they are suddenly attracted towards the anode wires. The anode wires are designed to be as thin as possible (20 μm) to take full advantage of the inverse square law of the electromagnetic force. As the electrons rapidly approach the

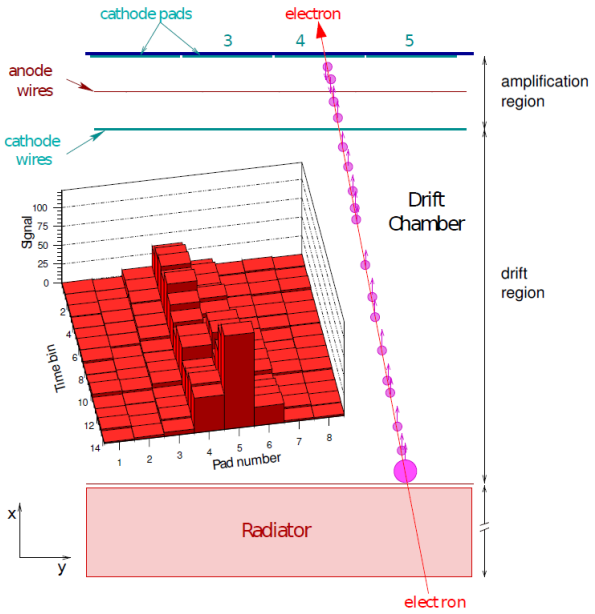

Figure 1.5: Cross-section of a TRD chamber showing the radiator, drift region, and amplification region surrounding the anode wires (top, shown in red) between the cathode wires (light blue) and cathode pads (dark blue). The path of a hypothetical electron is shown as a red line with clusters of ionized electrons shown in pink. The large cluster shown just above the radiator represents the contribution from transition radiation. A three-dimensional plot of the induced signal distribution over time and pad coordinate is overlaid. In the plot, the peak caused by the transition radiation can be seen predominantly in pad 5 at a late time assuming that the clusters drift vertically upwards. Coordinate axes are shown in the local chamber coordinate system. [14].

anode wires, a phenomenon known as a Townsend avalanche takes place whereby energetic electrons collide with and subsequently ionize more atoms which, in-turn, ionize even more atoms and so on. This results in a multiplication of the total current absorbed by the anode wires by a factor of up to 10^4 [14]. For this reason, the region of space between the cathode and pad planes is referred to as the amplification region.

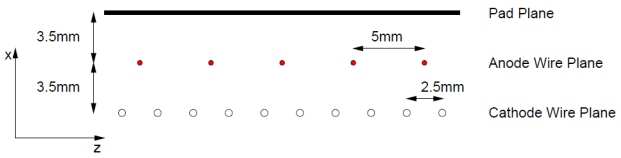

Figure 1.6: Cross-section of the pad plane region at the top of a TRD chamber. Wires are shown to run in the ϕ direction [14].

The electrons that are absorbed by the anode wires gradually dissipate and do not directly produce a signal. Rather, the signal is induced by the motion of the positive ions created in the vicinity of the anode wires toward the pad plane. All the pads in the TRD share a well understood Gaussian pad response function which describes the way the charge is shared between adjacent pads. The magnitude of the induced current is proportional to the proximity of the pad to the positive ion. By comparing the signal strengths in a group of adjacent pads, the TRD is able to achieve sub-pad position resolution. Any transition

radiation that was produced will have been absorbed by the gas in the drift region, thus liberating additional electrons which, in combination with the electrons ionized from the gas, will generate a larger signal. In general, pions produce close to zero transition radiation than electrons because of their higher mass and therefore, lower γ factor [17], as well as lower dE/dx. By comparing the distributions of induced signals over time, a distinction can be made between measurements where transition radiation was and was not present. An example of such distributions can be seen in Figure 1.7 where three different pulse height plots are shown. The distinction is more apparent at higher p_T and it is for this reason that the TRD performs better at PID in these momentum regions.

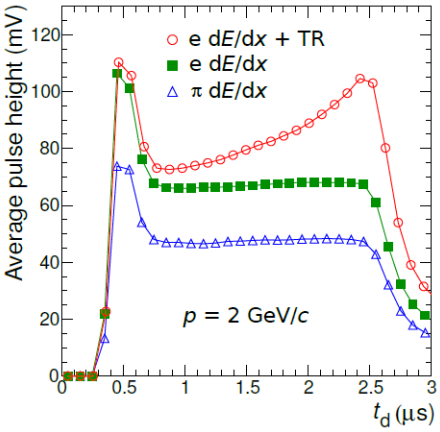

Figure 1.7: Comparison between a TRD-measured signal with and without the presence of transition radiation (TR). In red, a TR photon was produced and therefore a secondary peak is observed at later times. In blue, the TR peak is not present and the signal takes the shape of a plateau. The plateau represents the constant rate of absorption of electrons ionized in the drift region which drift at a near-constant velocity due to a homogeneous electric field. The peak on the other end at lower times is caused by the simultaneous absorption of electrons ionized both above and below the anode plane. [17].

Analysis of the distributions shown in Figure 1.7 give direct information regarding the energy loss over time of a particle traversing the TRD. By examining the length of the plateau region, information regarding the drift velocity of ionized electron clusters can be deduced. A longer plateau indicates a longer time period during which electrons ionized in the drift region were being absorbed and therefore a slower drift velocity. Further information about the trajectory of incident particles can be gathered from the pad positions of the hits.

1.5 ALICE and TRD Recommissioning

1.5.1 CERN Timeline and ALICE Upgrades Overview

CERN runs the LHC periodically. The very first beam was in November 2009 followed by the first collisions in March of 2010 at a beam energy of 7 TeV [18]. Since then, there have been 2 physics runs. The first during the period of 2011 to early 2013 and the second from 2015 to 2018. A projected timeline of commissioning plans for the future is shown in Figure 1.8. After each run, CERN enters a period of shutdown where systems can be maintained and upgraded. This time period is referred to as Long Shutdown (LS). LS1 took place immediately after Run 1 and LS2 is taking place at the time of writing with plans to begin Run 3 in early 2022 and continue until the end of 2023.

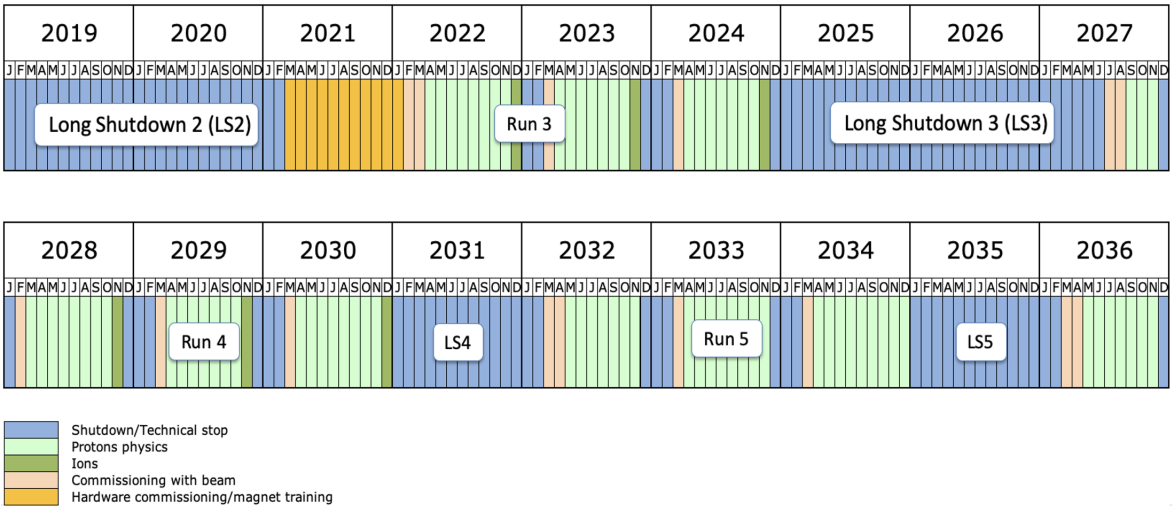

Figure 1.8: Run commissioning timeline for the LHC [19].

For Run 3, a factor 10 increase in the integrated luminosity of Pb-Pb beams is expected at the LHC [20]. In light of this, new physics opportunities arise for ALICE including studies in:

- Heavy-flavour production. Forming one of the primary motivations for the upgrade of the ALICE detector, the study of the production and thermalization of charm and beauty quarks in the QGP will benefit considerably from an increased luminosity and collision energy. These easily-identifiable particles provide information about the evolution and transport properties of the QGP based on their kinematical distributions in momenta and azimuthal angle [21]. Since they are massive, they are less likely to thermalize completely and therefore preserve more information about prior interactions [21]. Additionally, their high mass accommodates easier theoretical modelling by allowing certain approximations to be made [21]. However, production is rare and therefore proper analysis requires large statistics.

- Low-mass dileptons. EM-interacting e^+e^- pairs are emitted during all stages of the formation of the QGP and therefore provide valuable information concerning its evolution [21]. Furthermore, low-mass dilepton spectra are currently the only proposed observable toward investigating the chiral properties of QCD [21]. These measurements will require the significant increase in ALICE tracking performance specifically at low p_T which will come from the upgrades of tracking detectors like the ITS and the TPC. Higher luminosity and therefore more statistics is a further aid.

- Charmonia at low p_T toward a better understanding of the QGP deconfinement phase [22]. The strongly bound charm - anti-charm quark pair quarks are entangled and their states and wave functions are well understood. The J/ψ meson is the most common form of charmonium. According to deconfinement, these bound states are disallowed in the QGP and therefore J/ψ mesons should not be found. This effect is known as J/ψ suppression. However, in the initial collision, many J/ψ pairs are created. Therefore, the suppression of the J/ψ may be studied as a function of QGP temperature and a better understanding of suppression can be gained [23]. The J/ψ decays 6% into an e^-e^+ pair and therefore, good electron PID is important in this study.

- Thermal photons toward investigating the initial state of the QGP [22]. This will help with determining the temperature of the QGP and its equation of state [21].

- Searching for bound states involving strange mesons [22].

Together, these areas of study will further propel research at ALICE in the direction of its final goal of delineating the complete properties of the QGP and thereby gaining a deeper understanding of QCD. During LS2, the ALICE detector will undergo significant upgrades to take advantage of the new scientific opportunities and cope with the data load from the significantly increased luminosity. These upgrades include:

- Major upgrades to several ALICE detectors including the ITS, TPC, and the new Muon Forward Tracker (MFT).
- Upgrades to readout electronics with the aim of boosting readout speeds from 8 kHz to 50 kHz for Pb-Pb collisions.
- New Fast Interaction Trigger (FIT) with < 425 ns latency [22].
- Introduction of the new online-offline (O^2) software framework.

As upgrades are made to the CERN LHC during the period of LS2, corresponding upgrades and modifications need to be made to the TRD to cope with the changing conditions of data-taking. The TRD was originally designed to function at an interaction rate of 8kHz for Pb-Pb collisions. It is planned to raise this rate to 50 kHz in Run 3 [24]. Since ALICE will no longer be using a trigger and will, instead, operate under continuous readout, the TRD data flow needs to be significantly optimized. To achieve this, it will only read out tracklets in Run 3 and no raw data. A considerable amount of effort is going into re-working TRD data classes and porting all TRD software to the new O^2 software framework while making necessary modifications to accommodate the new conditions. No significant hardware changes are being made to the TRD in LS2 aside from modifications necessary to accommodate the new Common Readout Unit (CRU).

Work will also need to be done in preparing software for the TRD control and communications systems. Software will need to be installed and upgraded on TRD nodes at P2 as a departure is made from Scientific Linux 6 in favour of CentOS 7 which will be supported until the year 2024. Software packages need to be ported to the new distribution and hosted in repositories at CERN with continuous integration in place for automated deployment. As an additional consequence of the changing data-taking conditions, updated firmware will also need to be installed on the detector modules at P2.

When the new data-taking run begins in 2022, the TRD will need to be properly calibrated to ensure that the data is processed correctly and can be analysed in a meaningful way. Several parameters need to be tuned in order to produce reliable measurements. These parameters include:

- Relative gain. The relative factor by which detected signals are amplified in the amplification regions of each chamber. This is a hardware issue specific to each chamber. Gain calibration may be done using an active krypton source. The krypton gas may be distributed throughout the TRD gas volume in addition to the Xe + CO_2. If the distribution of the gas is uniform, then the response from each chamber pad should be equal. In this way, individual pads can be calibrated relative to each other.
- Drift velocity. The velocity at which ionized electrons move towards the cathode plane. This varies primarily due to fluctuations in atmospheric conditions such as temperature and pressure as well as the gas composition. Calibration needs to be performed at regular intervals during run time.

- Lorentz angle. The lateral movement of drifting electrons due to the magnetic field. This is measured directly by comparing the measured trajectory of the primary particle with the trajectory of the drifting electrons.
- Time-offset. The drift start time. This is approximated as the time when the ionizing particle reaches the pad plane.

Each of these parameters need to be calibrated for all 522 chambers individually. Software to perform the calibration will be written and tested on collections of sample data.

1.6 Thesis Outline

This thesis comprises three distinct yet connected parts. Following the brief introduction to experimental particle physics and CERN in chapter 1 above, the first main section of work is presented. This section is related to core TRD software and firmware and is discussed in chapter 3. This is software that is essential for the operation of the detector. Some parts of the software are installed on Linux nodes at CERN while other parts are installed on the individual detector modules themselves. Following this, the next three chapters concern TRD calibration development and testing. These, collectively, comprise the second main body of work. In chapter 4, the development of the Run 3 calibration software is documented and testing is done in a Run 2 environment. In chapter 5, a number of quality control parameters are investigated that inform conclusions made in 6 where the performance of a test calibration is examined in more detail. The final body of work is related to the development of the O^2 software framework and various classes that were written to perform functions necessary during data-taking. This can be found in chapter 7. Finally, in chapter 8, a brief analysis is done on some fresh data that was taken by the TRD in the build-up to and during the pilot beam runs of late 2021.

In the appendix, an assortment of small, miscellaneous projects that do not fit well into the main narrative are briefly discussed. These include documentation on a custom WSL2 CC7 distribution in appendix B, information regarding the migration of the alicetrd web page in appendix C, and details of work done toward the TRD self-tracking project in appendix D.

Together, all these pieces of work form a significant contribution to the recommissioning of the TRD for Run 3 and the eventual collection and analysis of new physics data.

Chapter 2

TRD Software Introduction

In this brief chapter, further detail is provided regarding the means by which the TRD is controlled and how raw data is read out. This will provide the necessary context for the work described throughout the following chapter.

2.1 Raw Data Readout

On the top side of each TRD chamber (that is, the side at larger r), the chamber is sealed with a printed circuit board that houses the cathode read out pads [14]. On top of this board, outside the chamber, are mounted all the Front-End Electronics (FEE) that control initial raw data processing, readout, and power management. The smallest processing unit within the TRD is the Multi-Chip Module (MCM). This small computer performs the initial processing of raw data. Physically, the individual MCM module is connected to 18 pads, however each MCM receives inputs from 21 different ADC channels meaning that some pads are shared between two MCMs [14]. To be exact, MCM m receives one additional input from MCM $m+1$ and two additional inputs from MCM $m-1$. The pad sharing is to allow for the calculation of tracklets that cross the border between two different MCMs. Each padrow is therefore made up of $144/18 = 8$ MCMs. The raw data itself is written in binary. One way to view it is in the form of hexadecimal with a series of headers that identify the exact MCM module from which the data is coming followed by a long list of ADC values. Data is sampled at a rate of 10 MHz (100 nanoseconds per sample) and ADC counts take values in a 10 bit range from 0 to 1023. Higher values correspond to larger charge deposition in the detector gas. This raw data is amplified and shaped by the Pre-Amplifier and Shaper Amplifier (PASA) [25]. The amplification boosts the signal while the shaping assists in correcting the signal for the ion tail effect. The ion tail effect is a physical effect derived from the finite response time of the pads. Each pad is characterized by a signal-time response function which is sharply peaked, but followed by a tail. This means that any signal read by an MCM will persist over a small amount of time and may constructively interfere with subsequent signals. This results in a noticeable distortion of the signal and needs to be corrected for. This effect is explored more in section 4.5.

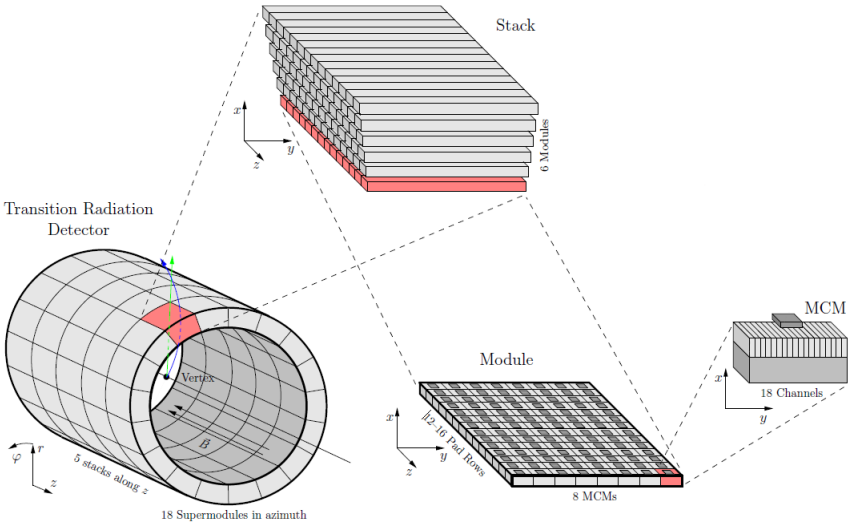

Figure 2.1: Alternative schematic to the one shown in section 1.4 of the TRD architecture showing the broader context of the single MCM [25].

After shaping and amplification, the raw data is then digitized by the Tracklet Processor (TRAP), which is also situated on the MCM. In the context of raw data, a digit refers to a 5-dimensional tuple that associates a 10 bit ADC value with a particular half-chamber, pad row, pad, and timebin coordinate. Therefore it is not a vector in Cartesian space, but rather in detector pad-timebin coordinates. However, this vector may be transformed into a spatial vector if certain other parameters are known such as drift velocity and sampling rate. This is discussed in more detail in section 4.2. The number of timebins that are read out during a single event is related to the sampling rate and is a variable parameter programmed into the TRAP by the TRAP configuration. It typically takes values between 24 and 30. A time bin represents a unit of time during which ADC signals are accumulated by the read out pads and then written into a single digit per pad. The time bin coordinate of the digit is proportional to the actual time elapsed since the beginning of the read out event. The TRAP configuration defines a set of parameters related to raw data read out and processing and is loaded by the TRAP from a database before each run. New configurations can be created and uploaded to the database as needed. With over 1 million pads in the TRD and 30 samples per pad, the first 10 bits for the ADC values of the digits alone already account for almost 40 MB of data for a single 3 μs read out. Several other parameters aside from just ADC values are recorded as well which makes streamlining the data an imperative.

After digitization, the TRAP will fit a straight line to the digits in pad-timebin space called a tracklet. The shared pads of the MCMs allow for calculation of tracklets which cross the border between two MCMs. The range of the fit is typically done over the drift region only as opposed to the full timebin range. This is because only the digits in the drift region are expected to take a linear shape in time due to their constant velocity. The exact range of the fit is controlled by the TRAP configuration[1].

[1] See [26] for exact details on the tracklet fitting procedure.

2.2 DCS Overview

2.3 Communications Flow

Data from individual MCMs are merged first by column, then readout board (ROB), and finally by half-chamber mergers, and then sent via optical fibre to the Global Tracking Unit (GTU) for further processing. C0 chambers have 6 readout boards, while C1 chambers have 8. A half-chamber will consist of one half of the readout boards on a single chamber. The half-chamber ID (HCID) is a value ranging from 0 to 1080 and is used as a unique identifier for each half-chamber. Each TRD chamber is fitted with a Detector Control System (DCS) board which facilitates all external communications. The DCS board consists of an Altera EPXA1 chip with an ARM processor and FPGA as well as 5 MB of flash memory [25]. The processor runs a minimal Linux distribution which can be interacted with via secure shell (SSH).

Every ALICE detector control system is organized into three distinct yet integrated communications layers: the supervisory layer, control layer, and field layer [27]. A diagram of the control flow is shown below in Figure 2.2.

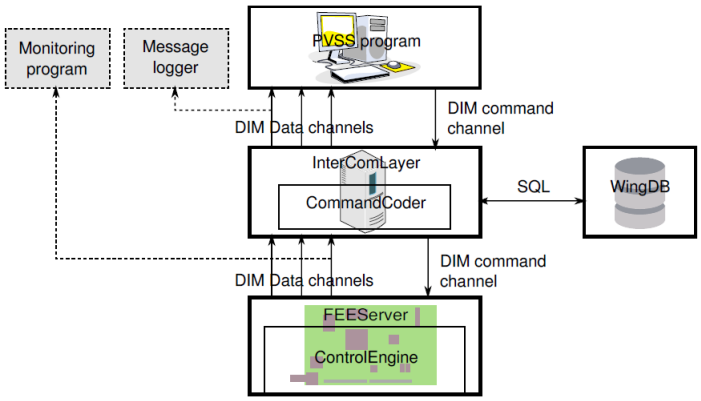

Figure 2.2: Control flow diagram for communications with the TRD [27].

The top supervisory layer consists of the Process Visualization and Control System, WinCC. This provides a front end interface for users to interact with the DCS board to control the TRD as well as many other detector systems like high and low voltage. Commands and configurations sent by the user via the PVSS system are interpreted by the Intercom Layer (ICL) in the control layer. Aside from sending commands via PVSS, detector experts may also control the TRD via command-line either from nodes within the DCS network[2] or from the DCS boards themselves. Each command is identified by a unique integer that will be matched to a particular configuration which consists of a set of Slow Control Serial Network (SCSN) commands which are low-level commands to be interpreted by individual MCMs. The TRD command coder will look up the specified integer tag in the wingDB and report the configuration back to the ICL [27]. These configurations are stored in a central Oracle database called the wingDB. Next, the configuration may be communicated to the FEE server which runs on the DCS board in the field layer. The FEE server will register all the individual commands in the configuration and send them to the MCM modules. In this way the TRAP and PASA chips may be configured in a particular way for data taking.

[2]The DCS network is a secure network at CERN with connection to all DCS boards in the TRD. It is only accessible via a gateway network by detector experts.

Communications between all layers are handled by the Distributed Information Management System (DIM) which is a robust communications library developed in-house at CERN [27]. The FEE server will publish information regarding detector state that can be read by other programs via the DIM network.

2.4 Ansible

Ansible is an open-source tool providing automated remote configuration of Windows and Unix operating systems from a central node. A user may declare a list of tasks called a playbook which can be run on one machine and act on any number of other hosts. Roles are a more complex task structure which may consist of additional components like configuration files and templates. Several roles may be called from a single playbook. The TRD Ansible code is hosted in a private GitLab repository under the `alitrd` name space [28]. It contains various Ansible roles and playbooks designed to manage core TRD software. The goal is to accelerate system configuration over several machines by automating the majority of the procedure and providing an easy way to repeat it when necessary. In some sense, it also serves as documentation as the tasks are easily human-readable and provide a history of setup commands and procedures.

Chapter 3

TRD DCS Linux Software

This chapter describes work done toward preparing the TRD core software stack for Run 3. This is essential software required for the control and communications of the TRD. Software components are installed and configured on Linux nodes at P2 and on the TRD chambers themselves. Until 2020, much of the software was still being stored in a Subversion repository at Heidelberg University [23]. This included the use of a Continuous Integration (CI) service called Build Bot. These repositories were ported to the CERN GitLab under the `alitrd` namespace and recompiled for various versions of CentOS which is the new Linux distribution that will be running on the nodes at P2. Gitlab CI was also set up for each ported software repository so that packages can be built and deployed automatically whenever changes are committed. The built binaries are uploaded by the CI pipeline to a yum repository on the alicetrd web page where they can be picked up by other machines. After porting, the appropriate software was installed and configured on the new CentOS nodes at P2 that were provided by the ALICE DCS team. The configuration was primarily done using Ansible which allows the setup to be easily repeated by a non-expert at a later date if needed. Further work was done toward preparing and testing new TRAP configurations for Run 3 and migrating the TRD high-voltage power supply control systems to a new network.

3.1 Migration from Subversion to Git

3.1.1 Methodology

The motivation behind migrating the TRD software stack from Subversion (SVN) to git was, firstly, in order to make use of a more widely-used version control system with support for robust continuous integration services. And secondly, to move the code to a more central location not owned by a specific academic institution. During the migration, the desire was to preserve full commit history and authorship details of the original creators of the software. This required the use of the `git-svn` package which comes standard with most installations of git to facilitate the move[1]. The complete process from checking out the remote SVN repository to pushing to git is described below.

First, begin by checking out the SVN repository to a local directory with

```
svn checkout <SVN_REPO>
```

From the top directory of the newly checked-out SVN repository, run

[1] Documentation is available at `https://git-scm.com/docs/git-svn`.

```
svn log -q | awk -F '|' '/^r/ {sub("^ ", "", $2); sub(" $", "", $2); print
↪  $2" = "$2" <"$2">"}' | sort -u > authors-transform.txt
```

This will generate a new file called `authors-transform.txt` in the working directory with a list of all authors who have ever committed to the repository. In the file, authors are identified by their SVN usernames which will not, in general, be recognized by git. These need to be edited and replaced by the authors' full names and active email addresses. A list of contributors common to many of the repositories in the TRD group is shown in appendix A. After the author list has been modified appropriately, run

```
git svn clone <SVN_REPO> --no-metadata -A authors-transform.txt --stdlayout
↪  ~/temp
```

to create a new local git repository based on the local SVN checkout and the new author list. The git repository will be created in `/temp` given the above command, but this parameter can be changed as desired. Next, navigate to the fresh git repository and run each line shown below in sequence.

```
git svn show-ignore --id=origin/trunk > .gitignore
git add .gitignore
git commit -m 'Convert svn:ignore properties to .gitignore.'
```

These three lines will copy the SVN ignore property and rename it to `.gitignore` so as to comply with the standard git format. It will then tell the local git repository to begin tracking this file and commit it with the message `Convert svn:ignore properties to .gitignore.`. This message may also be changed to suit the user's preference.

After that, the following two bash loops should be run to create corresponding git tags for each SVN tag and corresponding git branches for each SVN branch.

```
for t in $(git for-each-ref --format='%(refname:short)'
↪  refs/remotes/origin/tags);
do
  git tag ${t/'origin/tags/'/} $t && git branch -D -r $t;
done

for b in $(git for-each-ref --format='%(refname:short)'
↪  refs/remotes/origin);
do
  git branch ${b/'origin/'/} refs/remotes/$b && git branch -D -r $b;
done
```

Sometimes it is also necessary to manually delete the `trunk` git branch and tag if they were created by the above commands. The `master` branch will be a duplicate of the `trunk` branch making it redundant. This action can be performed with the two commands,

```
git branch -d trunk
git tag -d trunk
```

Finally, it is necessary to configure the remote address of the new git repository and push all commits.

```
git remote add origin
↪  ssh://git@gitlab.cern.ch:7999/alitrd/<REMOTE_REPO_NAME>.git
git push -u origin --all
git push -u origin --tags
```

The SVN repository should now be successfully migrated to git with a fully preserved commit history.

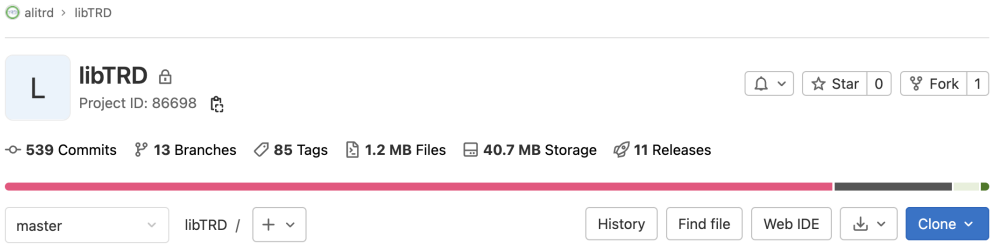

Figure 3.1: An example of a ported Subversion repository on GitLab. The package `libTRD` is shown. All commits, branches, tags, and releases are preserved. See section 3.1.1 for more information on this package.

3.1.2 Summary of Packages

Following, more detail is given about each of the TRD packages which were migrated.

libdim
Distributed Information Management (DIM) System libraries for detector communications (see Figure 2.2). Failure to compile on CentOS 7 required all files in the `dim` directory to be replaced by Dim version 20r26 from the official CERN-DIM download page.

libTRD
Collection of libraries required for operation of the TRD. Builds of this package are required for CentOS as well as the DCS boards. `#include <unistd.h>` header was added to `read_volt.cc` and `shutdown.hh`. CI support for building on CentOS 8 was added to a new `centos8` branch.

FEEServer
`FEEServer` was merged into `trdCE`. It has been ported to GitLab, but has no CI support since it does not need to be maintained any longer. Instead, refer to `trdCE` below.

trdCE
FEE Server Control Engine. This package is built only for the DCS boards (see Figure 2.2). The FEE Server receives commands via DIM and calls the appropriate function(s) from the control engine. `trdCE` is one of two libraries which together comprise the control engine. The other is `libTRD`. The work is divided among the two with `trdCE` controlling higher-level flow, while `libTRD` provides all the lower-level functionality [27]. No modifications were necessary for this particular repository.

pvss_mockup
Front-end for DIM communications with TRD. For the CentOS 7 build, the `AC_CHECK_LIB` for the `net-snmp` library was disabled in `configure.ac` as it threw a fatal error and is not required.

gateDB
Global Alice TRD Electronics Database for ALICE TRD electronics components. The database stores complete information about the location, history, status, tests, etc for all TRD components in a complex PostgreSQL DB. No CI was required as `GateDB` is not packaged or used in the current Run 3 system.

wingDB
The TRD Oracle Database containing all possible configurations for the FEE. It requires a functional installation of Oracle Instant-client on the host which is the database client. A shell script was committed to the repository containing instructions for setting up Oracle Instant-client. At the time of writing this, it is configured to install Oracle Instant-client 18.5. This is compatible with Oracle DB XE 18c, but not Oracle DB XE 19c. Version numbers are required to match if running on the same host. The shell script installs Instant-client 18.5 from the Oracle Linux yum repository. This is to avoid the password and click-throughs required when downloading the packages from the Oracle downloads page. The yum repositories must be downloaded and configured and then the packages can be downloaded. Finally, the Instant-client installation must be configured. This is done with a single command as per the shell script.

The shell script can be configured to instead install Oracle Instant-client 19. This is done by simply changing the PACK_VERSION variable to either 19.5 or 19.6. This will require an additional step which is to create a symbolic link between the `libocci` files which will otherwise not be found by the WingDB package. The user will also need to change lines 33 and 34 of the `configure.ac` file to set the flags correctly. These lines need to correspond correctly to the installation directory of the Instant-client. Lastly, the user should also change lines 73 and 74 of the `configure.ac` file to correctly reflect the version of `libnnz` which corresponds to the Instant-client version number.

CommandCoder
Command Coder library for TRD. This package also requires Oracle Instant-client. The notes written above for `wingDB` may be consulted for more detailed instructions. Paths and includes were changed for legacy `Boost` headers.

iseg2dim
Package to control Iseg HV power supplies with DIM. When installing this package, it is necessary to install DIM and libDIM. The dim dns server should be started with the command `dim_dns` on the correct node and then the `DIM_DNS_NODE` environment variable can be exported by running
`export DIM_DNS_NODE=alitrddimdns`.

trapasm
TRAP Assembler. No modifications were made.

trapcc
TRAP Configuration Compiler. No modifications were made.

3.2 GitLab Continuous Integration

GitLab offers a versatile CI service which allows users to create automated processes that are triggered by specified code changes. This can be used to test, merge, and deploy software packages. On GitLab, the CI is configured using a file called `.gitlab-ci.yaml`, located in the top level directory of the repository. Creating this file will effectively "switch on" CI for the corresponding repository. All the ported TRD packages share a similar structure in terms of the layout and functionality of the CI. Some repositories also use a CI template which is imported by `.gitlab-ci.yaml` when the pipeline is triggered. These template files are located in `build-aux/` and are accompanied by other scripts which control version numbering. A typical example of a CI template common to many of the ported repositories is documented below.

```
variables:
  EOS_PATH: \${RPM_EOS_PATH}/download/yum
  EOS_ACCOUNT_USERNAME: \${EOS_SERV_USER}
  EOS_ACCOUNT_PASSWORD: \${EOS_SERV_PASS}
```

In these first few lines, three variables are declared which remain in scope for the entire pipeline. These are the default variable names used by the container[2] that facilitates the deployment of the built packages to the secure alicetrd yum repository. The repository is hosted on CERN EOS Open Storage (EOS)[3]. The deployment stage is discussed in more detail later. The variables `${RPM_EOS_PATH}`, `${EOS_SERV_USER}`, and `${EOS_SERV_PASS}` are masked variables that are declared in the repository settings under `settings > CI/CD > Variables`.

Following this, a template is created called `build_centos_dnf`. All template definitions must contain a leading period. This template may be included in the primary `.gitlab-ci.yaml` and extended with the `extends` keyword. Here, the `build` stage is defined. A Docker image[4] is specified within which the stage will be executed. The image used in this example is one of a series of custom minimal CentOS distributions which are configured with basic compilation tools and dependencies as well as access the alicetrd yum repository (see section 3.2.1 for more detail). Using the `only` keyword, the pipeline can be restricted to running only after changes to branches or tags.

```
.build_centos_dnf:
  stage: build
  image: gitlab-registry.cern.ch/alitrd/alitrd-containers:\${OS}
  only:
    - branches
    - tags

  before_script:
    - dnf -y --enablerepo=PowerTools install \${REQUIRED_PACKAGES}

  script:
```

[2] In this context, a container is a standalone runtime environment configured in a particular way to support the execution of a program or task.

[3] EOS is a secure, extensive, disk-based storage platform primarily used by collaborations at CERN to store experimental data.

[4] An image is a file that is essentially a snapshot of a container and will be used to instantiate a container at runtime.

```
    - ./bootstrap
    - ./configure
    - make
    - make rpm

    - chmod -R 755 /root/rpmbuild

    - mkdir -p public/\${OS}/x86_64/packages &&
      cp -r /root/rpmbuild/RPMS/x86_64/* public/\${OS}/x86_64/packages &&
      cp /root/rpmbuild/SRPMS/* public/\${OS}/x86_64/packages

  artifacts:
    paths:
    - public/**
```

Next, a series of commands are run in bash beneath the `before_script` and `script` tags. These commands are divided into two sub-stages that can easily be extended in the `.gitlab-ci.yaml` configuration file. This template was specifically created with CentOS 8 in mind and therefore uses `dnf` package manager as opposed to `yum`. The `PowerTools` repository is also required for the installation of some dependencies. The package is then compiled using `autotools` and the output RPM files are moved from the default build directory of the system to the default directory where packages are deployed from by the container used in the deployment stage. Some packages were migrated to a `CMake` build system (see table 3.2). A specific file structure is chosen here so that the files may be easily merged with the remote repository file system in the deployment stage to which all build artifacts are passed.

Below is a second template which itself is an extension of the previous. With a second definition of the `before_script` keyword, the original `dnf` command is overwritten by this new `yum` command which is compatible with CentOS 6 and 7.

```
.build_centos_yum:
  extends: .build_centos_dnf
  before_script:
    - yum install -y \${REQUIRED_PACKAGES}
```

Next, the deployment stage is defined. While the previous stage will be triggered by commits to any branch or any new tags, the deployment stage will only be triggered by new tags. The tag should take the form of a "v" followed by the version number. Some repositories are configured to use the tag directly for release numbering. The deployment stage uses a Docker image developed specifically for use with CERN EOS [29]. The `deploy-eos` command launches a binary that reads the previously defined EOS credential variables and copies all files in the default `/public` directory to the specified `EOS_PATH` directory on EOS using `rsync`. At this point in the process, the files will have been copied to the alicetrd yum repository, but the repository details still need to be updated to reflect the changes to its local files. To do this, a secure shell is opened to `lxplus` which is a computing cluster for CERN users and has EOS mounted by default. Then, a shell script is executed which contains the necessary commands to update each of the CentOS 5, 6, 7, and 8 repositories. To authenticate access to `lxplus`, Kerberos is used after writing a brief configuration file to `/etc/ssh/ssh_config` that contains a number of settings to ensure that an ssh connection can be established with this particular authentication method by copying the Kerberos token to the remote server. `StrictHostKeyChecking` is also disabled to block a prompt that would appear while connecting and requires additional user input.

```
deployment:
  stage: deploy
  only:
    - tags
  image: gitlab-registry.cern.ch/ci-tools/ci-web-deployer:latest
  script:
    - deploy-eos
    - printf "Host *\n\tGSSAPIAuthentication
    ↪  yes\n\tGSSAPIDelegateCredentials yes\n\tGSSAPITrustDNS yes\n" >
    ↪  /etc/ssh/ssh_config
    - echo "\${EOS_SERV_PASS}" | kinit -f \${EOS_SERV_USER}@CERN.CH
    - echo "/eos/project/a/alice-trd/www/download/yum/updateRepos.sh" | ssh
    ↪  -o StrictHostKeyChecking=no \${EOS_SERV_USER}@lxplus.cern.ch
```

In a similar way as the yum packages, ipkg packages built for the DCS boards are deployed to an ipkg repository located at `/eos/project/a/alice-trd/www/download/ipkg`. Each package has slightly different requirements so the CI is customized for each unique case. Therefore, each repository should be examined individually for full understanding of the CI. Above is merely one example.

3.2.1 Custom CentOS Docker Containers

All building of TRD software packages on GitLab occurs in several custom-built Docker containers. The containers and the corresponding Docker configuration files are kept in the `alitrd-containers` GitLab repository [28]. Each Docker image for each distribution version is kept in its own git branch. Images may be built locally and tested before pushing them to git. Images are configured using a single `Dockerfile`. This file contains a reference to a base image followed by a series of commands to run that can perform further configuration. As an example, to build an image from a directory containing a valid `Dockerfile`, the command `docker image build -t alicentos7` may be run. This will build the image and tag it with the label `alicentos7`. After building, the image may be explored via an interactive shell by running `docker run -it alicentos7 /bin/bash`. This command will run the container and launch `/bin/bash` for the user.

Each branch is also configured with a CI pipeline which will be triggered by any push to the respective branch. The CI pipeline uses the `gitlab-registry.cern.ch/ci-tools/docker-image-builder` image to automate building of the Docker image and deployment to the `alitrd-containers` repository's container registry where it will be accessible to other external CI pipelines such as the ones managing the TRD software packages.

Building each Docker image involves three main steps. During the first step, repositories are configured to accommodate the installation of all the required packages. Usually, the Extra Packages for Enterprise Linux (EPEL) repository is installed followed by the alicetrd yum repository which can be configured by simply downloading the appropriate `.repo` file from the EOS web server. In the case of older distributions which are in End Of Life (EOL) phase for example CentOS 5 and 6, additional measures need to be taken to configure EPEL and re-configure base repositories. Base repositories are usually unofficially maintained by other users of the distribution and can be found in various places on the web. Once the repositories are correctly configured, all necessary packages may be installed. A list of all packages for each distribution is shown below in table 3.1.

	Installed on distribution image? [Y/N]			
Package	CentOS 5	CentOS 6	CentOS 7	CentOS 8
autoconf			Y	
automake			Y	
libtool		Y	Y	Y
make			Y	Y
cmake		Y	Y	
wget		Y	Y	
git			Y	
lesstif-devel		Y	Y	
boost-devel		Y	Y	Y
gcc-c++		Y	Y	Y
rpm-build		Y	Y	Y
java-1.8.0-openjdk-devel		Y	Y	Y
byacc			Y	
flex			Y	
flex-devel			Y	
bison			Y	
bison-devel			Y	
sshpass		Y	Y	
oracle-release-el6		Y		
oracle-release-el7			Y	
oraclelinux-release-el8				Y
curl	Y			

Table 3.1: A list of packages that are installed on the CI pipeline Docker containers used to build TRD packages.

Finally, Oracle client is installed on the CentOS 6, 7, and 8 images. Oracle offers a yum repository for use with many Linux distributions. Oracle is used by the `CommandCoder` and `wingDB` TRD packages. It requires manual creation of the Oracle `.repo` file to configure the repository. A key must also be downloaded from the Oracle website to gain access to the yum packages[5].

3.2.2 Package Status

Below, in table 3.2, is shown an overview of the distribution support for the core TRD software packages. All builds are done automatically via the gitlab CI service.

[5]More information can be found at https://yum.oracle.com/getting-started.html#configuring-your-system-to-install-software-from-oracle-linux-yum-server under the "CentOS & Red Hat Enterprise Linux" section.

	Compiled for distribution/system?					
Package	CentOS 5	CentOS 6	CentOS 7	CentOS 8	DCS	Build System
libTRD	Y	Y	Y	Y	Y	cmake
trdce					Y	cmake
libdim		Y	Y	Y	Y	autotools
wingDB			Y			cmake
CommandCoder			Y			autotools
trapcc			Y	Y		cmake
trapasm		Y	Y	Y		autotools
iseg2dim		Y	Y			autotools
pvss_mockup			Y			autotools
gateDB			Y			N/A

Table 3.2: Summary of Linux distribution support for TRD software packages.

3.3 Migration of HV Controllers

Several high-voltage crates can be found at P2 which supply power to the amplification and drift electrodes of the TRD chambers. These were migrated from the old PEAK control system to the new AnaGate system. This facilitates communications between a PC running WinCC (Windows control layer using PVSS) for DCS and the HV crates which contain the individual power supply modules. Among other motivations, the new system offered a more centralized method of control including the ability to be controlled remotely as well as no longer needing to be connected to a machine. The new system was first tested on a spare Linux worker node at P2 before being migrated to the new `alitrdwn005-new` node. Several components needed to be configured. This was done using Ansible (see section 3.3.1). The playbooks and accompanying configurations and scripts were copied into the DCS network as a git bundle (see section 3.4 for more information on importing git bundles into the DCS network). This method allows for the preservation of version control capabilities in order to ensure consistency between the files in the remote git repository and the files on the worker node. The repository contains the configuration file `iseg2dim.cfg` which is copied to `/etc` by the Ansible playbook. The configuration of the `iseg2dim` software should be changed by editing this file inside the repository. The repository also contains two `systemd`[6] services. The first is for `iseg2dim` itself and the second is for `SocketCANGateway`[7] which opens the communication channel between the machine running `iseg2dim` and the AnaGate controller. These two services are copied to the relevant directory on the host machine in `/etc/systemd/system/` so that they can be managed via standard Linux `systemctl` commands. A shell script that accepts and parses an argument is called by the `SocketCANGateway` service to execute the `SocketCANGateway` binary which can be downloaded from the official AnaGate website [31]. The web page lists the binary as for Ubuntu, but it was found to be compatible with CentOS 7 as well without any modifications. A schematic of the new set up is shown in Figure 3.2.

3.3.1 Ansible Playbook

Below is documentation for the iseg2dim Ansible playbook. The playbook begins with

[6] `systemd` is a Linux utility which controls the behaviour of system services.

[7] Controller Area Network (CAN) is a networking technology used prominently in automation and embedded systems [30].

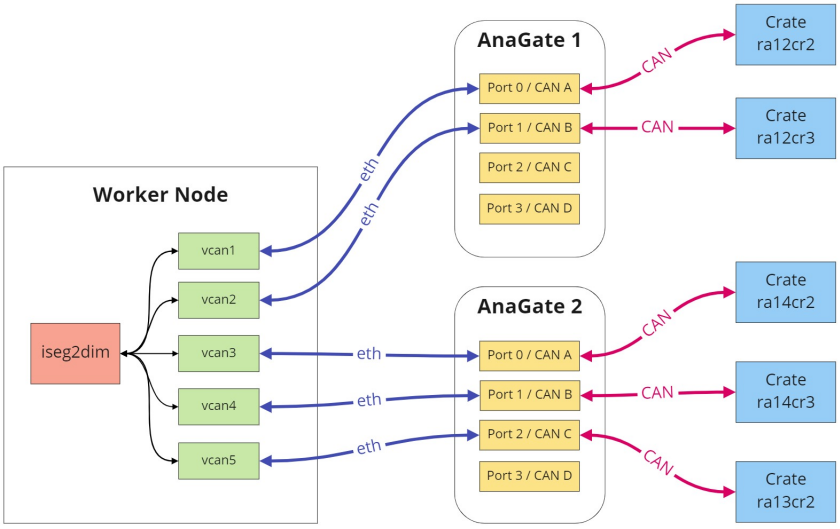

Figure 3.2: Communications flow for the new high-voltage control system. Worker nodes at P2 interact with the HV crates via two Anagate controllers which are communicated with through virtual CAN (vcan) interfaces on the node. Although a single Ethernet cable facilitates communication between the vcan interfaces and the AnaGate controllers, each interface is connected to a unique port. Each of these ports are, in turn, connected to a unique crate via CAN cables.

```
vars:
  iseg_crates:
  - name: ra12cr2
    iface: vcan1
    bitrate: 250000
    port: 0
    ip: 10.160.33.155
```

Here, each crate is defined. These are the crates that are connected to the AnaGate controllers which are, in turn, connected to the machine. The value of `iface` should be the name of the virtual CAN interface on the machine which is used to communicate with this particular crate. The `bitrate` is determined by the controller and is fixed at 250 000 b/s. The `port` is the corresponding port on the AnaGate controller that the CAN interface should connect to. `ip` is the IP address of the AnaGate controller on the local network.

Several tasks are executed next.

```
- name: Load vcan kernel module
  modprobe:
    name: vcan
    state: present
```

In this task, the `vcan` kernel module is loaded so that virtual CAN interfaces can be configured on the machine. This module is available by default on most modern Linux distributions. After this, the vcan interfaces are configured using the `iseg_crates` variable declared above. Next, the configuration file, `iseg2dim.cfg` is copied to `/etc` where it will be picked up by `iseg2dim`. In the following task, the `systemd` services for the `SocketCANGateway` and `iseg2dim` are copied to the local `systemd` directory. The accompanying shell script used to launch the `SocketCANGateway` binary is also copied to the same location.

```
- name: Start SocketCAN gateway services
  systemd:
    name: socketCANGateway@{{ item.iface }}:{{ item.port }}:{{ item.ip }}
    state: started
  with_items: "{{ iseg_crates }}"
```

In this task, the `SocketCANGateway` service is started. Everything after the "@" sign are arguments passed to the service. It accepts three arguments which are `iface`, `port`, and `ip`. These are passed as a string with a colon as a delimiter between arguments. This is the format recognized by the shell script. This is also a way to uniquely identify each `SocketCANGateway` service which all share the same `systemd` service template. Lastly, once the `SocketCANGateway` services have been started, the `iseg2dim` service is started. The `DIM_DNS_NODE` environment variable is declared in the playbook and may be edited here.

3.4 Configuring New Worker Nodes

After testing the software on an experimental worker node in the DCS network, the software could be configured on the new primary worker nodes. These were `ALITRDWN005-NEW.cern.ch`, `ALITRDWN006-NEW.cern.ch`, and `ALITRDWN008-NEW.cern.ch` which could each be accessed from the DCS gateway node[8]. The worker nodes in the DCS network do not have a direct internet connection therefore a special data pipeline had to be set up to transfer files to and from the new nodes. Furthermore, due to security reasons, package binaries are not permitted to be imported into the DCS network. This means that all packages need to be compiled on the relevant worker node before they can be installed. The source code for each package was transferred into the DCS network in the form of git bundles. This flow involves several stages and different systems and networks. Repositories need to first be cloned from the gitlab remote. This can be done on the CERN terminal service machines which are on the General Purpose Network (GPN). Since these are Windows machines, `git bash`[9] should first be installed to allow for command line use of `git`. Once the repository has been cloned locally, the command `git create bundle <repo>.repo --all` may be run. This will create a git bundle named `<repo>.repo` where `<repo>` should be replaced by the name of the repo or any other valid file name. The option `--all` instructs `git` to package all references into the bundle. The bundle can then be copied into the TRD import directory of `aldcsfs001` which is mounted on the CERN terminal service machines. The absolute path to this directory is `\\aldcsfs001.cern.ch\Import\trd`. Once inside the `import` directory, the file will automatically be copied into the DCS network usually within 15 minutes after being scanned for security threats. The file will appear in the corresponding directory on `alidcsfs001` which will be `\\alidcsfs001.cern.ch\Import\trd`. The principle is the same in the opposite direction. To export files out of the DCS network, they should be copied to `\\alidcsfs001.cern.ch\Export\trd`. In the same timeframe as for imports, the file(s) will be automatically copied to the `aldcsfs001` and appear in `\\aldcsfs001.cern.ch\Export\trd`. From inside the DCS network, the git bundle can be copied to the relevant worker node using `smbclient`. This should be done from the node itself using a command of the form

```
smbclient -U CERN\<username> '\\alidcsfs001.cern.ch\Import' --directory
trd -c 'get <file>'
```

[8] These names are temporary and will drop the "-NEW" suffix once the old nodes are decommissioned.
[9] Git bash can be obtained from https://gitforwindows.org/.

where <username> and <file> are replaced by the appropriate values. Conversely, to copy a file from the node into the Export directory of the DCS gateway node, the command would be

```
smbclient -U CERN\<username> '\\alidcsfs001.cern.ch\Export' --directory
trd -c 'put <file>'.
```

Alternatively to smbclient, the program WinSCP may also be used which is installed by default on the windows machines inside the DCS network. The bundle copied on to the node may be cloned if the local repository does not already exist or pulled if it does. A schematic of this flow is shown in 3.3.

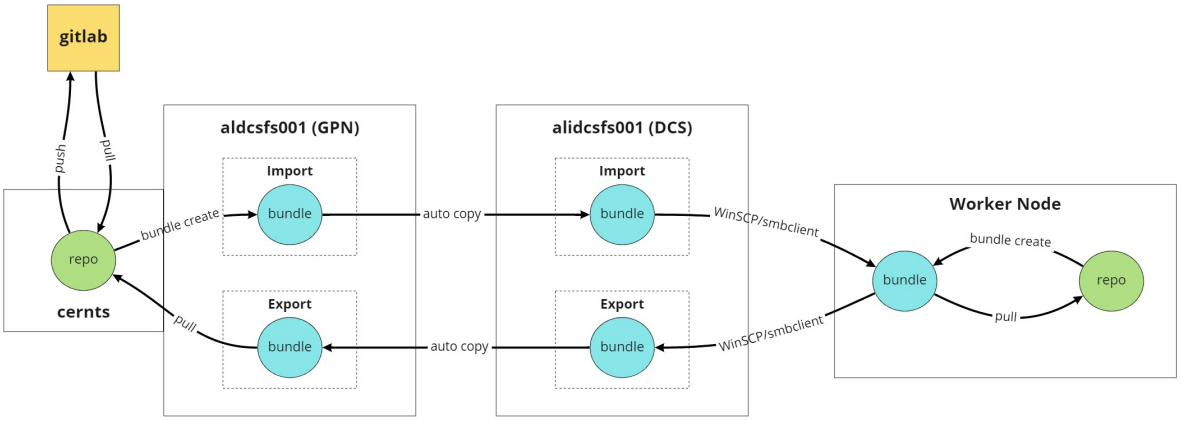

Figure 3.3: Data flow for file import and export from the DCS network.

Packages were compiled on the worker nodes and served from a http server within the network on ALITRDWN005-NEW.cern.ch. The server is started automatically on boot. It can be found in /var/www/html and serves both yum and ipkg packages to other nodes on the DCS network.

In addition to configuring the new nodes, it was also necessary to update some of the firmware on the DCS boards. This was done via cluster ssh (cssh) from one of the worker nodes within the DCS network. Cluster ssh is a utility that allows multiple servers to be managed from a single terminal. Individual DCS boards can be accessed via one of two aliases. The first is alidcsdcb{0000..0539} where the boards are identified by a four-digit detector number with padding. The other way is alitrddcb{<sector><stack><layer>} where the boards are identified by their location within the sector-stack-layer system. The DCS boards are configured to connect to an ipkg repository at the alias of alitrdsds. This alias was changed to match the new location of the ipkg repository on ALITRDWN005-NEW.cern.ch. Then, from the cssh terminal, firmware could be updated by simply running ipkg update to refresh the repository information followed by ipkg upgrade to download and install the updated packages. The firmware that was updated was libTRD and trdce (refer to table 3.2). These were updated to versions $\geq$ 1.3.0 and 1.3.13 respectively. As of writing, versions beyond these involved changes bearing no effect on the core functionality of the firmware.

3.5 TRAP Configurations

Herein is described a process for building TRAP configurations and uploading and tagging them in the wingDB. This assumes that the working machine is running an instance of

Oracle Database with `wingDB`, `trapcc`, and `trapasm` installed. These packages can be obtained from the alicetrd yum repository. The repository can be configured by downloading the relevant `.repo` file for example https://alicetrd.web.cern.ch/alicetrd/download/yum/centos7/x86_64/alicetrd-7.repo for CentOS 7 where the version number 7 can be replaced by any of 6, 7, or 8.

3.5.1 Compiling in Run 2

With all dependencies mentioned above correctly configured, the TRAPconfig Subversion repository can be checked out from https://alice.physi.uni-heidelberg.de/svn/trd/TRAPconfig/trunk/. This repository contains a more detailed readme from which many of the instructions repeated here originated. Then, the configuration files can be built by running

```
make compat
```

in the trunk directory. This will build all configurations contained in the `gen_configs.list` file. Additional configurations can be placed in this file if necessary by following the format as described in the `gen_configs.list` file itself. Once the configurations have been built, they should be visible in `trunk/configurations`.

Next, a .sqlldr file must be built which is in an appropriate format for upload to the wingDB. This can be done by running

```
make upload/VERSION.sqlldr
```

from the trunk directory. VERSION should be substituted for the subversion revision number. This will build all configurations which are in the `trunk/configurations` directory and write the .sqlldr file to `trunk/upload`.

3.5.2 Compiling in Run 3

In the new way, Run 3 configurations are built from the same subversion repository, but using Ninja. After cloning the repository, do

```
python3 -m pip install -r requirements.txt
```

to install all prerequisites and then

```
make build
```

to produce the `.sqlldr` file.

To add or modify the configurations that are built. Two files need to be edited. Firstly, in `cfgbundles/`, one or more `.yaml` files may be created, each defining a ”bundle” of configurations. In the `yaml` files, a new entry should be added for each configuration following the format of the previous examples. Under the `vars:` key, specific parameters may be defined. These should match parameters which are listed in the `src/common/run_parameters.tcs` file. The `run_parameters.tcs` file may be edited by adding a question mark before the equals sign for any one of the variables for example `asm DYN_L1A ?= 1`. This will allow the variable to be defined in the `yaml` file under the `vars:` key.

3.5.3 Uploading to WingDB and Tagging

Now the .sqlldr file can be copied to a worker node in the DCS network and uploaded to the wingDB. Transferring the file into the DCS network is achieved in the same way as for the git bundles (see Figure 3.3). Currently the upload is still done from `ALITRDWN008`. Specifically, it should be done from the `/home/trd/users/jkl/instantclient` directory after loading the environment variables defined in `env.sh`. Then, the upload can be performed by running

```
./sqlldr 'userid=<USER>/<PASS>@<DB_NAME> control=<REVISION>.sqlldr'
```

If the configurations were uploaded successfully, they should be visible when running

```
wing_tags avail
```

At this point, they can be tagged. Tagging is done using the command

```
wing_tags add <NUMBER> CFCFG <CONFIG_NAME> <REVISION>
```

If done correctly, the tags should be visible by running

```
wing_tags ls.
```

3.6 Resolving Readout Issues

An issue arose during recommissioning that involved current spikes when configuring or triggering the TRD. This was found to occur when using a newly built Run 3 krypton TRAP configuration. These spikes would result in low-voltage trips as super modules attempted to draw more power than what safety limitations allow to be supplied. Tests were done on the C0 TRD chamber at the University of Cape Town[10] involving configuring the chamber with an identical configuration as to what was used at P2 and sending software triggers. This is an alternative to the typical hardware triggering which, of course, could not be done due to the lack of a trigger. Contrary to what was seen at P2, all current and voltage readings were found to be nominal during these tests. When testing at P2, however, it was found that the current would remain at around 10 A after configuring, but then would immediately spike to around 16 A after sending the first hardware trigger. The process for sending hardware triggers is described briefly: From `lxplus`, do `ssh -Y trd@aliflpcom-ctp` to log in to the flp. From there, navigate to the TRD `run_defs` directory and make any necessary edits to the `trd.par` file. This file controls the number and rate of triggers to be sent. Next, enter the Experimental Control System (ECS) environment with `ecsc` and then start the run with `rs trd <RUN_NUMBER> trd.par` where `<RUN_NUMBER>` should be a unique integer of any length. This will start the run and immediately begin sending triggers at the predefined rate. To stop the run, do `rstop`. During the tests, the run was set up to send a single trigger. When looking at the run state with the command `nginject -s rstate sm17 10000` after stopping the run, many board mergers were seen to be stuck in readout. An example of the `rstate` output for stack 4 is shown in Figure 3.4.

[10] Along with Heidelberg University, the University of Cape Town has a working TRD chamber which can be used for experiments and testing.

```
17_4_0 6:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_0 1:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_0 7:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_1 2:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_1 3:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_2 2:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_2 7:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_3 0:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_3 7:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_5 0:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
17_4_5 2:16     +-BM----> full_rd    idle_rr  events: 1 pretriggers: 0
42 ERRORS, 26 ROCs, 0 pretriggers, 1 events
```

Figure 3.4: `rstate` of stack 4 in super module 17 after configuring with the problematic Run 3 krypton configuration and sending 1 hardware trigger. Board mergers can be seen to be stuck in the "full_rd" state meaning that they are still attempting to read out data.

Similar output was seen for all other stacks except stack 2. Initially it was thought that the cause was the MCM's filter settings. It was found that the settings indeed differed between the old and the new configurations. In the new configuration, the gain and tail cancellation filters were enabled, while in the old configuration, they were disabled. The filter settings are set in the TRAP register and can be checked by running `peek <REGISTER>` from the DCS board. Registers of the various settings can be found in the TRAP User Manual [26]. The filter settings were reconciled, but the issue persisted. It was also suggested that if the filter settings were the problem, the current would be high already from the moment of configuration and should not only spike when the first trigger is sent. Communication was initiated with Venelin Angelov, Ph.D., who recommended that all ADCs be disabled to minimize the power consumption of TRAP and determine if the readout was stable under these conditions. This was done with `poke ADCMSK 0; poke 0xF0F5 0`. As opposed to the command `peek` which reads a value at the given address, `poke` sets a value. The second address in the above command refers to the `ADCMSK_DB` which gets written back into `ADCMSK` after the first trigger to protect from Single-Event Upsets (SEUs)[11]. Therefore it was necessary to set both registers to 0 so that the setting was preserved. This idea was tested and it was found that the current before and after triggering was reduced by about 2 A. However, the board mergers still got stuck in readout. A second suggestion was offered by Venelin which was to more closely examine the registers related to readout order. These are `NTRO` and `NRRO` which refer to the NI Trigger Readout Order and NI Raw data Readout Order respectively. It was found that the readout order differed significantly between the old and the new configuration for the board mergers. In the new configuration, the NTRO register instructed the board merger MCM to readout internal data before any other port. Since the board merger does not have any internal data of its own, this was leading to the error. Furthermore, it was discovered that the TRAP configuration builder had a bug such that this faulty NTRO value was only written to C1 chambers. This explains why the issue was not seen at the University of Cape Town where a C0 chamber was tested and also why stack 2 did not present any of the board merger errors that were seen in all the other stacks with `rstate`. This issue was patched in the TRAP configuration builder by using the old values for NTRO and NRRO. Several runs were done with the fixed configuration at P2 and the current consumption was found to be nominal and matched what was seen with the old configuration.

[11] Due to the large amounts of ionizing radiation passing through the detector electronics, bits in memory may actually be flipped on rare occasions. Therefore, it is necessary to protect against this by refreshing registers with correct values at regular intervals during data taking.

Chapter 4

TRD Calibration

The next three chapters describe work done toward TRD calibration for Run 3. The collaborative work of Alexander Schmah, Ph.D., Anastasia Berdnikova, Ph.D., and the author is documented. In this chapter, the development of the calibration software is discussed as well as initial testing. This will include detailing the way in which the software implementation works and how calibration parameters were produced (see section 4.2). It will also include a detailed description of the principles behind tracklet reconstruction (section 4.3) and an investigation into the ion tail effect (section 4.5) as it relates to the calibration outcome. In the two chapters following this one, the performance of the test calibration is examined in more detail.

4.1 Objective

In practical terms, the purpose of drift velocity (v_D) and Lorentz drift ($E \times B$) calibration is to measure the macroscopic[1] drifting velocity of ionized electron clusters through the gas volume and to correct for the disturbance of the tracklet angle by physical phenomena local to the TRD. This is done so that the trajectory of ionizing particles through the TRD may be reconstructed accurately. This is essential for efficient tracking and meaningful physics analysis. In general, it is expected that TRD tracklets should be reasonably well aligned with TPC and ITS tracks. In the absence of proper calibration, this will not be the case. One of the primary reasons for this is the magnetic field that saturates the ALICE detector. This magnetic field causes the electron clusters in the TRD drift chamber to move at an angle rather than vertically upwards towards the pad plane. This is simply a consequence of the Lorentz force law which states

$$F = q\,(E + v \times B) \tag{4.1}$$

Therefore, electron clusters are subject to components of force in the local x direction from the electric field in $-x$ and in the local y direction from the magnetic field in z. These sum together to result in a direction of motion of the clusters somewhere in the region of $8°$ from the vertical (see Figure 4.1). This force causes an initial acceleration of the clusters, but they reach a terminal as the energy from the electric field obtains an equilibrium with the energy lost to collisions with gas molecules. As well as knowing the angle that the clusters are drifting at, to be able to reconstruct TRD tracklets, it is also necessary to know the drift velocity. The drift velocity is constant in the drift region due to an equilibrium that

[1] Of course, on the microscopic scale, the motion of the individual electrons is chaotic, but the electric field applied to the TRD drift volume causes a net flow in one direction. The quantity v_D refers to this average motion.

is achieved between the energy input to the system from the electric field and energy lost through collisions of the electrons with gas molecules [32]. Fluctuations in the temperature and pressure of the gas within the drift chambers, therefore, cause variations in v_D over time and small differences in mechanical structure and imperfections in electronics result in variation among TRD chambers [33]. Once these two pieces of information are known, the correct spatial positions of clusters can be reconstructed from the timebin and pad coordinates of digit data. Consequently, accurate tracklets may be determined.

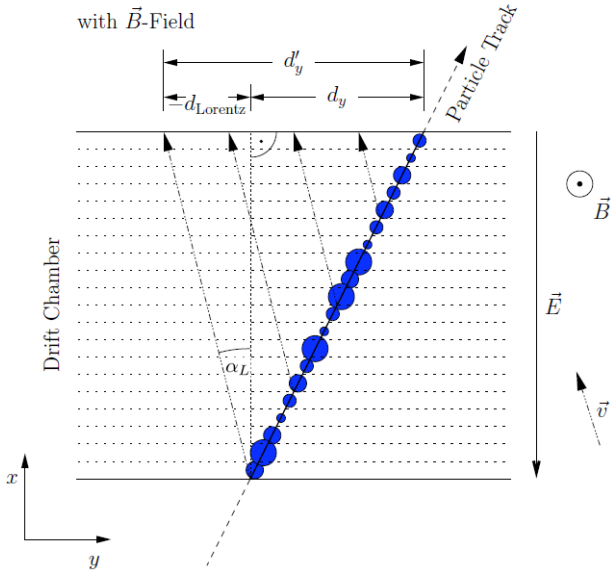

Figure 4.1: The Lorentz force acting on a tracklet. The primary clusters produced by the incident charged particle are shown in blue. Since the clusters experience a force in $-y$ ($E \times B$) direction due to the magnetic field, they drift at an angle upwards and to the left. Therefore, if they were reconstructed vertically downwards, they would appear to the left of the original clusters. The quantity, dy, known as the deflection length is shown above indicating the contribution from the tracklet angle as well as the Lorentz drift. It is important to note that the above diagram describes the velocity in the direction of motion of the clusters, however, the new convention is to give v_D as the vertical component of the drift velocity [25].

4.2 Calibration Software

The Run 3 v_D and $E \times B$ calibration algorithm was developed and tested in Aliroot[2] on Run 2 data. This can be understood as the research and development phase before the final software implementation is done in O^2. The data was taken from run 265338 in 2016 and was acquired from the Offline Conditions Database (OCDB). All code may be found in the `TRD-Run3-Calibration` repository on Github under the user `aschmah` [34].

The calibration procedure that was developed consists of two stages. Put simply, the first stage is the calculation of the residuals of offline tracklets with respect to tracks and the second stage is the extraction of the calibration parameters from the residuals via a physics-encoded fitting algorithm. Two distinct pieces of software are used in each stage. The

[2] Aliroot was the old ALICE software framework used during Run 1 and 2 now being replaced by O^2.

high-level flow of the full calibration procedure is visualized in the flow chart in Figure 4.2.

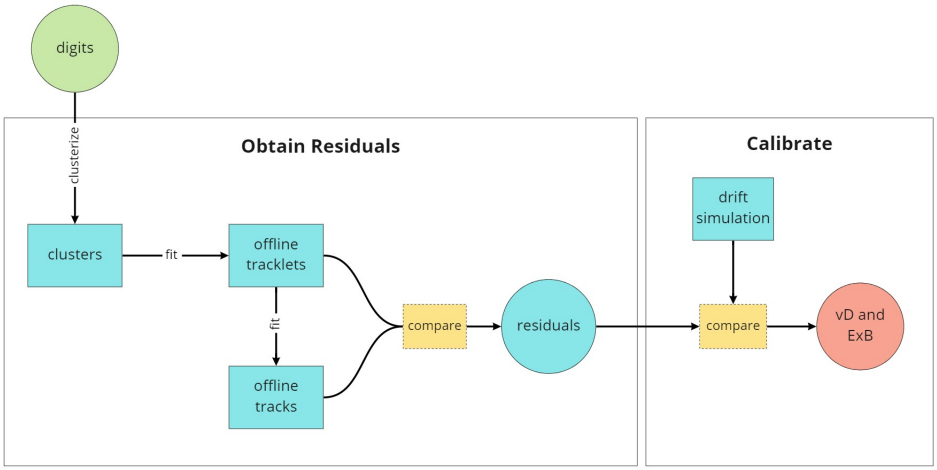

Figure 4.2: Overview of the v_D and $E \times B$ calibration approach implemented in the Aliroot software framework.

The calibration procedure produces two numbers per TRD chamber that are the drift velocity measured along the local x axis and given in units of cm/s and the Lorentz angle measured from the vertical (see Figure 4.1) and given in units of radians. It is necessary to calibrate each chamber individually since small variations in position, electric field strength, and environmental conditions can produce changes in the motion of the clusters.

4.2.1 Producing Residuals

From within an Aliroot environment, running `root Macro_GUI_TRD_Calib.C` from the root directory of the calibration repository will initialize the calibration software responsible for producing the residuals and launch a small GUI with several functions. Furthermore, this command will instantiate the primary `TBase_TRD_Calib` class and load the raw digit data. The constructor for this class serves several purposes:

- The names of the output files for the residuals histograms are declared.
- Quality Control (QC) data related to high-voltage measurements is loaded from a local ROOT file.
- The `TEve` event display is drawn with a 3D model of the TRD barrel based on the dimensions in the Aliroot geometry class.

At this stage, one of the class methods reads a text file containing a list of ROOT files which store the digit, TPC tack, and TRD online tracklet[3] data to be processed. The text file should be located in the top level directory of the repository, at the same level as `Macro_GUI_TRD_Calib.C`. The function will proceed to load the data from each file in the list and output some debugging information about which files were loaded and how many events were found. Events are loaded into memory sequentially. The first event is loaded at this time. Lastly, the number of timebins for which to calculate clusters during the clusterization

[3]Online tracklets are those which are calculated by the TRAP during data taking as opposed to the custom (offline) tracklet fits discussed later in section 4.2.1.

procedure is also declared. This procedure is explained in the next section.

From the GUI itself, several functions may be selected. Some are directly involved in the calibration process, while others are only used for debugging. To begin calibration, the `Calibrate` button should be clicked. Using the `Calibrate` function will produce residuals based on offline tracklets which is the default method[4]. Described briefly, raw digits are loaded and clusterized. Then offline tracklets are fitted to the clusters and global circle fits are done to the tracklet offset points. Once both sets of fits are done, the program compares them and residuals can be obtained. More detail is described below.

The `Calibrate` function accepts no arguments and returns void. The output residuals are written directly to a ROOT file instead. At the top level, the calibration procedure will loop over events contained in the provided ROOT data file. The range of the loop will determine the number of events from which the program should collect statistics used to measure the residuals. For enough statistics, it was found that at least 20 000 events were needed, but some calibrations were done with as much as 50 000 events. Tracklet multiplicity and p_T cut is also a factor that will influence the appropriate number of events to choose. For debugging purposes, the event number is logged every 100 events. For each event, several stages of processing are done.

Clusterization

All TPC tracks in the event are looped over with a cut being made on track p_T. For most of the testing, the cut was made at 1.5 GeV. Next, the clusterization process is carried out for each set of digits associated with the track. This is the process where raw digit data that may have entries across multiple pads in each timebin, are merged into a single cluster with a well-defined pad position for each timebin. Clusters typically span 2-3 pads with some exceptions. This cluster will take the form of a point in space and time that represents the weighted mean of the digit data in the corresponding timebin. This is done in the `TBase_TRD_Calib::make_clusters` function.

First, digits in pad - time bin coordinates are converted to spatial points. The coordinates are first calculated in the local chamber coordinate system before being transformed into the global coordinate system using a set of functions provided by the Aliroot geometry class which contains information about the location and orientation of each TRD chamber. The local x position of the digit is obtained using the drift velocity from a previous calibration. Similarly the local y position takes into account the Lorentz angle from a previous calibration. The v_D and α_L (Lorentz angle) parameters used are not important. The algorithm simply requires that the clusters be reconstructed to some positions in spatial coordinates so that initial residuals can be measured. Later, during the calibration fit, the old parameters will be taken into consideration when determining the new parameters needed to reconcile the reconstructed clusters with those from primary tracklets. This will be discussed in more detail later. Once the digits have been successfully transformed into space points, a quality cut is made. Digits which are greater than a distance of 3 cm away from the track at the closest point in ϕ direction are excluded. This is known as the Distance of Closest Approach (DCA). The results from official TRD QC are also taken into account here. If the detector in which the digit is found was flagged by the QC, it is skipped. The weighted fit to the digit data is done in the standard way using

[4]It is also possible to produce residuals based on online tracklets, but this is not discussed here.

$$x^i = \frac{\sum_{p=1}^{n} q_p x_p^i}{\sum_{p=1}^{n} q_p}$$

where x^i refers to the ith component of the position, x_p^i refers to the ith component of the position in pad p and q_p refers to the ADC value in pad p. n is the total number of pads in timebin t with digit data. This averaging is done in spatial coordinates. For q_p, a baseline correction of -10 is made. In addition to the weighted mean taken in each timebin, a final weighted mean is taken over the first 5 timebins and appended to the end of the cluster vector. This point is called the tracklet offset point and is used to perform the global circle fits. It was chosen as a good approximation of a point near the anode plane that the primary tracklet would have passed through that is not heavily affected by the $E \times B$ effect. The clusters are used for the tracklet fits.

Tracklet Fitting

With the clusters having been calculated, the tracklet fits may be done. Tracklets are fit to each group of clusters in each layer of the track. The tracklets are defined by two end points given in 3D global coordinates. The fit is done between the first and the last clusters that could be calculated during clusterization. For example, if there are 24 total timebins but no cluster was calculated for timebin 23 (where timebins are indexed from 0), then the fit will be done between clusters 0 and 22 (inclusive). Clusters are permitted to be missing from other timebins in between the limits of the range. In general, six parameters (two for each coordinate axis) are required to fully describe a line in 3D space, but only four of those parameters are independent. This means that with some knowledge about the properties of the line, two of those parameters can be fixed and the other four can be solved for instead. This has the effect of making the minimization process less computationally expensive. The initial guess for the direction of the tracklet line is taken to be the direction of the line pointing from the first endpoint to the second. Therefore, the choice of fit function is determined by a comparison of the distances between the two end points in the x and z directions. If the distance is found to be greater in x, then the fit function in which the 3D line is parameterized in terms of x is used and vice versa. This is to allow for cases where the line is parallel to either the $x-y$ or the $y-z$ plane and therefore could not be parameterized in terms of z or x respectively whose value would not vary. For clarity, an example of the parameterization in terms of z is shown in eq. 4.2.

$$x = x_0 + at,\ \ y = y_0 + bt,\ \ z = t \tag{4.2}$$

Where (x_0, y_0) is some point in the $x-y$ plane that is on the line and $\boldsymbol{u} =< a, b, c >$ would be a vector that is parallel to the line with $c = 1$ in this case of parameterization in terms of z. This is valid so long as the line is not parallel to the $x-y$ plane and therefore is guaranteed to have some extension in z. The endpoints of the line in z can then be selected arbitrarily and the other four parameters, x_0, y_0, a, b, which are passed to the minimization function, can be solved for. The endpoints in z are taken to be 0 and 1 for simplicity. During the minimization process, the distances between the clusters and the fit are computed using the dot product. This is shown in eq. 4.3 where $\boldsymbol{r_p}$ is the position of the cluster and $\boldsymbol{r_0}$ is any point on the line.

$$\boldsymbol{d} = (\boldsymbol{r_p} - \boldsymbol{r_0}) - \boldsymbol{u}\left[(\boldsymbol{r_p} - \boldsymbol{r_0}) \cdot \boldsymbol{u}\right] \tag{4.3}$$

In this way, the component of $\boldsymbol{r_p} - \boldsymbol{r_0}$ that is parallel to the line is subtracted off leaving only the perpendicular component which gives the shortest distance to the line. Each component

of the resultant vector $\boldsymbol{d}$ is then weighted according to the TRD resolution of the axis before being squared and summed. The resolution weightings were $0.725/\sqrt{12}$ for x and y and $8.5/\sqrt{12}$ for z. To apply the weightings, each component of $\boldsymbol{d}$ was divided by its corresponding weighting resulting in the x and y components being weighted ≈ 12 times higher than z. The distances for each cluster are summed up, this time weighted according to ADC value and the final sum of distances is returned as the minimization value. This value is minimized by an instance of the ROOT `TVirtualFitter` class. This was implemented by Dr. Schmah.

Global Circle Fit

Next in the track loop is the calculation of the global circle fit. A custom track fit is used instead of the TPC tracks because the TPC requires calibrated TRD tracklets before it can, itself, be calibrated [35]. The circle fit is done over tracklets in multiple layers. The first condition is that there are at least 3 layers with a tracklet to be used for the fit. If this condition is not satisfied, then no fit is done and the tracklets are discarded from the remainder of the calibration procedure. The circle fit takes three parameters, R and (a, b) which are the radius and centre of the circle in the x - y plane respectively. The circle is therefore only 2-dimensional and does not have any extension in z. Separation of the motion of the particle in the $x - y$ plane from its motion in z direction was determined to be a sufficient approximation during this phase of development of the calibration software[5]. The sum of the squared distances between the circle defined by these parameters and the offset points of the tracklets in each of the relevant layers is minimized in a similar way as during the tracklet fit.

Residuals

Once the circle fit has been determined, tangents to the circle fit are calculated at each tracklet offset point. To be precise, they are calculated at the nearest point on the track along the local y direction. This is then used to calculate the $\Delta\alpha$ residuals which fill up the primary histograms used to calibrate the TRD tracklets. For each tracklet-tangent pair, $\Delta\alpha$ is defined as the difference in the angle between the tracklet and the tangent as represented by two 3-dimensional vectors. Therefore, the angle is measured in the plane defined by the two vectors. Since the track can be thought of as the most accurate representation available of the true trajectory of the particle, well calibrated tracklets should be found to align with the tracks. In Figure 4.3, each of the intermediate components used in determining $\Delta\alpha$ can be seen.

The histograms are written to a ROOT file which is then used to calculate the calibration parameters. An example of the $\Delta\alpha$ histograms are shown in Figure 4.4 as a function of impact angle. The impact angle is defined as the angle of the circle tangent relative to the local y axis.

As can be seen in Figure 4.4, the tracklets are, in general, poorly aligned with tracks as is indicated by the systematic deviation of the majority of the distribution from the $\Delta\alpha = 0$ line. With enough statistics, the distributions are expected to cross the line at the Lorentz angle that was used during the clusterization procedure. Primary tracklets with this impact angle will result in clusters that drift along the line of the tracklet. Applying the appropriate

[5] After being ported to O^2 the measurement of residuals is done against 3-dimensional helix tracks from global tracking.

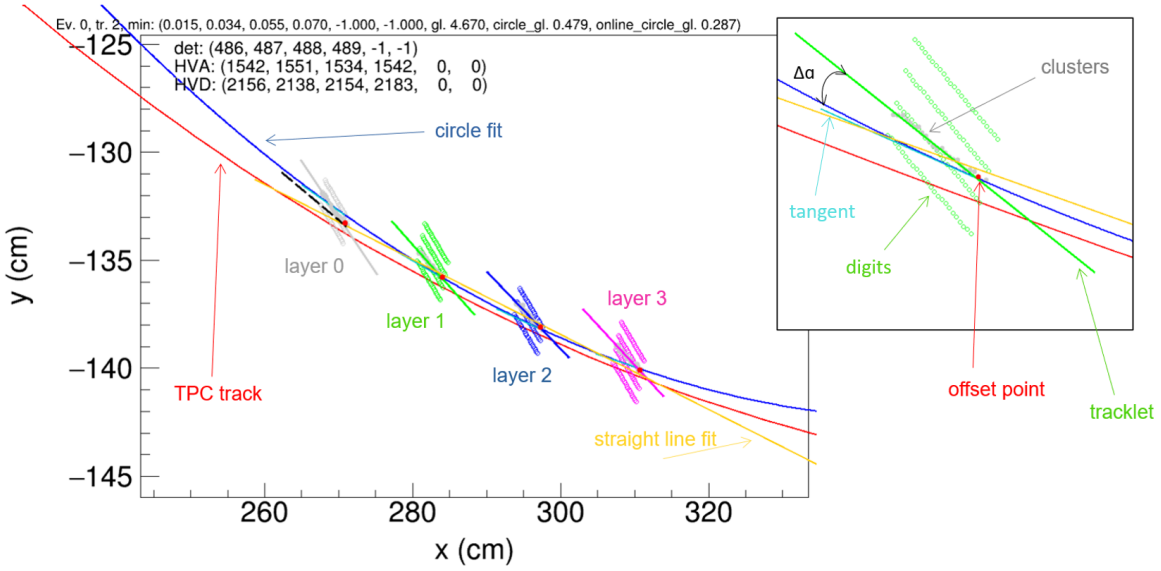

Figure 4.3: A single 4-layer track in sector 16 is shown including associated digits and raw tracklets. An enlarged view is shown of layer 1 with the angle $\Delta\alpha$ labelled. Note that $\Delta\alpha$ is measured between the tracklet fit and the tangent (light blue) drawn to the circle fit (dark blue). The coordinates are given in the global coordinate system.

Lorentz shift in y direction will then place them back in their original position on the primary tracklet.

4.2.2 Calculation of Fit Parameters

In the second stage of the calibration process, the fit parameters, v_D and α_L, are extracted from the $\Delta\alpha$ residuals. This is done by a program called `GUI_Sim_drift`. Running it with `root` will launch another GUI with more key calibration functions. Running `Do Minimize` will begin the fitting procedure and will attempt to produce fit parameters for all 522 chambers. Chambers that were flagged as defective by QC will be skipped. Other cuts are also made on the following conditions:

- There are at least 10 entries in the $\Delta\alpha$ histogram for this chamber.
- $E > 100$ V/cm. Where E is the electric field approximated as the ratio of drift HV to the height of the drift region.
- Anode HV > 1500 V.

Then, for each chamber, the sum of the differences between two different $\Delta\alpha$ measurements are minimized. This is done through a process of comparing simulated calculations of $\Delta\alpha$ for given pairs of v_D and α_L with the measured $\Delta\alpha$ from the histograms in the previous stage of calibration. Then, by minimizing the difference of this comparison, the correct values of v_D and α_L can be determined. Essentially, what needs to be solved for is the transformation $\boldsymbol{T}$ such that for a raw cluster in pad - timebin coordinates, $\boldsymbol{y_r}$, and a corresponding cluster on the primary tracklet in spatial coordinates, $\boldsymbol{x_p}$, the transformation can be applied to obtain

$$\boldsymbol{T}(v_D, \alpha_L)\boldsymbol{y_r} = \boldsymbol{x_p} \tag{4.4}$$

That is to say that a raw cluster may be reconstructed using some transformation, $\boldsymbol{T}$, at the correct original position of the corresponding primary cluster from the same time bin.

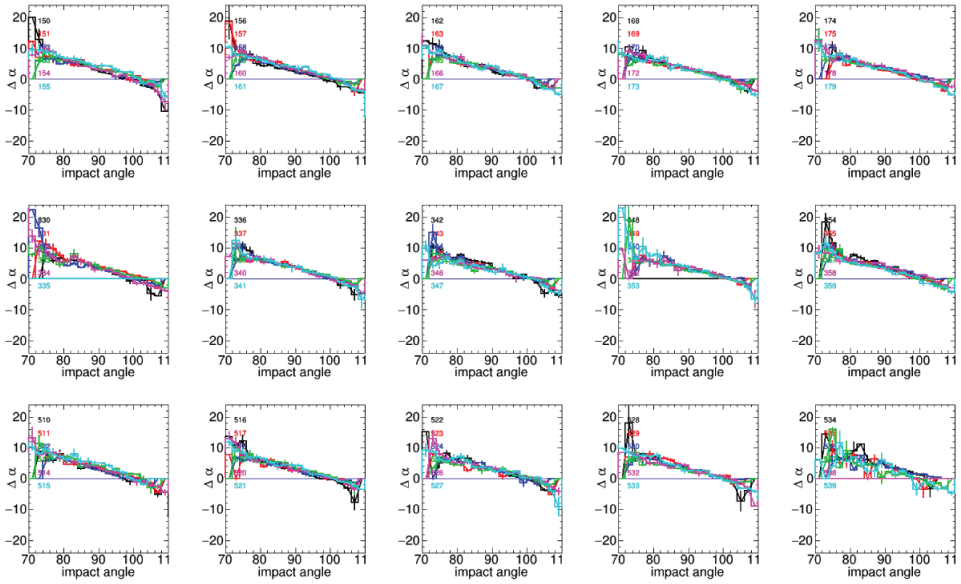

Figure 4.4: Histograms of differences in angle between tracks and matched tracklets ($\Delta\alpha$) for a selection of TRD chambers. Each plot displays residuals from 6 chambers. Detector numbers can be seen written along the top left edge of each of the plots. A blue line is drawn at $\Delta\alpha = 0$ indicating the desired shape of the histograms.

This is done via two intermediate calculations. The first is the determination of the initial reconstructed cluster, $\boldsymbol{x_1}$ in spatial coordinates, during the clusterization and tracklet fitting stages where the old calibration values, v'_D and α'_L are used. In a sense, this may be thought of as a guess toward $\boldsymbol{T}$ although it is not intended to be precise. This transformation may be expressed as

$$\boldsymbol{A}(v'_D, \alpha'_L)\boldsymbol{y_r} = \boldsymbol{x_1} \tag{4.5}$$

Where, in general, $\boldsymbol{x_1} \neq \boldsymbol{x_p}$. Then, a second transformation, $\boldsymbol{B}$ is determined by the fitting algorithm such that

$$\boldsymbol{B}(v_D, \alpha_L, v'_D, \alpha'_L)\boldsymbol{x_p} = \boldsymbol{x_2} \tag{4.6}$$

Where $\boldsymbol{x_p}$ is taken from hypothetical tracks for a range of impact angles corresponding to the range for which residuals were collected. Parameter $\boldsymbol{x_2}$ is also given in spatial coordinates. This means that the measured $\boldsymbol{x_1}$ (encoded in $\Delta\alpha$) can be looked up in each case. The fitting procedure is then essentially an attempt to enforce $\boldsymbol{x_1} \stackrel{!}{=} \boldsymbol{x_2}$ by varying v_D and α_L appearing in $\boldsymbol{B}$. Then, once $\boldsymbol{B}$ is known, $\boldsymbol{T}$ is simply given by

$$\boldsymbol{T} = \boldsymbol{B}^{-1}\boldsymbol{A} \tag{4.7}$$

The exact minimization parameter is a sum over all impact angles $80° \leq \theta \leq 100°$ and is calculated as

$$\sum_{\theta=80}^{100} \frac{[\Delta\alpha_{\text{sim}}(\theta) - \Delta\alpha(\theta)]^2}{\sigma^2} \tag{4.8}$$

Where $\Delta\alpha_{\text{sim}}$ is the calculation of $\Delta\alpha$ that is obtained from the simulation and σ is the statistical error in $\Delta\alpha$. Once this difference is minimized, the parameters that would be necessary in order to produce the observed residuals, have been determined. Errors for the

two fit parameters are extracted from the minimization function. If no fit is determined, then v_D defaults to $v'_D/1.05$ and α_L defaults to $-7.5°$.

In this way, the calibration is determined. The output of the procedure is a ROOT file containing two graphs of α_L and v_D as functions of detector number. For test calibrations, the parameter α_L was found to vary around 8° to 10° across the TRD. The parameter v_D was found to vary between 0.9 and 1.6 cm/μs with a high concentration of results near 1 cm/μs. This was discovered to be much lower than typical calibration results from Run 2 which were in the region of 1.5 to 1.6 cm/μs. A possible explanation for this is explored in section 4.5.

4.3 Reconstruction of Calibrated TRD Tracklets

After the calibration algorithm had produced the fit parameters of v_D and α_L, it was necessary to test them. If the algorithm had worked and produced the correct parameters, then these parameters could be used to re-orientate the tracklets by transforming them in accordance with the method in the calibration algorithm. This should produce new tracklets which should be aligned with reference TPC tracks. An implementation was created to transform online tracklets to this end. This method is applicable to the offline tracklets as well since they obey the exact same principles. To begin, it is necessary to first understand the way in which varying v_D and α_L will transform a given tracklet. This is described below within the local chamber reference frame[6].

The Lorentz angle can be understood as a rotation of the tracklet around the offset point where the primary tracklet meets the cathode plane (see Figure 4.1). However, note that the angle by which the tracklet is rotated is not, in general, equal to α_L since α_L is measured from a point directly above the latest reconstructed cluster and not from the offset point. The rotational effect on the tracklet is therefore inversely proportional to the angle of the tracklet and is only equal to α_L in the case of a 90° tracklet impact angle. Since the tracklets follow a straight line, to draw them, only the offset point and the position of the last cluster are needed. Therefore, only the last cluster needs to be transformed and then the tracklet line can be drawn by connecting the last cluster to the offset point. The Lorentz angle has the effect of shifting the last cluster horizontally by an amount,

$$d_L = \pm h \tan(\alpha_L) \tag{4.9}$$

where $h = 3$ cm is the height of the cathode plane as measured from the start of the drift region and the sign depends on the direction of the B field. A B field in the $+z$ direction (out of the page as in Figure 4.1) will incur a $-$ sign while a B field in the $-z$ direction will incur a $+$ sign.

Changes to the drift velocity will have the complementary effect of moving clusters on the vertical axis. Since the x position of reconstructed clusters is proportional to $v_D t$, increasing v_D will shift clusters downwards while decreasing v_D will move them upwards. Therefore a v_D ratio of $\frac{v_D}{v'_D} > 1$ will signify an increase in v_D and thus a downward shift of the clusters and a ratio of $\frac{v_D}{v'_D} < 1$ will have the opposite effect. A ratio is considered since the

[6] In this section, tracklets are understood at 2-dimensional lines in the local chamber x - y plane. In reality, they also have some extension in z although before tracking, information regarding this parameter is strictly limited. A more accurate understanding would be to think of them as x - z planes with an extension in z determined by the uncertainty equal to one pad length.

clusters comprising the online tracklets are already in Cartesian coordinates and therefore have already made use of a value for v'_D in their calculation. The x position of the clusters is scaled by the drift length and can be expressed as

$$d_{v_D} = h\frac{v_D}{v'_D} \tag{4.10}$$

These two transformations are both non-commutative and the Lorentz angle transformation is non-associative as well. These transformation principles are encoded into the drift simulation used to determine calibration parameters. The calibration algorithm transforms tracklets in the following sequence: it first selects the point of intersection between the primary tracklet and the start of the drift region (black dotted line at $y = 0$ in Figure 4.5). This is the point (0,0). Beginning from this point, a line is then drawn with a slope determined by α_L. The black star in Figure 4.5 indicates the point of intersection between this line and the anode plane. The horizontal distance between the first and second point is equal to d_L from eq. 4.9. Next, the point is shifted down by the quantity d_{v_D} from eq. 4.10. This new position is marked by the cyan star. This will be below the beginning of the drift region if $v_D > v'_D$, otherwise it will be above. Lastly this point is shifted horizontally by an amount equal to $d_L{}' = \mp h\tan(\alpha'_L)$.

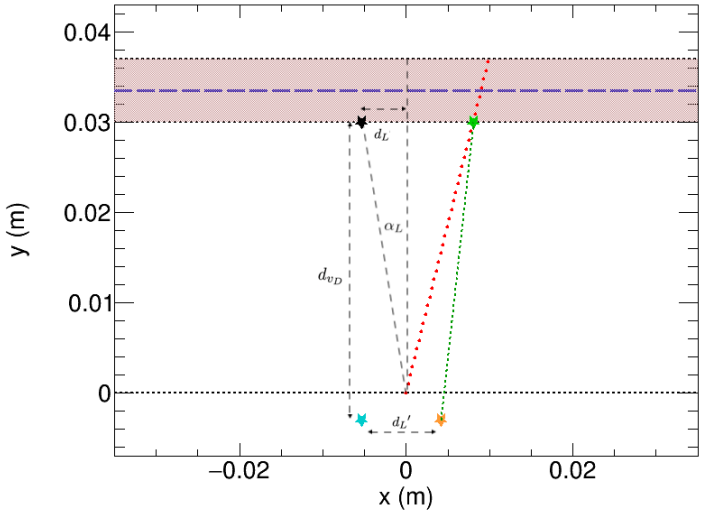

Figure 4.5: Tracklet transformation in the calibration algorithm. The primary tracklet is shown in red while the reconstructed tracklet is in green. The stars indicate calculations intermediate to the reconstruction process. The order of the transformations is from green to black, cyan, and finally orange. [34].

With this geometry correctly understood, the inverse transformation can be applied to transform the reconstructed tracklet back into the primary tracklet. The online tracklets are defined by an offset point and a slope. This slope is given in a global coordinate system and needs to be transformed into local chamber coordinates. A standard Cartesian rotation is applied using

$$\begin{aligned} x &= x'\cos(\theta) - y'\sin(\theta) \\ y &= x'\sin(\theta) - y'\cos(\theta). \end{aligned} \tag{4.11}$$

Where the primed coordinates are those in the global reference frame. Here, θ is determined

by the sector number using

$$\theta = 90^\circ - (20^\circ s + 10^\circ). \tag{4.12}$$

Where s refers to the sector number. With this information, the inverse transformation to obtain the corrected cluster position, (x_r, y_r), can be expressed as

$$x_r = h\tan(\alpha'_L)\frac{v_D}{v'_D} - h\tan(\alpha_L), \tag{4.13}$$

$$y_r = h(\frac{v_D}{v'_D} - 1) \tag{4.14}$$

keeping in mind that for equations 4.13 and 4.14, the origin of the coordinate system is where the reconstructed tracklet intersects the beginning of the drift region. Therefore, a final transformation back into global coordinates is required. Since the offset point of the online tracklets is already known in global coordinates, this transformation is only applied to the resultant tracklet angle which can be derived from the corrected cluster position. An example of some reconstructed tracklets are shown in Figure 4.6.

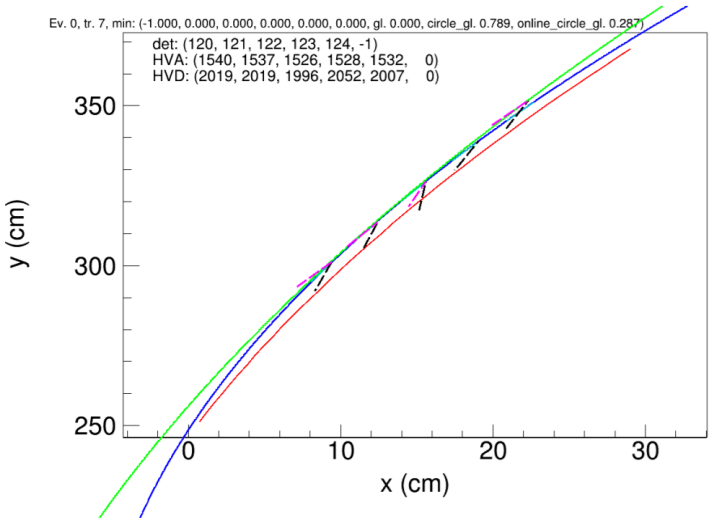

Figure 4.6: Online tracklets reconstructed using calibration parameters. Original uncalibrated tracklets are shown in black and calibrated tracklets are shown in pink. In green and blue are circle fits to the tracklet offset points and in red is the associated TPC track. Tracklets in defective chambers are not drawn.

In Figure 4.6, reconstructed tracklets can be seen to align better with tracks than their uncalibrated counterparts. This example was found to be representative of many that were looked at and an overall improvement was observed. Of course, there is variation among tracklets and only the mean of $\Delta\alpha$ can be minimized so not all tracklets will be perfectly aligned.

4.4 Closure Test

In addition to testing the calibration on the online tracklets, it was necessary to analyse a larger-scale test at the digit level. During the clusterization stage, the calibrated values of v_D and α_L could be used instead of the values from a previous calibration. If the calibration

was performed correctly, then the $\Delta\alpha$ histograms should be found to be flat and close to 0° as the difference between the angle of the tracklets and the tracks should now be minimized. This was done for the same data set from 2016 and the results are shown in Figure 4.7.

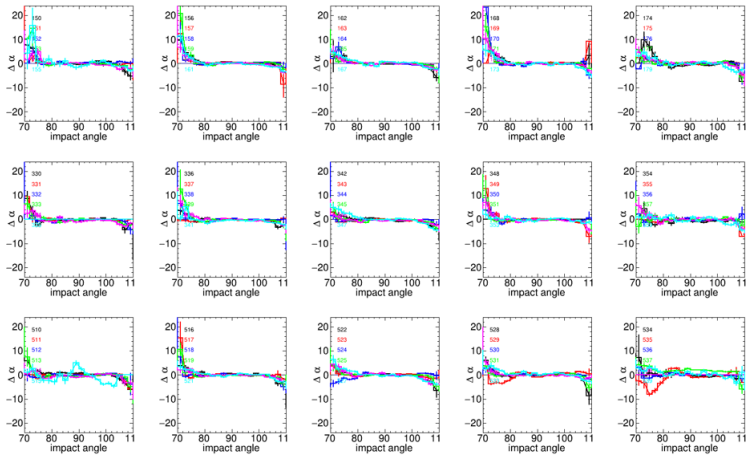

Figure 4.7: Histograms of differences in angle between tracks and matched tracklets ($\Delta\alpha$) after the calibration has been applied to the tracklets for the same selection of TRD chambers as in Figure 4.4.

It can be seen that the histograms are much flatter than what was seen in Figure 4.4 earlier. There is a noticeable loss in performance at the edges of distributions for low p_T tracks. This is partly the result of limiting the minimization to only between 80° and 100°. The plots show a range of between 70° and 110° which is 10° more on either side. Statistics are also lower in these regions. Furthermore, tracking is more difficult at lower p_T and the effect from ion tails becomes less significant even though it is already accounted for in the calibration in the case of higher p_T tracks which are more common (see section 4.5) [35].

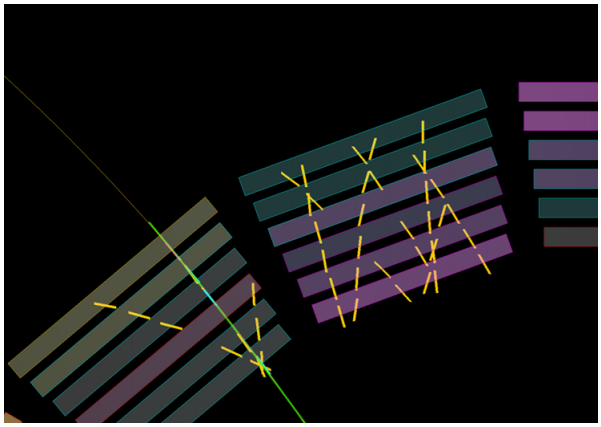

Figure 4.8: A visualization of calibrated tracklets in the alternative `TEve` display. Two sectors are shown with tracklets in yellow. The track shown in green can be seen to align well with the matched tracklets.

4.5 Ion Tail Analysis

An investigation into the effect of a lack of ion tail cancellation on the angle of reconstructed tracklets was conducted. This was suggested as a possible explanation for the large discrepancy that was observed between results from the new calibration and those from Run 2. To be more exact, old measurements of v_D were found to be, on average, a factor 1.4-1.5 higher than recent measurements [36].

To this end, a simple simulation was created to test the ion tail effect. Two phenomena were encoded. The first is related to charge sharing between adjacent pads and the second is the distribution of the absorbed charge as a function of time, also known as the ion tail.

- Time Response Function (TRF). Any point charge deposit inside the gas volume of the chamber will diffuse as it drifts towards the pads. This effect, combined with the particular response of the PASA to incoming signals, results in a spreading out of the ADC signal over time. The function describing this effect is plotted in Figure 4.9.

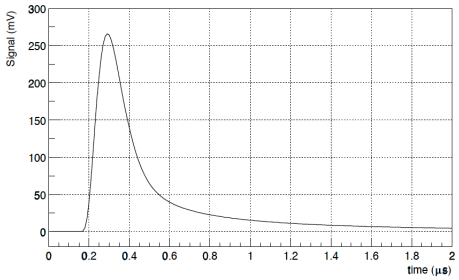

Figure 4.9: The TRF showing the signal measured by a pad as a function of time [14]. Similar in shape to a Landau distribution, a large peak can be observed followed by a long tail. Although the tail is low, it can result in a noticeable effect when compounded with additional tails from subsequent ions.

- The Pad Response Function (PRF) describes the distribution of the induced signal across adjacent pads. This behaviour is by design and allows for sub-pad position resolution of electron clusters by considering the relative signal among groups of pads. The PRF is plotted in Figure 4.10.

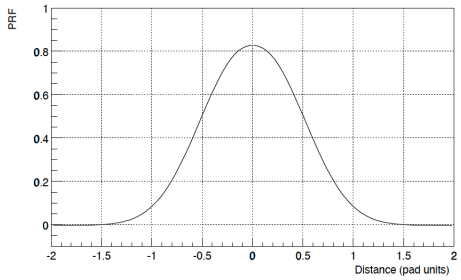

Figure 4.10: The PRF describing the way in which charge is shared between neighbouring pads [14]. It is roughly Gaussian in shape and shows that any signal induced in one pad will also be induced to a lesser degree in both adjacent pads.

An application called `xyscan` was first used to digitize the distributions of the TRF and PRF seen in figures 4.9 and 4.10 respectively [37]. The program loads an image of the distribution and is then able to construct a pair of axes based on limits defined by the user. Points on

the image can then be extracted by intelligently mapping pixel coordinates to the coordinate system defined by the axes. The output of the program is an array of (x, y) pairs in text format that can be copied directly into the code.

A model of the TRD drift chamber is drawn with 24 time bins. Simulated primary clusters are first drawn in black (see Figure 4.11). Then each of the primary clusters are shifted along x (now, the horizontal axis) in accordance with the Lorentz angle affect and drawn in red. A default Lorentz angle of 7.5° is used. It should be understood that each cluster effects all other clusters that succeed it chronologically based on the TRF described above. As pictured in Figure 4.11, time advances in the $-y$ direction. Therefore, as one moves up the line of clusters from $y = 0$, the scope of effect of each cluster is all clusters that are below it up to a certain limit determined by the length of the TRF. For each of the red clusters, then, the time as well as the y position of each time bin of effect are calculated. The drift velocity is taken to be 1.546 cm/μs and one timebin is taken to be 100 ns. For each time bin of effect, two integrals are calculated in order to take into account both of the above mentioned effects. The first is the contribution from the TRF and is calculated by taking the integral of the TRF between two times t_1 and t_2. Time t_1 is the time in μs since the induced signal from the particular cluster began plus an offset of 0.3 μs which is approximated to be the peak of the TRF. In order to take the middle of the time bin, a further offset of -0.05 μs is applied which is half the length of one timebin. Then the upper limit of the integral is taken as $t_2 = t_1 + 0.05$ μs. As an example, for the first time bin of effect of a particular cluster i.e. the time bin of the cluster itself, the integral is calculated over the range 0.25 - 0.35 μs. Then, for the second time bin, the integral is done over the range 0.35 - 0.45 μs which is 100 ns, the length of one time bin, after the previous range. Calculated in this way, the range of significant effect of each cluster is limited by the TRF to approximately 18 time bins. The second contribution is from the PRF and is, similarly, calculated by taking the integral of the PRF between two points x_1 and x_2. The point x_1 is offset -0.5 pad units away from the pad currently being considered. The range of the pads considered is [-2, 2] with pad 0 being the pad that is directly above the current Lorentz shifted cluster. The contributions are then multiplied and filled into a 2D histogram in the appropriate bin as an ADC value. If the timebin being considered is one of the first 4 timebins, the ADC value is doubled to approximate the effect of the amplification region which experiences simultaneous absorption of charges from both above and below the anode wires.

Once the ADC values were calculated, a new set of clusters could be determined by taking the weighted mean of the ADC values for each time bin. A fit was then made to this new set of clusters and this was interpreted as the reconstructed tracklet. Any differences then between the primary and reconstructed tracklets would be due only to these two effects since v_D and α_L were kept constant. An image of the results is shown in Figure 4.11.
Comparing the primary (black) and reconstructed clusters (green) in Figure 4.11, the effect can be interpreted as a shifting down of the clusters and a reduction of the deflection length identical to the effect of an increase in v_D. Thus, we can calculate what v_D ratio would be required to produce a shift of this magnitude. This was done by looping over every integer impact angle in [80,100) degrees and feeding the tracklets into the calibration algorithm described earlier in this chapter. In this instance, the algorithm only had the v_D ratio as a free parameter while α_L was fixed. This would ensure that only the v_D ratio needed to compensate for this effect would be determined.

After running the algorithm, a result of v_D ratio = 1.677 was obtained. This overshoots the initial discrepancy of 1.4-1.5. This could be due to any number of factors related to

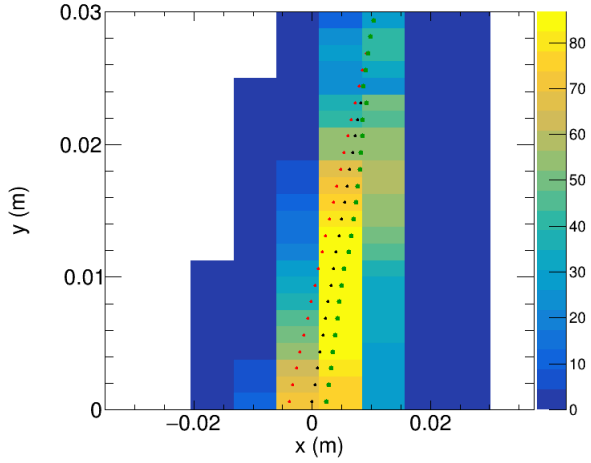

Figure 4.11: Example of ion tail simulation with an impact angle of 70°. Shown are the original primary clusters (black), Lorentz shifted primary clusters (red), and reconstructed clusters (green). Here, y is the time axis.

simplicity of the model. The new convention of defining v_D as the vertical component of the drift velocity rather than the velocity in the direction of motion of the clusters results in a negligible effect. Assuming a Lorentz angle of 8°, this would result only in a factor 1.01 difference. This effect has been noted in previous studies where it has been suggested that "The reconstructed deflection is, however, biased to smaller values by the ion tails from preceding clusters. This results in an over-correction of the Lorentz drift. The effect can be counteracted by understanding the configured drift velocity as an effective quantity, which is tuned to minimize the width of the deflection residuals." [38]. Therefore, it should be understood that the v_D that is determined by the new calibration procedure is merely an effective drift velocity that, when applied, minimizes the residuals of the tracklets with respect to tracks. It should not be confused with the real physical drift velocity of the electrons.

Chapter 5

Run 2 Quality Control

5.1 Quality Control Approach

In the previous chapter, the development of the Run 3 calibration software was documented. In this chapter, Quality Control (QC) is done on TRD chambers to better understand the performance of the calibration. After performing the test calibration and reconstructing the calibrated tracklets, it was clear that the performance of the calibration varied across the TRD. In some chambers, residuals were still poor. To understand this, it was necessary to look at a number of chamber-specific factors that could be expected to influence the quality of the calibration. From there, it would be possible to identify and eliminate problems in the calibration software. The chamber-specific factors that were considered are listed:

- Status according to the official QC.
- Whether or not a fit was determined (i.e. chamber is indeed calibrated).
- Whether there were sufficient statistics to determine a fit.
- Magnitude of measured Anode HV.
- Magnitude of measured Drift HV.
- Quality of ADC pulse height spectrum.

Each criterion is explained in further detail below. The information obtained is then referred to in the following chapter 6 when interpreting the calibrated tracklet residuals.

5.1.1 Official QC

Each chamber in the TRD undergoes official QC testing and this information is stored online [39]. A list is available consisting of 91 total chambers including those not installed (PHOS hole) that were flagged as problematic for various reasons most frequently being reasons related to low anode or drift voltage.

The official QC was considered to be authoritative and therefore any poor calibration results in the above flagged chambers were not taken as a concern. The majority had very low anode HV readings which resulted in unusable data. Further investigation revealed that there were additional issues that affected calibration quality and these are discussed in the proceeding sections.

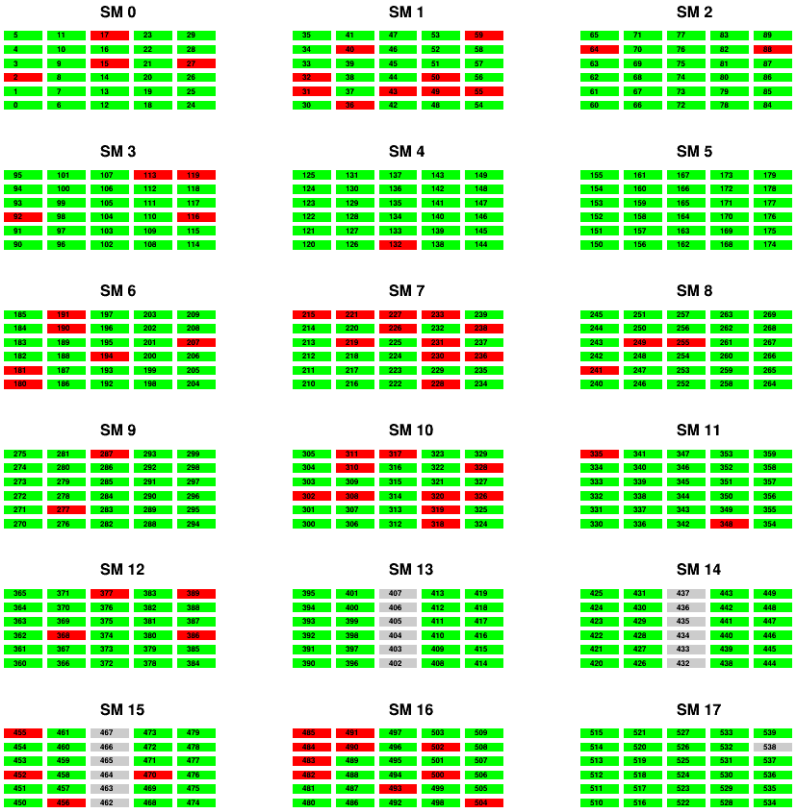

Figure 5.1: Official QC overview showing chambers where no QC issues were observed (green) and chambers where some QC issues were observed (red). Grey chambers are those which are not installed including the PHOS hole shown in stack 2 of sectors 13, 14, and 15.

5.1.2 Chambers Without Successful Fits

Uncalibrated chambers are those chambers for which no fit was determined when the calibration software was run. The result is that no tracklets in these chambers can be aligned and therefore they cannot be involved in the resolution or error analysis and cannot be used for tracking.

No fit was attempted on any of the chambers which were flagged by the official QC process. It was assumed that fitting would not be possible or, at the very least, unreliable due to poor data quality. In addition to the chambers flagged by official QC, chambers which did not have enough statistics could also not be calibrated. This is the most common reason, aside from official QC, for chambers not being calibrated. The reasons for discarding a given tracklet or track are summarized below:

- Tracklet occurs in a chamber flagged by official QC
- Tracklet is not part of a track which consists of 3 or more tracklets.
- Tracklet occurs in a stack with 4 or more layers flagged by official QC. This means that tracks will seldom consist of 3 or more tracklets except in rare cases where more than 1 stack is crossed.
- A circle fit could not be made to the tracklets.

- Digit ADC values were found to be too low and no tracklet fit was attempted due to the data being considered too unreliable.
- Associated TPC track $p_T < 1.5$

Initially, during the development of the calibration software, fits required tracks of 5 or more tracklets, but this was found to be too restrictive as entire stacks would remain uncalibrated if they consisted of more than 1 chamber that was flagged by official QC. With this requirement imposed, it was found that after running the calibration, 56 chambers would remain uncalibrated (excluding the 91 from official QC). By lowering this condition to only 3 tracklets, this number was reduced from 56 to only 7. Of these 7 chambers, 2 were found to be in a stack with only 2 functioning layers so no fits could be made in the stack. The other 5 all showed additional QC issues which will be shown in the proceeding sections.

5.1.3 High Voltage

High voltage QC involved creating plots of voltage as a function chamber number and identifying outliers using a straight line cut. In Figure 5.2, plots for each of anode and drift HV are shown. The data used was obtained from the OCDB for the relevant run number 265338 from 2016.

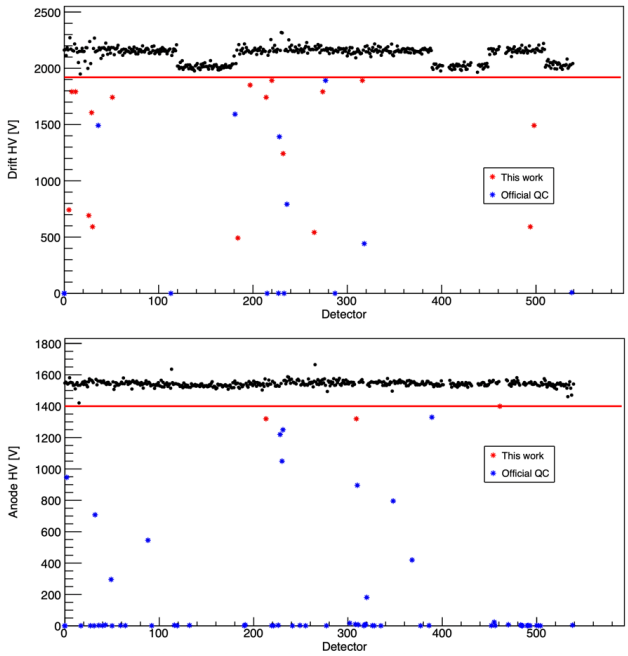

Figure 5.2: High voltage measurements across all chambers in both the drift (top) and anode (bottom) regions. The absolute value is taken. The cuts are shown as red lines at 1400V (anode) and 1920V (drift). The red stars indicate chambers which were flagged using this method, that had not previously been flagged during official QC.

Looking at Figure 5.2, appropriate cuts can easily be determined visually. For the anode HV, a cut of 1400 V was selected and for the drift HV, a cut of 1920 V was chosen. Therefore, all chambers which had a voltage measurement below these cuts were flagged. From the anode HV measurements, 80 chambers were flagged in total. This includes the chambers which were not installed and for which, therefore, no values were given. Of these 80, 3 were not in the official QC list. For the drift HV, 47 chambers were flagged in total. Of these 47,

17 were not in the official QC list. In total, across both sets of measurements, 104 unique chambers were flagged, 20 of which were not in the official QC list.

Looking at the distribution of red stars across both plots in Figure 5.2, it is surmised that the official QC used a slightly lower cut for the anode HV and did not use the drift HV as a strong indicator of a faulty chamber. This is based on the observation that the red stars in the anode HV plot all appear just below the red line, however in the drift HV plot, they are many and more evenly distributed. Regardless of whether drift HV was considered a strong indicator during official QC, for the purposes of calibration, it was found to result in poor fits. The reasons for this are discussed later in section 6.1.2.

5.1.4 Pulse Height Spectrum

Next, average ADC vs. time bin plots (pulse height spectra) of each chamber are examined. A typical example of what this spectrum looks like is shown in Figure 5.3 alongside an example of one pulse height spectrum from a chamber which was not flagged by official QC. Plots were made from statistics accumulated over 50 000 events from the same data file used for the calibration test.

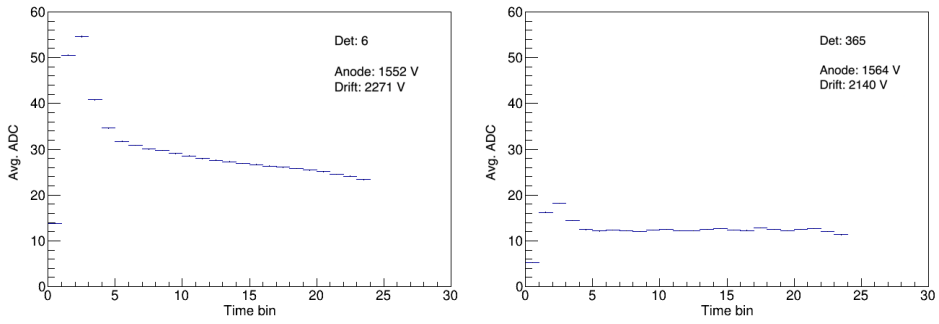

Figure 5.3: Typical example of a healthy pulse height spectrum (left, taken from detector 6) alongside a low pulse height spectrum (right, taken from detector 365). The good spectrum has a well-defined and prominent peak with high ADC values. The poor spectrum has a very low peak and low ADC values that are not reliable and could produce poor tracklet fits. Recall the baseline subtraction of -10 mentioned in chapter 4 which would result in ADC values close to 0 for the right-hand plot.

A distinction is made between two parts of the histogram; the peak and the plateau. The peak occurs within the first few timebins and is expected to have an ADC value approximately equal to twice the value of the plateau. This is because the first few timebins will contain signals from charges ionized both above and below the anode plane. The plateau should be nearly flat with little to no tail at the end. This will be the contribution from the electrons ionized in the drift region which drift, and therefore are received by the pads, at a constant rate. The tail should be removed by the tail cancellation filter in the TRAP.

Three histograms are shown in Figure 5.4 of the ADC values across all the chambers for a high number of events in two time bin windows as well as a ratio of the two. In them, many outliers can be seen both in the high as well as the low range. This is for all chambers without any chambers removed because of other QC issues.

From these histograms, typical values for each measured quantity can be estimated and then outliers can be eliminated. The cut was chosen to be at a mean ADC value over timebins

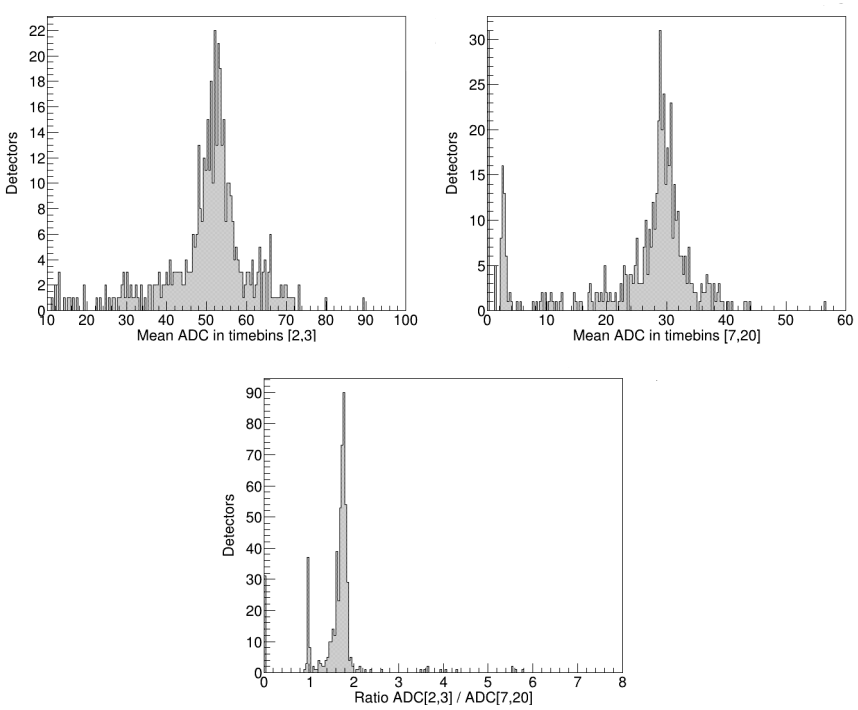

Figure 5.4: Histograms of mean ADC value per time bin over the range of time bins [2,3] (top left), mean ADC value over time bins [7,20] (top right), and the ratio of the mean over time bins [2,3] to the mean over time bins [7,20] (bottom).

[2,3] of between 35 and 70 so that any chambers which produced mean ADC values above or below this were flagged. The cut is visualized in Figure 5.5 by the red lines.

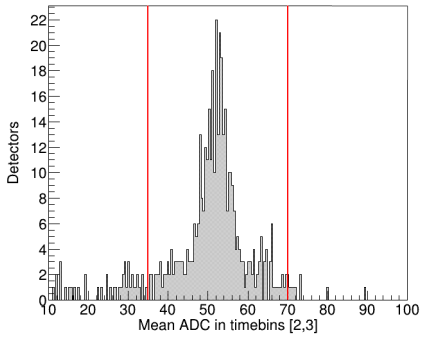

Figure 5.5: Histogram of the mean ADC value over time bins [2,3]. The red lines indicate the lower and upper cuts which were made.

There was shown to be a strong correlation between anomalous behaviour in the measurement over timebins [2,3] and the measurements over timebins [7,20] as well as the ratio of the two. This meant that the one cut alone was enough to exclude the majority of chambers which could be identified as defective based on grounds relating to the pulse height spectrum. No other cuts were made. A second set of histograms is shown after this cut was made and the defective chambers were removed from the data in Figure 5.6.

Looking at the plots shown in Figure 5.6, it can be seen that the majority of outliers are removed after making the appropriate cut to the [2,3] window histogram.

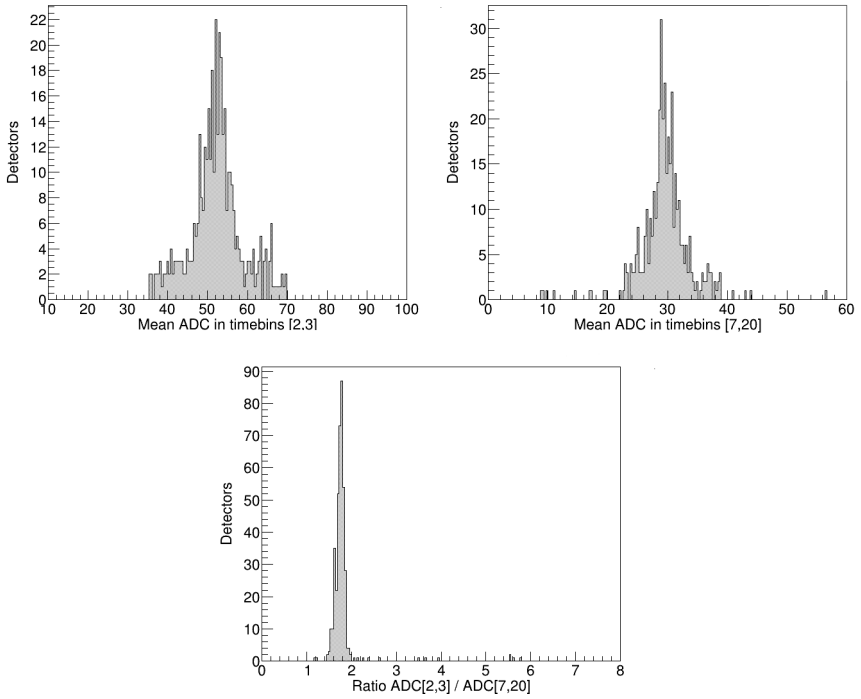

Figure 5.6: Mean ADC histograms after the cut has been made. Compare this with those shown in Figure 5.4.

.

5.2 Summary

Shown in Figure 5.7 is a summary plot of QC showing each chamber with either a red cross (flagged) or a green circle (not flagged) for each of the criteria mentioned in the previous section. In general, it can be seen that there is a good correlation between different criteria and columns of red crosses are often observed. ADC appears to be the least correlated with other items. This can be seen for instance from chambers 136 to 175 where there is a long row of ADC flags without any others. The validity of each criterion as an indicator of calibration quality is examined further in chapter 6.

5.3 QC Macro

A QC macro was written to flag chambers based on the criteria discussed above. With the cuts mentioned, the macro flags a total of 167 chambers. The macro outputs a vector of three-digit binary numbers of length 540 to a ROOT file called `chamber_QC_flags.root`. From least significant to most significant bit, each bit represents the result of QC relating to pulse height spectrum, anode HV, and drift HV respectively. If the bit in the given position is a 1, then the chamber was flagged based on the corresponding criterion. As an example, a decimal value of 3 would correspond with 011 in binary and this would indicate a chamber which was flagged based on analysis of the pulse height spectrum as well as the anode HV, although the drift HV appeared to be fine. The output ROOT file was used to inform other programs including the tracklet resolution macro discussed in the next chapter and the TRD Kalman filter. Information about the other two indicators, fit status and official QC, could be obtained from other sources.

This macro is contained in the Run 3 TRD calibration git repository at [34].

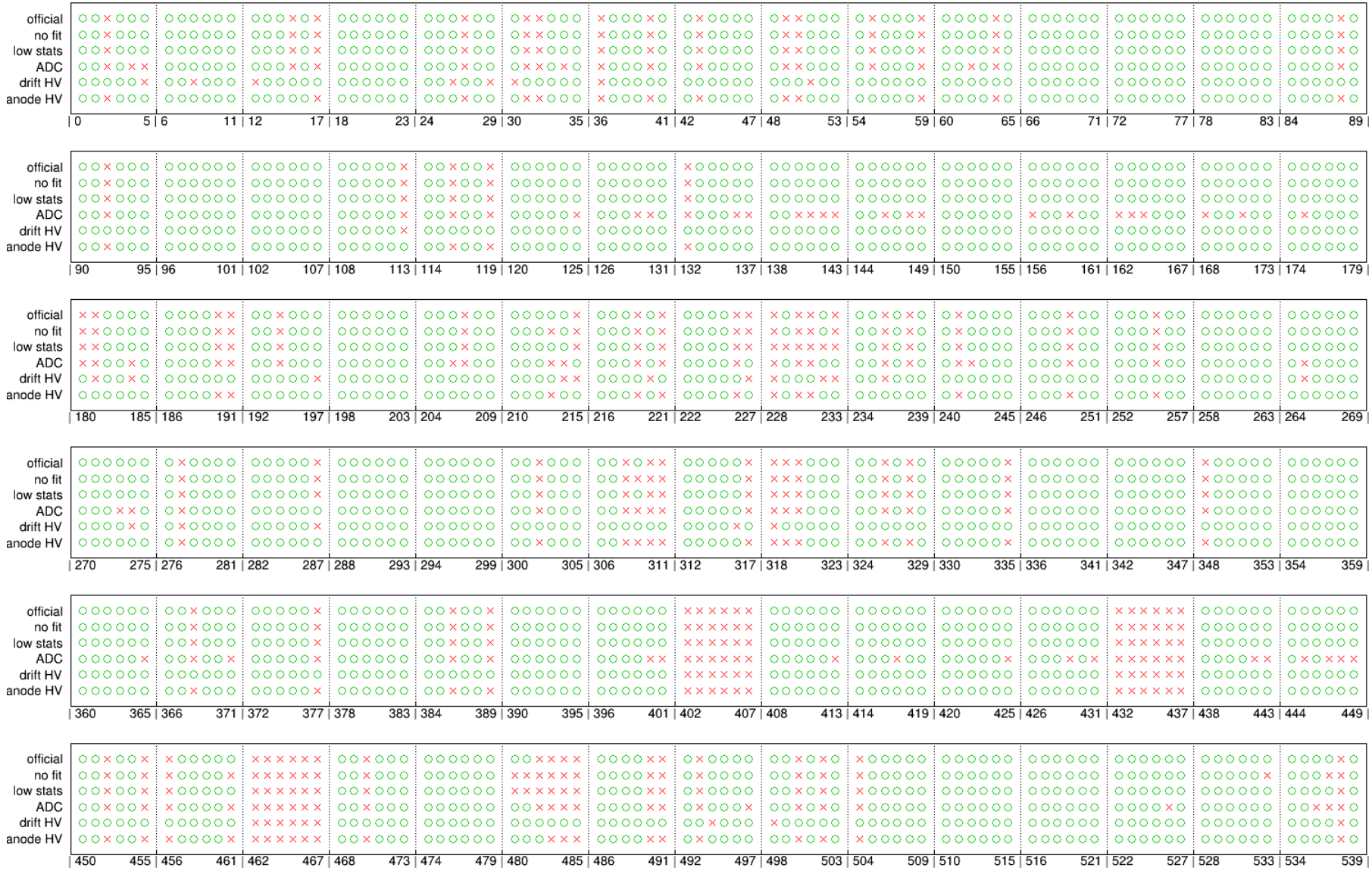

Figure 5.7: Summary of the QC results showing red cross (flagged) or green circle (not flagged) for each chamber.

Chapter 6

Tracklet Resolution

In continuation from the previous two chapters, the performance of the test calibration is now examined in more detail. Using the QC findings from the previous chapter 5, it can be determined whether a poorly calibrated chamber is the result of problems in the calibration software or if it is caused by detector-related QC factors that cannot be accounted for by the software. The chapter is divided into two parts. In the first, the angular resolution of the calibrated tracklets will be discussed, and in the second part, the position resolution.

6.1 Angular Resolution

First, the resolution of the tracklet angle is examined. This refers to the quantity $\Delta\alpha$ as it was defined in section 4.2.1. This is the primary result of the calibration.

6.1.1 2D Distributions

The goal of the calibration was to obtain a mean $\Delta\alpha$ of 0° across all impact angles. The deviation from 0° describes the error in the calibration and therefore the resolution.

First, a number of 2D histograms are plotted which show the distribution of $\Delta\alpha$ with respect to impact angle. These are shown in figures 6.1 and 6.2. The ideal case would be narrow distributions centred around 0°. Performance can be seen to vary among detectors and several observations are made:

The chambers which were flagged by the official QC are empty. This is because any tracklets found in these chambers are discarded and no circle fits can be done. Because of this, no residuals are collected and, therefore, no calibration fit can be attempted. Detector issues indicated by official QC need to be resolved before a calibration can be done.

Detectors 5, 26, and 30 have a very wide distribution not encountered elsewhere in the plots shown. These three chambers all had very low drift HV as indicated by the orange colouring of the text. Detector 29 also had low drift HV, although it is evident that it was still high enough to allow for a somewhat successful calibration. The ideal drift HV is around 2000 V. The first three chambers mentioned all have below 800 V drift HV, but detector 29 has a value of 1605 V. The voltage drop is not significant enough to cause major problems although the quality is still affected to some degree as indicated by the slightly wider distribution when compared to other detectors in layer 5. Statistics also appear to be lower and the distribution is not as well defined in shape. Detector 12 also shows a slightly lower HV of 1791 V. This is even closer to the optimal value and it is unclear whether this distribution

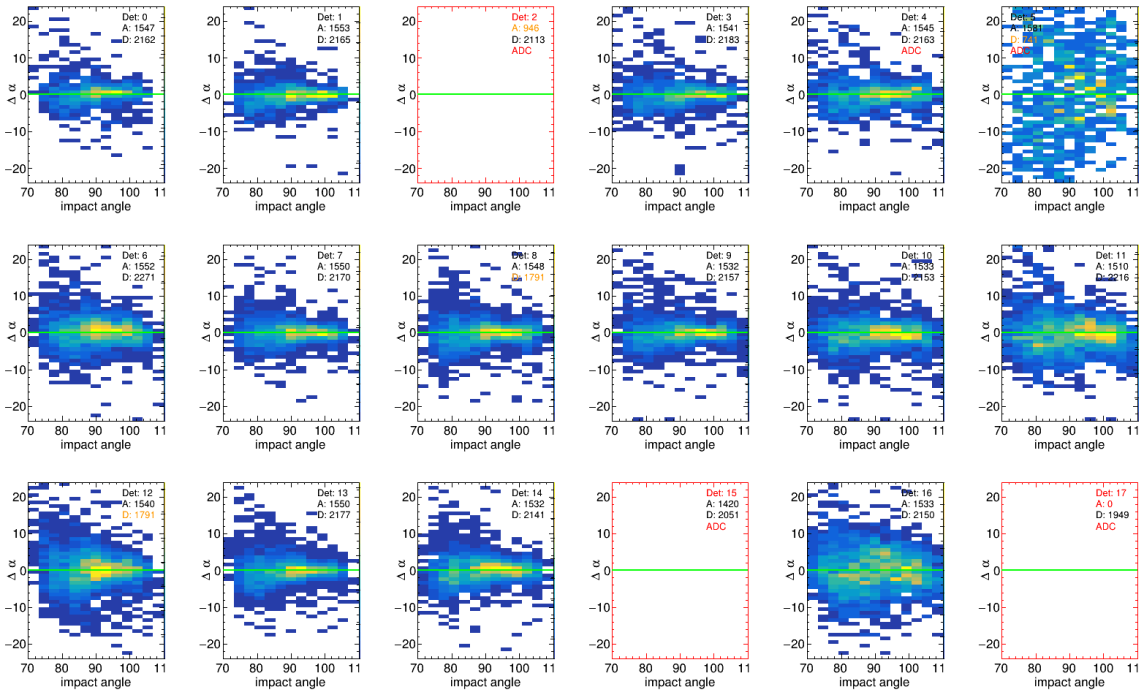

Figure 6.1: 2D histograms of the distribution of $\Delta\alpha$ against impact angle. The quantity $\Delta\alpha$ is given in degrees. This plot displays stacks 0-2 of sector 0. One row represents one stack. Each subplot shows the detector number and the anode and drift voltage for the specific chamber written in either of the top corners depending on where there is the most available white space. The HV numbers are drawn in orange if the voltage is lower than the corresponding QC cut that was made (see section 5.1.3) and are drawn in red if the value is 0 or missing from the record. Additionally, some chambers have a red "ADC" stamp indicating that the pulse height spectrum from this chamber was flagged according to the QC method described in section 5.1.4. A red detector number indicates that the chamber is not calibrated and a red border (axes) indicates that the chamber was flagged by the official QC.

is affected at all by this. The distribution still looks relatively narrow and similar to chambers that had optimal drift HV. One possible explanation for why the performance of the calibration suffers so badly for low drift HV is that the tracklet fits become unreliable under these conditions. If the drift HV is low, electron clusters will drift very slowly toward the amplification region and fewer will arrive within the read out time span. The effect of this will be a shortening of the plateau region of the pulse height spectrum. Essentially, this will result in there being fewer points to which a tracklet fit can be made. Fewer points results in higher uncertainty and, therefore, more deviation in the tracklets will be observed. This will widen the distribution of the $\Delta\alpha$ residuals and the calibration fitting algorithm will struggle to find a good choice of v_D and α_L. This causes a large error in the calibration and the wide distribution of $\Delta\alpha$ persists.

QC concerns relating to pulse height spectra can be seen to have no significant correlation with RMS. This is shown in Figure 6.3. No clear difference is observed between the RMS for layers with no QC issues and layers with QC issues related to the pulse height spectrum. However, low drift HV does result in a significantly higher RMS per layer.

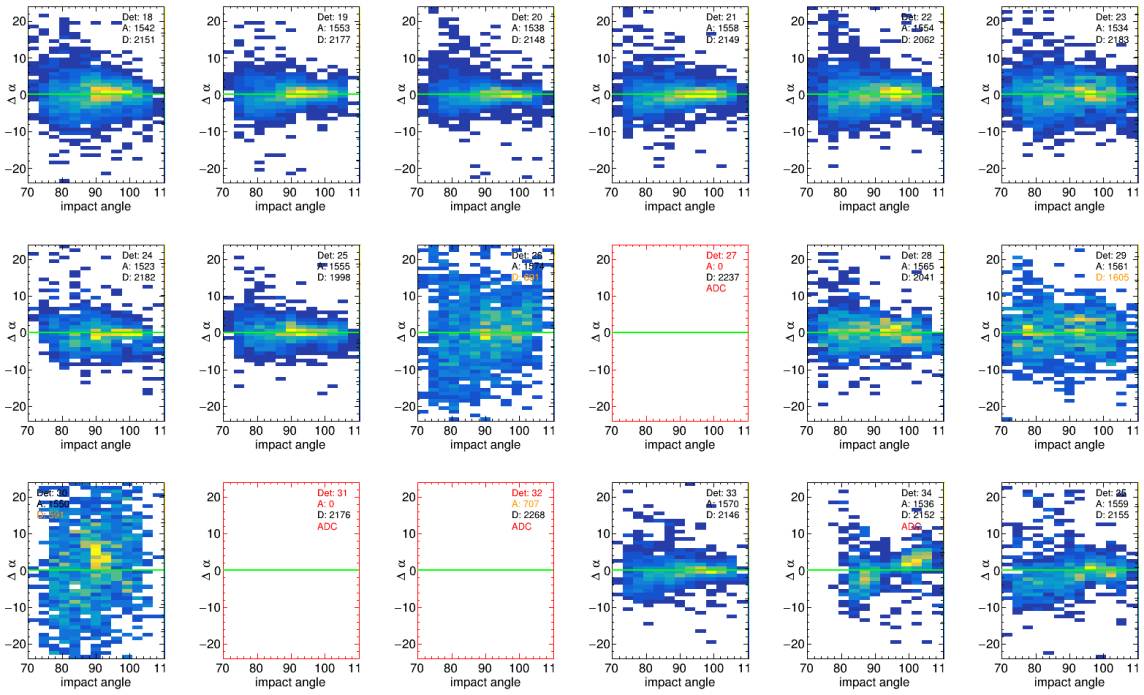

Figure 6.2: 2D histograms of the distribution of $\Delta\alpha$ against impact angle. This displays stacks 3-4 of sector 0 and stack 0 of sector 1 in the same way as Figure 6.1 above.

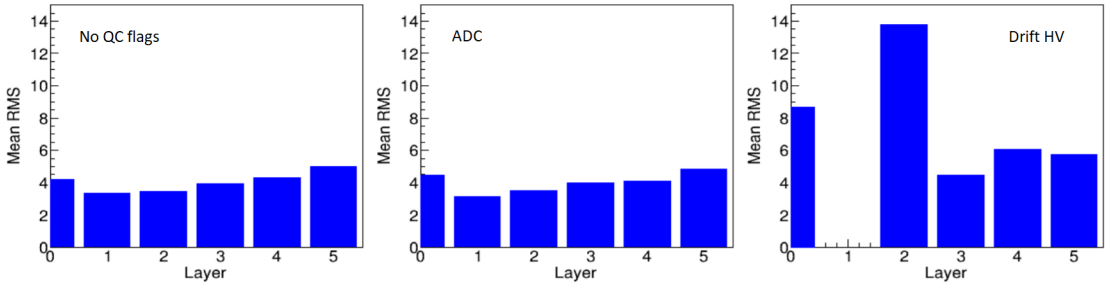

Figure 6.3: The mean RMS as a function of layer number for chambers with no QC flags (left), chambers with QC issues related to the pulse height spectrum (middle), and chambers with low drift HV (right). No chambers in layer 1 had drift HV issues.

A hot spot can be seen on the 2D histograms in figures 6.1 and 6.2 between 90° - 100° where there is a higher concentration of tracklets. This is the result of the Lorentz force acting on electrons ionized by tracklets passing through the detector at an impact angle of around 90°. An angle of 90° is most common due to the geometry of the TRD, but the standard Lorentz angle of around 8° results in this effect being seen between 90° - 100°.

In general, the width of the distribution can be seen to depend on the layer. Layers 1, 2, and 3 appear to have a narrower width compared to layers 0 and 4, and layer 5 seems to have the widest distribution. This effect can be seen in Figure 6.4 below. The general increase in RMS across layer number is likely a consequence of multiple scattering which is predicted to cause a drop in performance of approximately 2° between layer 0 and layer 5 [40].

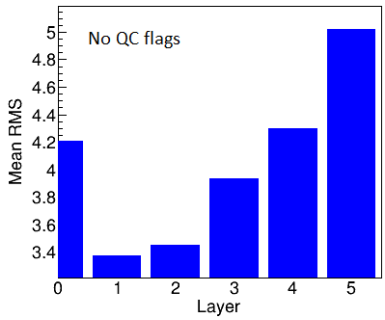

Figure 6.4: The mean RMS as a function of layer number. Here it can be seen more quantitatively how the RMS depends on the layer number.

Another interesting phenomenon is noted. In sectors 4 and 13 which are the two sectors that lie along the x axis in the global coordinate frame, a splitting effect is observed. The 2D distributions have a gap around 90° impact angle and then a dense cluster on either side of this angle. This effect is found almost exclusively and ubiquitously in these two sectors. An example is shown in Figure 6.5. The cause of this effect remains unknown but must have some connection to the azimuthal distribution of TRD tracks.

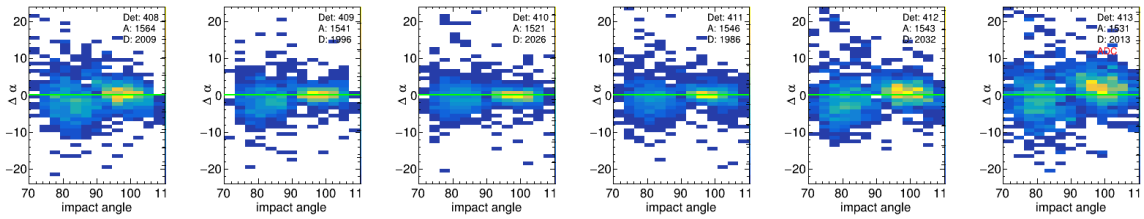

Figure 6.5: The "splitting" effect discovered in sectors 4 and 13.

6.1.2 Residuals

Gaussian fits were applied to each column of bins for plots shown in section 6.1.1 by taking a y projection of the 2D histogram for each bin along the x axis. Graphs of the Gaussian σ were drawn. These are shown in figures 6.7 and 6.8.

The Gaussian fits are done in two stages. The first stage involves a fit over the range of $[\mu - 1.5\text{RMS}, \mu + 1.5\text{RMS}]$. The RMS is obtained directly from the y projection histogram via a ROOT method. For the mean, μ, a value of 0 is used as a first approximation and for the Gaussian amplitude, the initial guess was taken to be 20 tracklets. From this first fit, new values of Gaussian σ, μ, and amplitude will be obtained. However, due to the large amount of variation in the shapes of distributions across different chambers and impact angle bins specifically in terms of statistics, this first fit was not found to be a general enough solution. Therefore, a second fit is done using the fit parameters from the first as initial guesses. This new fit is performed over the range $[\mu - 1.5\sigma, \mu + 1.5\sigma]$ where μ and σ are obtained from the first fit. The second fit was found to be more accurate and reliable. A small sample of what the Gaussian fits looked like is shown in Figure 6.6.

The residuals shown in figures 6.7 and 6.8 more clearly display the width of the distributions of each bin in the previous plots shown in figures 6.1 and 6.2. The red dotted line is the average which was calculated over all chambers that did not have any QC problems including issues flagged by the methods shown in the QC investigation in chapter 5. In some cases, there are not enough statistics to determine a Gaussian fit and a default value of $\sigma = 0$ is

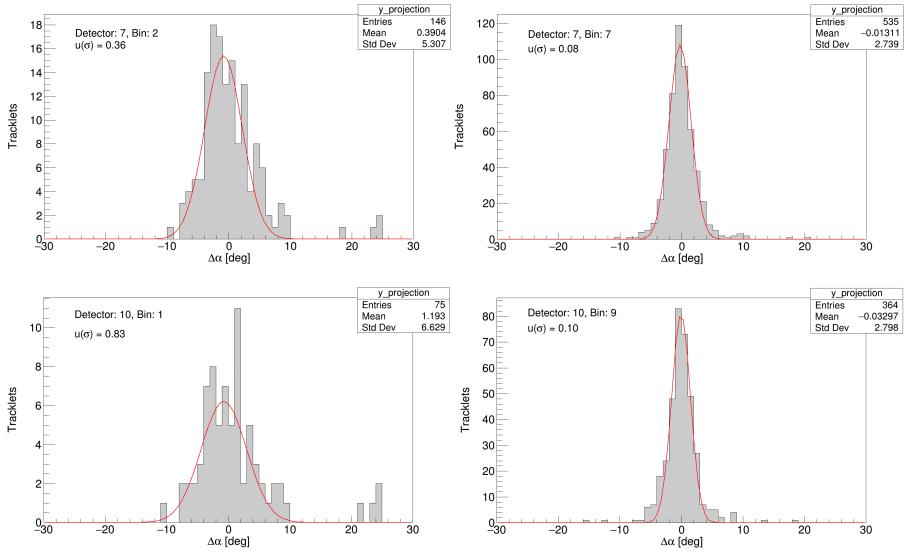

Figure 6.6: A representative sample of y projections obtained from data in Figure 6.1 with Gaussian fits shown atop. The quality of the fit depends heavily on the amount of statistics available. Therefore, fits in bins 0 and 1 could often not be made for example in the lower left plot. The results from this fit will not be factored into the determination of the average deviation based on the low amount of statistics (< 100 tracklets). The quantity $u(\sigma)$ is the error in σ extracted from the fit.

taken. It should be noted that these zeros do not factor into the calculation of the average. Furthermore, instances where the error in the standard deviation of the second Gaussian fit, $u(\sigma)$, is greater than 1.5 or cases where there are less than 100 entries in the y projection are, likewise, not factored into the calculation.

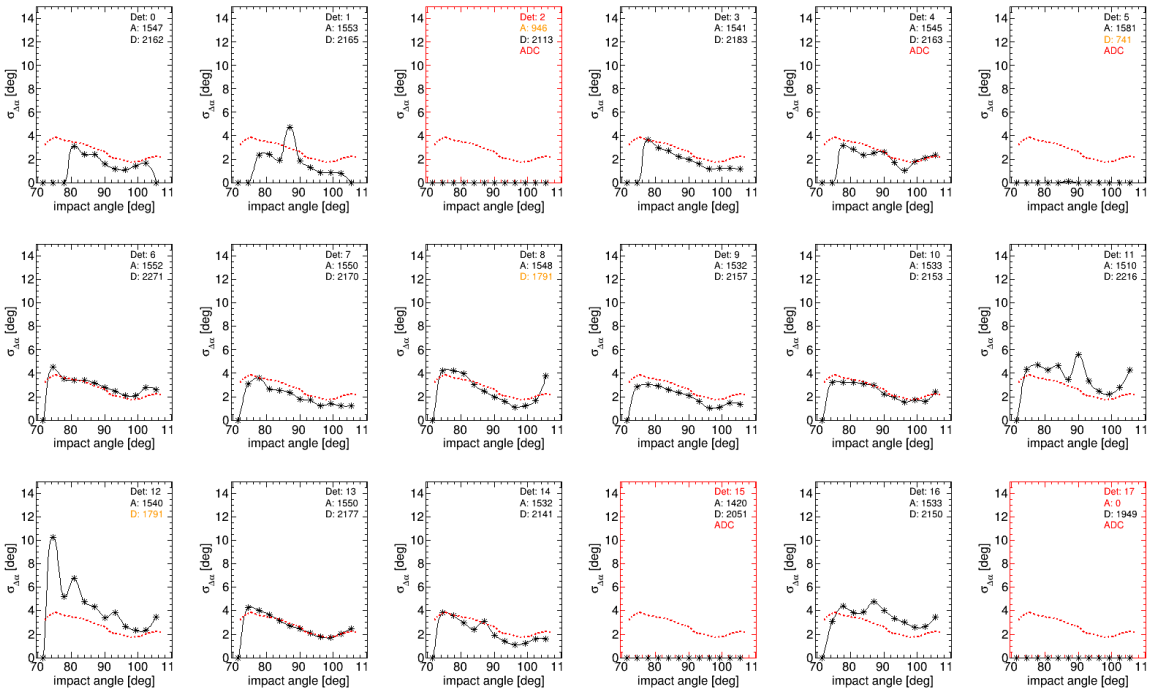

Figure 6.7: Standard deviations of Gaussian fits applied to each bin along the x axis of the 2D plots shown in Figure 6.1. The graph points are placed at the bin centres. The red dotted line shows the average over all good chambers in the TRD.

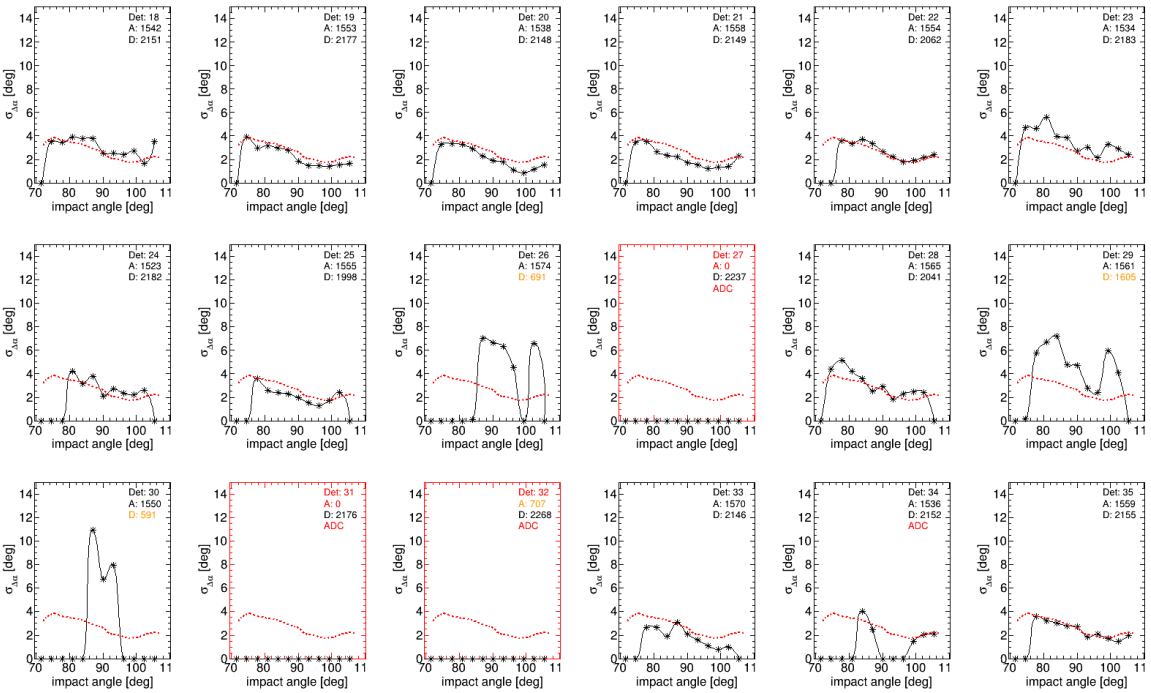

Figure 6.8: Standard deviations of Gaussian fits applied to each bin along the x axis of the 2D plots shown in Figure 6.2.

The effect of a low drift HV can be seen in a more quantitative way in chambers 26, 29, and 30 where the plotted curve is far above the average (red dashed line). Fluctuations can be seen to be common and there is a noticeable loss in resolution on the upper and lower edges

of the plot.

The minimum is more clearly seen around 95-99° which is in line with what would be expected from the Lorentz drift. This will be discussed more in the following section.

6.1.3 Comparison of Results with another Study

Here, a comparison is drawn between the residuals measured above and results obtained from a previous study by Ole Schmidt, Ph.D., and presented in their Ph.D. thesis of 2020 [35]. The study was accompanied by a plot shown in Figure 6.9.

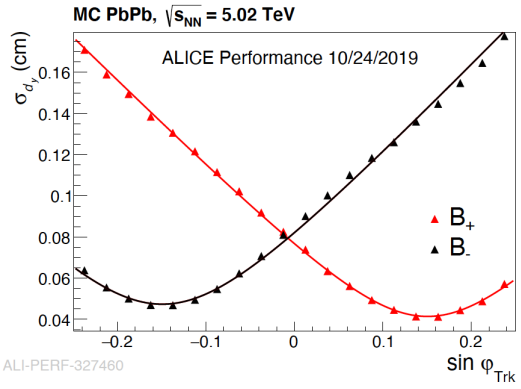

Figure 6.9: As a function of track angle, the deflection length resolution is shown which is proportional to the angular resolution. Results for both magnetic field polarities are provided [35].

The data points from the plot in Figure 6.9 were extracted using `xyscan` [37]. The axes were then transformed from sinϕ to ϕ which is defined as the impact angle on the x axis and from σ_{d_y} to $\Delta\alpha$ on the y axis. The deflection length could be transformed to a tracklet angle based on knowledge of the height of the drift region, h, and using the relationship $d_y = h/\tan\alpha$. For the residuals here, the Gaussian fits were used and the σ standard deviation of the Gaussian was taken to be the error. This is the same method as was used in the previous study. A visualization of the comparison of the previous study with the results gathered in the preceding section is shown below in Figure 6.10.

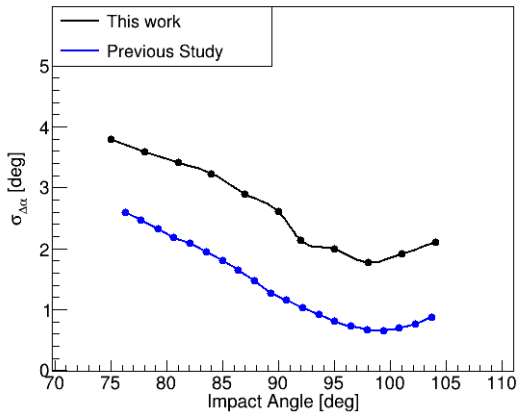

Figure 6.10: A comparison of $\Delta\alpha$ residuals obtained in a previous 2020 study with the results obtained in the preceding section of this work [35]. The impact angle range of coverage of the results from this work are reduced by 2.5° on either side compared to the results shown in the residual plots of figures 6.7 and 6.8. This was done to match better with the range obtained from the previous results.

In Figure 6.10, the average deviation can be seen more clearly. It varies approximately between 2° and 4° with a minimum occurring at 98° with an uncertainty of 3° from the bin width. The mean σ over all impact angles was calculated to be $\bar{\sigma} = 2.6°$. In theory, the minimum should occur at the Lorentz angle which equates to an impact angle of around 98°[1]. Due to the minimum being offset from the centre, the distributions are also observed to be asymmetric for the above impact angle range. In terms of the comparison, a correlation of shape is observed. Both of the curves reach a minimum at approximately 98°, however, there is some offset between them. The black curve (the results obtained in this work) has an offset of approximately 1° to 1.5° compared to the previous study in blue. Possible reasons for this are offered:

- Primarily, this discrepancy could be explained by pad-tilting. Pad-titling causes a deviation of $\approx 2°$ over the full width of the pads. The previous study factored in pad-titling when measuring residuals, while the result presented here does not. In Run 3, this will be taken into account.
- The previous study made use of high p_T tracks which are less prone to fluctuations and multiple scattering which can disturb tracklet reconstruction.
- The previous study considered only tracks with 6 tracklets while tracks with 3 or more tracklets were used to calculate the result presented here. Tracks with more tracklets are usually of higher quality and residuals tend to be lower.

Cumulatively, these reasons were regarded as grounds to conclude that the results of the preliminary calibration were within acceptable error and, therefore, that the calibration algorithm was working correctly. Further comparison is needed once final calibration is done to confirm this.

[1] In this study, no distinction was made between tracklet residuals from positive and negatively charged particles although it has been shown that those from negatively charged particles are smaller. This is due to negatively charged particles tending to have higher impact angles because of the direction of influence of the magnetic field. The Lorentz drift counteracts this effect resulting in a higher concentration of tracklets reconstructed close to 90° for which the angular resolution is optimal [35]

6.2 Offset Resolution

A similar procedure was followed to determine the resolution of the tracklet position based on the offset points. This result is secondary to the angular resolution which was the primary achievement of calibration. The position resolution is a function of the tracking performance as well as the t_0 calibration which had not yet been done at the time of writing.

6.2.1 2D Distributions

2D histograms were drawn displaying Δy as a function of impact angle. The quantity Δy is defined as the distance measured from the tracklet offset point, $\boldsymbol{x_o}$, to the circle fit along the local y direction. Calculating this quantity involves several steps that were integrated with the circle fitting procedure in the primary calibration code. Since the offset points in the calibration code are given in the global coordinate frame, it is first necessary to determine the local y unit vector in the same frame. This can be done purely with knowledge of the sector number and using a similar transformation as shown in eq. 4.12 in section 4.3. The unit vector $\hat{y}$ is given by

$$\hat{\boldsymbol{y}} = \langle \cos\phi, \sin\phi, 0 \rangle \, ; \qquad \phi = -90^\circ + (20^\circ s + 10^\circ) \tag{6.1}$$

Next, the magnitude of the distance between the offset point and the circle radius, Δr, in the radial direction is calculated. This is done by first calculating the magnitude of the distance between the centre of the circle and the offset point and then subtracting this from the radius of the circle. In a similar fashion, the radial unit vector, $\hat{\boldsymbol{r}}$, that points from the centre of the circle to the offset point is also calculated using

$$\boldsymbol{r} = \langle x_{o_1} - a, x_{o_2} - b, 0 \rangle \, ; \qquad \hat{\boldsymbol{r}} = \frac{\boldsymbol{r}}{|\boldsymbol{r}|} \tag{6.2}$$

where a and b is the centre of the circle in x and y respectively. With these pieces of information, it is possible to determine the point of intersection, x_i, between the circle and a line drawn from the offset point along the local y direction. This is given by

$$\boldsymbol{x_i} = \boldsymbol{x_o} - \Delta r \frac{\hat{\boldsymbol{y}}}{\hat{\boldsymbol{y}} \cdot \hat{\boldsymbol{r}}} \tag{6.3}$$

Finally, Δy can be calculated with

$$\Delta y = |\boldsymbol{x_o} - \boldsymbol{x_i}| \tag{6.4}$$

A selection of histograms from sector 5 are shown in Figure 6.11.

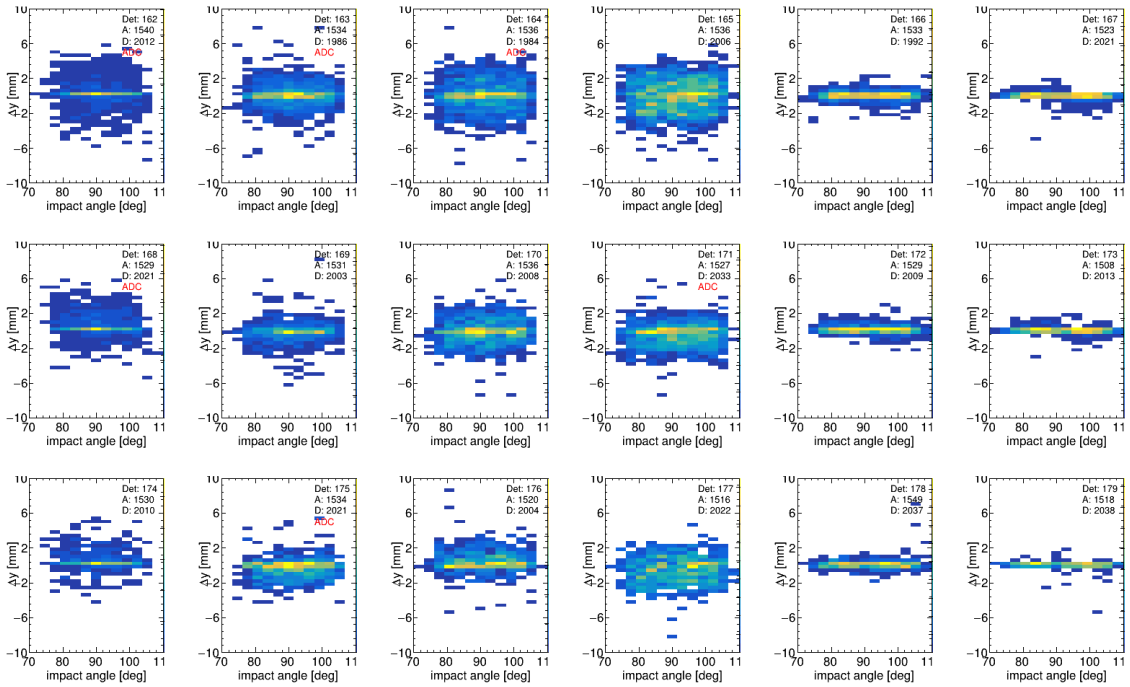

Figure 6.11: 2D histograms of the distribution of Δy against impact angle. This displays stacks 3, 4, and 5 of sector 5.

This particular selection of stacks is shown because the circle fitting algorithm was modified to only accept stacks that had at least 5 good layers and no detectors were found to have any QC issues in this group. Tracks that had been fit to 5 or more points would be more reliable than those based on fewer points. This decision acted toward obtaining the most accurate measurement of Δy. The condition of the track fit quality was also made stricter to ensure only high quality tracks were selected. This was found to make the 2D distributions narrower while still maintaining an adequate degree of statistics for analysis. Lastly, it should also be noted that a bin width of 400 μm was selected for the vertical Δy axis.

In Figure 6.11, a clear layer dependency is observed. The distributions in layers 4 and 5, are very narrow. The distributions of layer 0 have wider tails, but the majority of the entries are concentrated in the first bin, 0-400 μm. This layer dependency was largely determined to be a feature of the track fitting algorithm. It was found to bias towards certain points of the fit, specifically the outer points. Different fitting algorithms were experimented with. The histograms shown in Figure 6.11 represent the best outcome of the track fitting system compared to other algorithms which displayed even more bias. This layer dependency was determined to not be a problem since the average resolution across all detectors could still be calculated and this was the ultimate goal.

6.2.2 Residuals

Two methods of measuring residuals are compared. The first was done on a per detector basis using the RMS of the projections along the x-axis of each of the 2D histograms. The second method made use of Gaussian fits to the projections along the x-axis of a single 2D histogram which represented the sum of all the 2D distributions. These two calculations are thought to roughly represent an upper and a lower bound on the offset resolution.

A selection of graphs of RMS for each detector are shown in Figure 6.12.

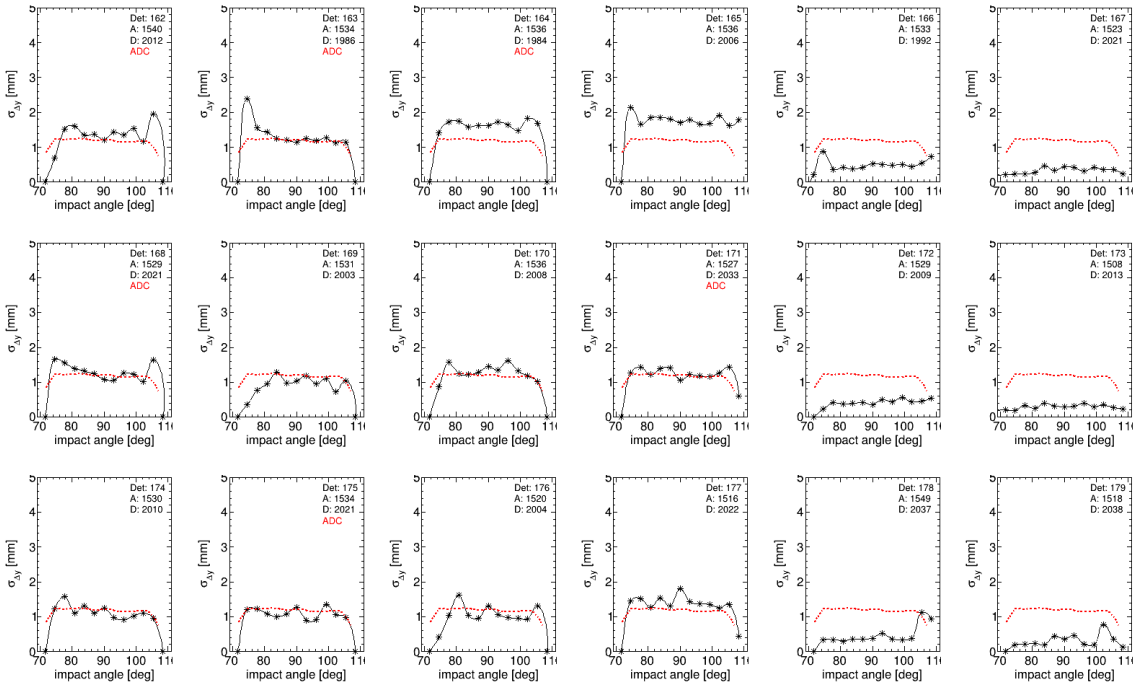

Figure 6.12: RMS of each bin projection along the x axis of the 2D plots shown in Figure 6.11 above. The graph points are placed at the bin centres. The red dotted line shows the average over all good chambers.

The average RMS (shown in red in Figure 6.12) was calculated to be approximately 1.2 mm and remains relatively flat throughout the range of impact angles. Dips are observed at the edges where there are less statistics. Again, QC done on the pulse height spectra was not found to be a relevant indicator of calibration performance, however, similar results were seen as in the previous section when drift HV was too low. RMS was used instead of Gaussian σ since there were, in general, not enough statistics in each bin to perform a successful Gaussian fit. This is largely due to the restrictive 5-tracklet track cut that was imposed. Performance is relatively consistent across layers 0-3, but increases significantly in layers 4 and 5. The circle fitting algorithm was found to bias itself toward certain points which could explain this sudden drop in RMS if it expressed a bias toward these layers. Again, when interpreting results, it should be understood that pad tilting was not taken into consideration in this study. The pad tilting causes a maximum deviation of about ≈ 1.4 mm at the edges of a pad in z and an expected deviation of ≈ 0.7 mm assuming that $\tan(\theta)$ is roughly linear for small angles. With this in mind, the error appears more reasonable. A previous study investigated the position resolution of tracklets with respect to clusters using simulated pp data [33]. That study measured an RMS deviation in the y position of between 0.2 and 0.3 mm depending on the impact angle. The tracklet was a custom straight line fit done to clusters originating from the drift region. Therefore, the fit would have used up to three times as many points as the track fits used to determine the residuals presented in this work. Based on a pulse height spectrum plotted in [33] (see Figure 4.19), the drift region had a length of 14 time bins. Recall the track fits used a minimum of 5 points (layers) and a maximum of 6, although 5 was more common. As an estimate, one may assume an error proportional to $\sqrt{n}$. Then the factor difference in error becomes ≈ 1.7. Cumulatively, these reasons provide grounds for thinking that the error presented here is acceptable.

A different approach was taken to assess the error of the offset involving summing together all the 2D distributions. The result of this sum is shown in Figure 6.13.

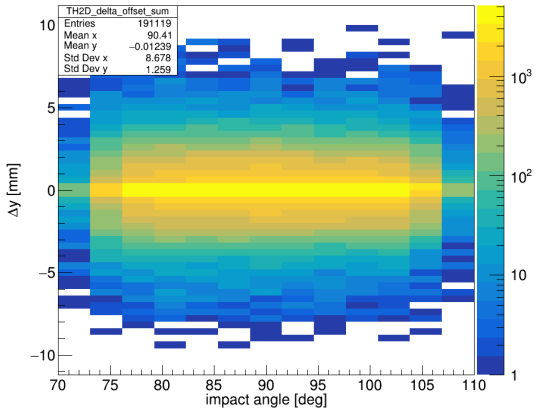

Figure 6.13: The sum across all chambers of the distributions of Δy as a function of impact angle. The z-axis is logarithmic.

Gaussian fits could then be applied to projections of each bin along the x axis of this summed histogram. The projections were roughly found to fall into 2 categories of appearance. These are shown in Figure 6.14. For extreme impact angles (left plot in 6.14), the distribution is more sharply peaked. These are the edge cases. In general, the distribution takes a form similar to the sum of two Gaussians of different widths. The fit is done over the peak.

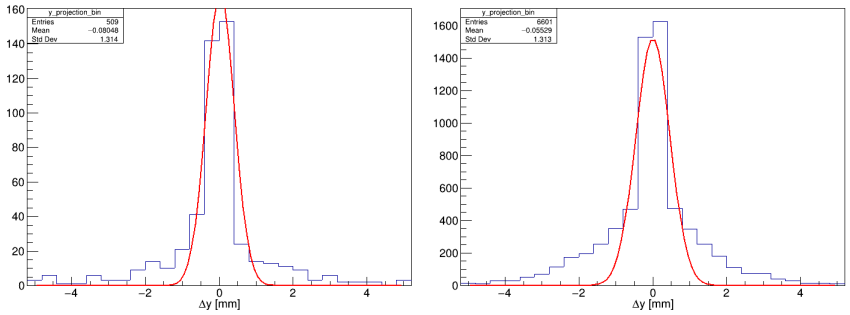

Figure 6.14: Gaussian fits applied to projections along the x-axis of the summed histogram shown in Figure 6.13. The typical appearance of the two outer bins is shown on the left and the typical appearance of the inner bins is shown on the right. In each plot, two distributions are observed. A narrow one and a broader one. The cause is related to the same feature of the track fitting algorithm discussed previously in this section.

The standard deviation of the Gaussian fits were extracted and a tracklet offset resolution was calculated from this method. Results of σ are displayed in table 6.1 below.

Bins	σ [mm]
all	0.473
[1,11]	0.490
[0,12]	0.378

Table 6.1: Table showing σ measurements made across different ranges of impact angle bins shown in Figure 6.13. Range limits are inclusive. This provides an upper limit on the narrow distribution.

In general, the second approach provides another way of thinking about the error, but is not considered to be as reliable as the first approach. As mentioned, the Gaussian fits are done only over the peak of the distributions and therefore, do not take into account a significant number of outliers, especially for the inner bins (right plot in Figure 6.14). The errors shown in table 6.1 are below the expected deviation from the pad tilting effect alone, but only take into consideration the narrow Gaussian. Nonetheless, this error metric may be used when drawing a comparison with other future studies that employ a similar method. This study could be repeated once a better track fitting algorithm is determined.

Chapter 7

O^2 Software

In this chapter, implementations of various pieces of TRD software in the O^2 framework are documented. This includes software related to tracklet calibration (see section 7.2), coordinate transformations (section 7.3), data conversion (section 7.4), and event visualization (section 7.5).

7.1 Introduction to O^2

O^2 has been developed by ALICE in collaboration with the FAIR Software Group at GSI in response to the changing conditions regarding data readout in Run 3. In previous runs, the ALICE detector operated on a trigger system that allowed events to be read out selectively. In Run 3, this will no longer be possible due to the high latency of gas detectors like the TPC, combined with the significantly increased data-taking rate [24]. Therefore, the detector system will operate under continuous readout mode instead. The result of this is a data readout rate of greater than 3 TB/s combined across all detectors [24]. 150 First Level Processor (FLP) nodes will receive the data first and apply initial compression using zero suppression where empty events are discarded. Each FLP receives data only from a part of a specific detector. The smallest unit of data is known as a heartbeat frame (HBF). One HBF is produced per orbit and has a typical length of 89.4 μs [41]. The FLP will group 128 consecutive HBF into one sub timeframe (STF) [41]. The number of HBF to be merged into a single STF is a configurable parameter and may also take the value of 32 during Run 3 [23]. Each STF will be approximately 10 GB in size at this stage [24]. The FLPs then feed the STFs into the Event Processing Nodes (EPN) for merging, reconstruction, initial calibration, quality control, and any other necessary synchronous data processing. Data from different detectors is synchronized using heartbeat signatures and aggregated into time frames up to 23 ms long. At this point, the time frames are further compressed to about 2 GB. The compressed time frames are then stored. All of this happens synchronously with the data taking itself while the beam is active. The length of the timeframe sets an upper limit on the synchronous processing time. In the asynchronous stage, further reconstruction and refinement of the calibration may be done without any strict limitation on processing time. Data from the asynchronous processing stage is then moved to permanent storage where it may be used in the future for analysis. The O^2 framework will facilitate the usage of the same software components across both processing stages (synchronous and asynchronous). Figure 7.1 shows an overview of the flow of data from readout to permanent storage.

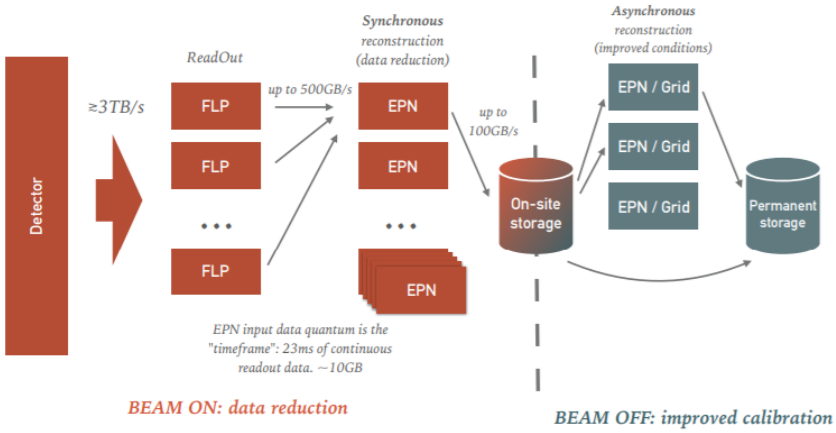

Figure 7.1: O2 data flow [24].

The O^2 framework may be said to consist of three primary parts:

- The Transport Layer, FairMQ. FairMQ defines the core data processing unit called the FairMQDevice. It also provides a structure for communication within a network of FairMQDevices [24].
- The O2 Data Model. A collection of well-defined data objects which are passed among the FairMQDevices for processing. Each object should consist of a header that describes meta information about the data and a payload which is the data itself [24].
- The Data Processing Layer (DPL). The implementation of the generic FairMQDevices written by the ALICE collaboration to perform particular data processing tasks such as reconstruction, quality control, and calibration. Each DPL device must declare its inputs and outputs. The Data Processing Layer will then take care of ensuring that the producers and consumers of each data object are connected.

For the TRD, new DPL devices need to be written to handle the above mentioned processing tasks. At the time of writing, this project is near completion. A workflow is a collection of interconnected DPL devices. They are modular with respect to the devices and allow the user to declare a string of data processing stages. Data will then be passed sequentially through the data processors by the framework's transport layer. It must be ensured that the inputs required by each data processor are accounted for in the workflow by including the necessary devices. For example, there are many "reader" and "writer" devices which read and write data from and to various sources and outputs respectively. DPL devices are most commonly executed within the context of a workflow since it is rare that only a single DPL device is needed to perform a specific task. Workflows can be run via the command line from within an O^2 environment. When one workflow receives inputs from another, they can be linked into a chain using the `|` operator. Workflows that are run synchronously, will be run over each timeframe.

7.2 Tracklet Transformer

As discussed in section 4.2, after a calibration is done and the drift velocity and $E \times B$ parameters are known, tracklets may be transformed geometrically according to those new parameters in order to align them with reference tracks. This step may be thought of as the application of the calibration to the tracklets. The `TrackletTransformer` is an O^2

class accompanied by a workflow which can be used to transform Run 3 tracklets from pad - timebin coordinates into sector coordinates with a given position and direction while applying a calibration stored in the Condition and Calibration Data Base (CCDB)[1]. The CCDB is a database that will be used in Run 3 to store calibration objects and other run parameters. It replaces the OCDB from Runs 1 and 2.

Inputs
As inputs, the `TRDTrackletTransformer` workflow receives two objects. These will either be read from a local ROOT file or received from the DPL control layer depending on the value of the `disable-root-input` option and the context in which the workflow is run. More detail can be read about the options in the section below. Written in the form `<type> <ORIGIN>-<DESCRIPTION>`, the two input objects are:

- `Tracklet64 TRD-TRACKLETS`. Raw TRD tracklets.
- `TriggerRecord TRD-TRKTRGRD`. TRD tracklet trigger record. The trigger record is a vector of length equal to the number of events in the time frame and contains information about the number and ordering of digits and tracklets across each event.

A distinction is made between the `DESCRIPTION` of the data object which is how the objects are identified in the DPL transport layer and the data type implemented as an O^2 class that may have a different name. The `DESCRIPTION` is always in upper case and is declared in the DPL device that produces the object. It should then be subscribed to by the receiving data processor via a standard syntax.

Outputs
The workflow produces two output objects. These are:

- `CalibratedTracklet TRD-CTRACKLETS`. The new data format for calibrated tracklets. These have member variables of transformed Cartesian x, y, z, the calibrated deflection length, dy, and inherit the `Tracklet64` word so that other information including MCM number and HCID can also be obtained.
- `std::vector<char> TRD-TRIGRECMASK`. A vector of bools indicating whether each event was accompanied by ITS data. This aids with tracking.

Options
The workflow accepts 4 different options. These are written in the form `<type> <option_name> (<default_value>)` below:

- `bool disable-root-input (false)`. Specifies whether inputs should be read from local ROOT files.
- `bool disable-root-output (false)`. Similar to the above, this option specifies whether outputs should be written to a local ROOT file.
- `int timestamp (555555)`, this specifies the CCDB timestamp to pull calibration parameters from if the parameters are obtained via the CCDB API. Timestamp 555555 is a special entry that contains all the default calibration values from Run 2. In the future, the CCDB API calls will be replaced by a more standard way of receiving CCDB objects via the framework. At that point, this option will become obsolete.

[1] Any transformation from pad - timebin coordinates into spatial coordinates implicitly involves a calibration since a value for v_D must be used.

- `bool filter-trigrec (false)`. If true, the workflow will ignore all interaction records that are not accompanied by ITS data.

Operation

The `TrackletTransformer` begins by transforming the coordinates of the input tracklets in pad - timebin space to Cartesian space within the local chamber coordinate system. The point that is used to represent the position of the `CalibratedTracklet` is, ideally, the point of minimal uncertainty. That is to say, the point with which it is most certain that the primary tracklet intersected. This is similar, in principle, to the "offset point" described in chapter 4, but implemented differently here.

The coordinate x is reported 5 mm below the height of the cathode plane at $x = 2.5$ cm. This is done to achieve a compromise between the two different sources of error involved in the calculation of the tracklet and provide a more probable point of intersection with the primary tracklet. The first source of error is the tracklet fit itself. Any straight line fit has an error that is maximal at the ends and minimal at the centre of mass which is determined by the distribution of the data points and their relative uncertainties. Since the tracklet fit is ideally performed exclusively over the clusters from the drift region, the centre of mass would be at the centre of the drift region at $x = 1.5$ cm. This assumes that the errors associated with the positions of each cluster are uniform. The second source of error is from the calibration fit and the determination of the v_D and α_L parameters. These parameters have the effect of rotating the tracklet about the cathode plane since their region of effect is also confined to the drift region and the magnitude of the effect is proportional to the time over which it acts. Therefore, the plane of minimal uncertainty from this source is the cathode plane at $x = 3$ cm. Assuming that the contribution of uncertainty from the second source is higher than the first, it becomes a reasonable proposition to report x on a plane slightly closer to the plane of minimal uncertainty from the second source than the first in order to minimize the overall uncertainty. Therefore, $x = 2.5$ cm was used. Further research is suggested to determine the exact plane of minimal uncertainty.

The coordinate y is a function of tracklet HCID, pad column, and `position`. The pad width is known from sources such as [14] and recorded in the O^2 framework amongst a collection of constants. Pad tilting is not taken into consideration at this point, since information concerning the z position is only known up to the resolution of 1 pad row. As pad tilting is a per-pad row effect, and it is not known where in the pad the tracklet is, it cannot be taken into account with any degree of certainty. Pad tilting is discussed in more detail in section 7.3 below. Column refers to the MCM column number on the readout board. It takes values from 0 to 3. With knowledge of the HCID and the MCM column, it is possible to determine the precise MCM column that the tracklet is in. The tracklet `position` is a 10 bit signed integer that describes the position of the tracklet within the MCM along y. It is 0 at the centre of pad 10. Keeping in mind that each MCM is connected to 21 ADC channels, this places the origin at the centre of the MCM's range of coverage. The `position` is converted into an unsigned integer such that the decimal value 1024 corresponds to the centre of the MCM. Then, the position of the `CalibratedTracklet` in pad units can be calculated with

$$\text{pad} = \left(\text{pos}_{\text{unsigned}} - 1024\right)\,[\text{pos granularity}] + N_{\text{col}}\,(N_{\text{MCM}}[\text{HCID mod 2}] \times [\text{column}]) + 10 - 1 \tag{7.1}$$

where $N_{\text{col}} = 18$ refers to the number of pad columns in an MCM and $N_{\text{MCM}} = 4$ refers to the number of MCMs in column direction in one ROB. The position granularity is a TRD

constant in the O^2 framework that represents the conversion scale from `position` to pad units. It has a value of 1/40. The final y position in cm is then given by

$$y = [\text{pad width}]\,(\text{pad} - 72) \tag{7.2}$$

The z coordinate of the `CalibratedTracklet` is given at the centre of the pad row since this is the expected value of the quantity and no further information is available to improve the precision.

A calibration is applied to the deflection length dy and the x position. The raw deflection length in cm is calculated as

$$dy_{\text{raw}} = t[\text{slope}_{\text{signed}}][\text{pad width}][\text{slope granularity}] \tag{7.3}$$

Where slope is an 8 bit integer taken from the `Tracklet64` word and is given in units of $\frac{\text{pads}}{\text{timebin}}$. The pad width is uniform for a given chamber, but is a function of layer number. Therefore, the detector number must be known. The slope granularity is a constant stored in the TRAP and is set to 1/1000. The quantity t_D is the time in units of time bins that it would take for an electron to drift across the entire drift chamber for this specific detector. This is, therefore, a function of v_D which is received from the CCDB. It can be shown that

$$t_D = \frac{h[\text{sampling rate}]}{v_D} \tag{7.4}$$

where v_D is in units of cm/μs and h is the height of the drift region which is 3 cm. The sampling rate is the number of time bins sampled per μs. The sampling rate is taken as 10 timebins per μs.

The raw dy is then corrected using the Lorentz angle α_L. Over the full length of of the drift region, the effect has a magnitude of

$$\Delta y = h \tan(\alpha_L) \tag{7.5}$$

The deflection length is measured in the positive y direction so that a tracklet sloped toward the positive end of the chamber in y results in a positive dy while a tracklet sloped away from the positive end of the chamber in y results in a negative dy. A positive α_L is taken to indicate that the tracklet was reconstructed with a larger slope than what is accurate. This means that the reconstructed tracklet will have a larger dy and therefore, a positive α_L results in a reduction of the raw dy during calibration. The final deflection is then the sum

$$dy = dy_{\text{raw}} + \Delta y \tag{7.6}$$

In addition to using the `TrackletTransformer` in a workflow, the transformer function is public and can be accessed by any ROOT script outside of a workflow. This function will take a `Tracklet64` as an argument and return a `CalibratedTracklet`.

7.3 Pad Plane

The `PadPlane` class contains detailed information regarding the geometry of the readout pads as well as a number of methods that accommodate transformations between pad coordinates and other reference frames. New public coordinate transformation functions were added to the `PadPlane` class. These functions are `double PadPlane::getPadRow(double z)` and `double PadPlane::getPad(double y, double z)`. Both take in points in local chamber

spatial coordinates and return float values in row and column coordinates. This allows for transformations between these two coordinate systems without any loss of information as there is when returning integer values for which functions already existed.

In order to calculate the exact pad position of a point in the pad plane given in spatial coordinates, information about the position in pad row direction, z, is also required so that pad tilting can be taken into account. It was important that the pad tilting geometry was correctly understood so that the effect could be encoded and tested properly. The authoritative 2010 thesis of David Emschermann, Ph.D. was consulted in this regard and a chamber in layer 4 was considered in particular [15]. In Figure 3.10 of the thesis cited, it is written that the "pads in layer 4 appear to be tilted in the clockwise direction from the perspective of a particle originating from the [interaction] vertex". From this information, and with knowledge of the ALICE global coordinate system, it is possible to draw a diagram of the tilted pads to gain a better understanding. This is shown in Figure 7.2.

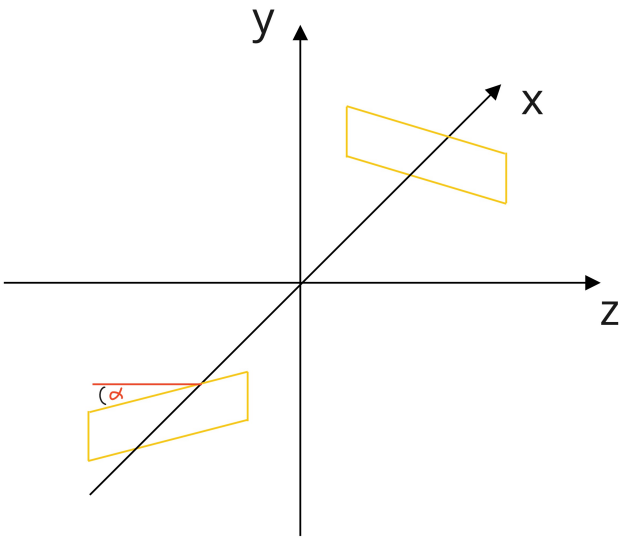

Figure 7.2: Two tilted layer 4 pads shown on opposite sides of the interaction vertex in hypothetical sectors -0.5 and 7.5 with sector -0.5 being at larger x. These sectors are chosen for simplicity. A red line is drawn parallel to the z axis with the tilt angle indicated by α (bottom left). The tilt of 2° is visually exaggerated for clarity. It should be understood that the side of the pad facing the reader is the side touching the adjacent pad. ALICE global coordinate axes are shown.

Based on this understanding, it is possible to transform the axes and draw a set of pads for a chamber in sector -0.5 with correctly labelled axes given in tracking coordinates. This is shown in Figure 7.3. From this diagram, a qualitative understanding can be gained concerning the offsets that are to be expected at small distances away from the origin in z. It can also be seen how moving a point from one end of the pad row to another along z can change the pad number depending on the value of y. Since the pads are tilted about their centres, a tilt of 2° results in a maximum offset in y of anywhere between 0.13 and 0.15 cm depending on the pad length. The pad length is itself a function of layer, stack, and whether or not the pad row is at the edge of the chamber.

The two new `PadPlane` methods were tested by varying the arguments along y and z and plotting the outputs. The results are shown in Figure 7.4. From these plots, it can be seen

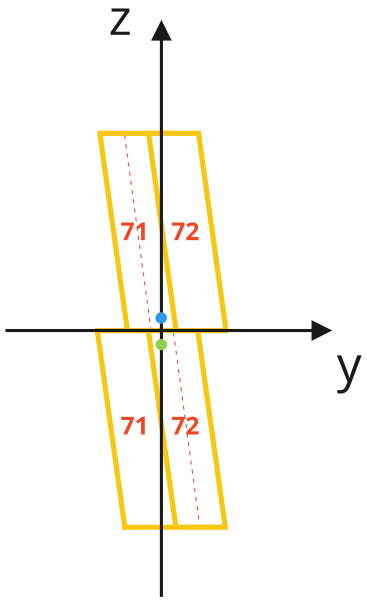

Figure 7.3: Pads 71 and 72 in two adjacent pad rows. Axes and tilt direction are derived from 7.2. The tilting angle is again exaggerated for visibility. From the origin, blue and green dots are shown at $z = \pm\epsilon$ respectively. The red dotted lines show the centres of pads 71 and 72 in the top and bottom pad rows respectively. From the plot, it can be clearly seen that at $z = \epsilon$, the blue dot is above the centre of pad 71, while at $z = -\epsilon$, the green dot is below the centre of pad 72 in the respective pad row. Axes are in tracking coordinates.

that pad position increases linearly with increasing y position while z position is fixed (top left). In this plot, the y coordinate is varied across the full range of the chamber and the slope of the resulting line appears to remain constant. In theory, the line should have slightly steeper slopes in the two outer pads since these are shorter along y. Similarly, the pad row position also increases linearly, but with a slope that depends on the pad row number (top right). In the outer pad rows, the pad row position increases more rapidly with increasing z because the pad rows are shorter in length. The most complex behaviour is observed in the third plot (bottom Figure 7.4). Here, the pad position is considered as a function of z. Firstly, discontinuities can be noted on the boundaries between each pad row. This is because the top of a pad in one pad row will be offset in the opposite direction as the bottom of a pad in the adjacent row since all pads are tilted in the same direction about their centres. This can be seen clearly in Figure 7.3. Then, as z increases, pad position increases since the tops of the pads are tilted away from the positive side of the y axis. The shorter outer pads can be seen at the top and bottom of the plot with smaller maximum offsets. Manual calculations of pad column and pad row positions corresponding to various sets of Cartesian coordinates were done and a CMake unit test was written in O^2 to ensure that the new methods of the `PadPlane` class aligned with the theoretical understanding. All tests were passed successfully.

A significant modification was also made to `double PadPlane::getTiltOffset(int row, double rowOffset)`. This was investigated prior to knowing that the pads were tilted about their centres. This method calculates the offset at a particular point in pad direction arising from the pad tilting effect. Previously, it did not take into account the shorter length of the outer pads. However, this information is required since determining the tilt offset requires knowledge of the distance in pad row direction from the point about which the pad is tilted.

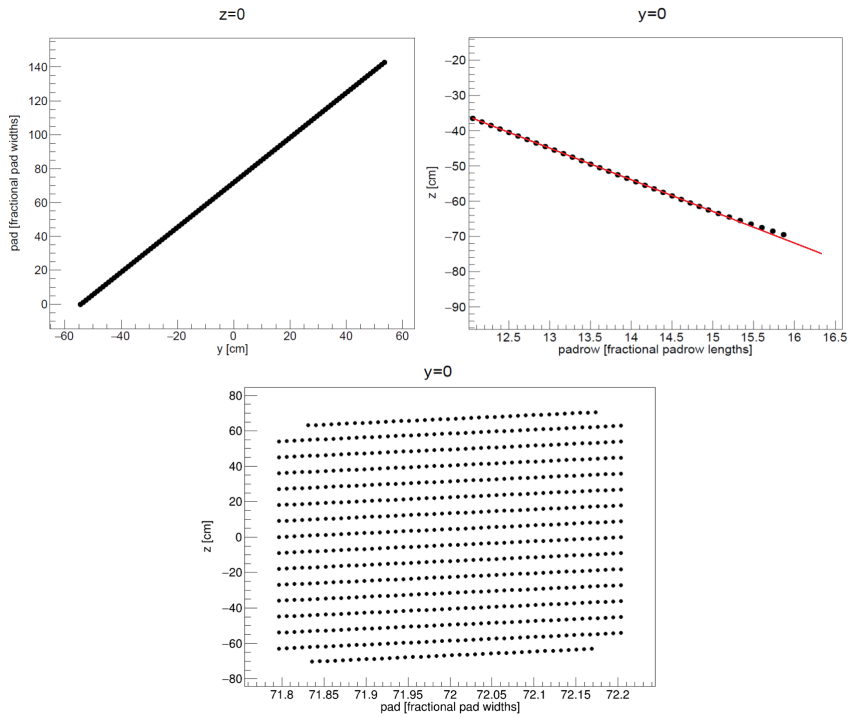

Figure 7.4: Output of the new `PadPlane` methods while varying the input coordinates along y (top left) and z (top right, bottom). In the top left plot, the z coordinate is fixed at $z = 0$ while for the top right and bottom plots, $y = 0$ is fixed. A straight red line is drawn in the top right plot as a visual guide representing the slope of the points from pad rows 12 - 15. From pad row 15, the slope increases since the two outer pad rows are shorter along z.

Then, the offset can be calculated using $d\tan(\alpha) = \Delta y$ where Δy is the offset and d is the distance from the point about which the pad is tilted. Through careful analysis of schematics of the TRD, it was ultimately determined that all pads are tilted about their centres.

The primary schematic investigated was Figure B.11 of [15]. This was the technical schematic used as reference to produce the pad planes. The cited PDF document was enlarged and a screenshot was taken of three pad rows including an outer pad row. PDF documents render vector graphics which allow for up-scaling of the resolution of images as they are enlarged. This ensured no loss of precision during analysis of the schematic. Once the schematic had been enlarged enough, it was rasterized in Paint. The pixel positions of the upper edge of a particular pad number in y were measured at both the top and bottom borders of each of the three pad rows. Results showed that the pad in the outer row had a tilt offset of 5 pixels less on both ends than the corresponding pads in the two inner rows. The pixel positions of the top and bottom edges of the pads in the inner two rows matched perfectly. This provides evidence towards the pads being tilted about their centres since the difference in the offset was the same on both ends. For further justification, the length of the inner pad row in z was measured to be 1846 pixels. For a C1 chamber in layer 4, the length of an inner pad row is 9 cm. This equates to $205.\bar{1}$ pixels per cm meaning that 5 pixels would be 0.024 cm. Assuming an error of 1 pixel in the measurement of the offset difference gives an uncertainty of 0.024 ± 0.005 cm. The length of an outer pad row is 7.5 cm. Therefore, the difference in maximal tilt offset between an inner and an outer pad row of a C1 layer 4 chamber is $4.5\tan(\alpha) - 3.75\tan(\alpha) = 0.026$ cm. This is well within the uncertainty of the

previous measurement and adds to the credibility of the analysis.

The initial implementation of the function assumed that all the pads are tilted about a point that is a distance equal to 1/2 the length of an inner pad away from the border of the pad row. If this were the case, then the above measurements would not be observed. Instead, the pixel measurements in y should have seen agreement across pad rows for one edge of the pad since the offset should be the same in one direction. In the other direction, a larger discrepancy should have been observed since the tilt offset would be calculated over a distance less than 1/2 the length of an outer pad row due the shorter length of the inner pad rows. This was not the case. It is also noted that there is no difference in the width of the inner and outer pad rows nor is there a difference in the widths of the individual pads in these rows. Cumulatively, this was enough evidence to conclude that the outer pad rows are, indeed, tilted about their centres and therefore a correction to the `PadPlane` class was made.

7.4 Run 2 Data Converter

A ROOT macro was written to facilitate the conversion of Run 2 digits or Run 2 raw data into Run 3 digits. This was created primarily in response to a need to test new O^2 software on the Run 3 data format before experimental data was available. This presents an alternative to using only simulated Monte Carlo data. The converter macro was merged into the O^2 code base as `convertRun2ToRun3Digits.C`. The original code was primarily written by Sean Murray and Thomas Dietel. The author reorganized the code and wrapped it into a root macro while making some minor modifications and optimizations where necessary.

The Run 2 digit format differs in some ways from the Run 3 format. Most notably, according to the old format, digits are defined per time bin whereas in the new format, a single digit stores information about the full time bin range of readout. The coordinates of the digit are also represented differently. The pad row coordinate was given explicitly in the old format, but in the new format, it is determined from a combination of MCM and ROB numbers. Furthermore, instead of giving the pad column explicitly, the new digit format stores information about the channel number relative to the MCM. The macro essentially reworks the old Run 2 digit format into the new format and generates the associated trigger record as well.

Paths to the input data should be given in the function definition. The path can either be set as the argument `rawDataInPath` or `run2DigitsInPath`. Depending on which of those two paths are defined, the macro will automatically convert the appropriate file. The file to be written can be given as `outputPath`. The default is `trddigits.root` for consistency with the TRD digit writer workflow. Both the `rawDataInPath` and `run2DigitsInPath` arguments cannot be set simultaneously or the output data will be overwritten by the converted Run 2 digits data since this runs after the raw data conversion and there is no distinction in output paths.

7.5 TRD Event Display

In this section, work done toward a new TRD event display system is described. The work primarily involved migrating the event display, originally written in `AliRoot`, to the new O^2 framework as well as a significant contribution to the data processing backend. Additions to the backend include a new rudimentary tracking algorithm designed to focus the event display on high-quality events when no TPC or ITS tracking information is available.

7.5.1 O2 Workflow

The TRD Event event display was initially created in 2019 to provide a complete visualization of TRD events including associated tracks, tracklets, and digits [42]. The event display consisted of two parts. Firstly, a JavaScript frontend that reads and displays data from a number of JavaScript and JSON files. Secondly, an `AliRoot` analysis task that was written by the original creator to transform ROOT data into the appropriate structure for use by the event display [43]. This analysis task needed to be ported to the new O^2 software framework and packaged as a workflow which could be run easily either offline or online. The workflow was named `TRDEventDisplayFeed`.

Inputs
On a basic level, the workflow accepts a variable number of between 3 and 5 input objects and transforms the data into the appropriate format similar to the previous analysis task. Written in the same form as in section 7.2 of `<type> <ORIGIN>-<DESCRIPTION>`, the five input objects are:

- `Digit TRD-DIGITS`.
- `Tracklet64 TRD-TRACKLETS`.
- `TriggerRecord TRD-TRKTRGRD`. TRD tracklet trigger record.
- `TrackTRD TRD-MATCH_ITSTPC`. ITS-TPC[2] tracks matched to TRD tracklets.
- `TrackTriggerRecord TRD-TRGREC_ITSTPC`. Trigger record for ITS-TPC tracks matched to TRD tracklets.

The last two input objects above related to tracks are optional with the `noTracks` flag discussed later in the options section. When running offline, the above data will be read from three ROOT files which are:

- `trddigits.root` - Contains TRD digits and is read by `TRDDigitReader`.
- `trdtracklets.root` - Contains uncalibrated TRD tracklets and is read by `TRDTrackletReader`.
- `trdmatches_itstpc.root` - Contains ITS-TPC tracks matched to TRD tracklets and is read by `TRDTrackReader`.

When running online, however, the data can be received directly from the workflows from which they are transmitted and no ROOT files need to be read.

Outputs
Run-level parameters as well as track and tracklet information for all processed events are written to a single `.js` file while digit information is stored in one JSON file per track.

- `events.js` - Stores run-level parameters as well as arrays of all track and tracklet data.
- `E<even_number>.<sector_number>.<stack_number>.json` - One file of this type is created per stack with at least one track in it and stores digit information for tracklets associated with the track(s).
- `script.js` - Two short functions to pass the event data to the frontend.

[2] ITS-TPC tracks are tracks which are matched from both the ITS and the TPC detectors.

Options
The workflow accepts 4 options:

- `int nEventsMax (5)`. This specifies an upper bound on the number of events to prepare to be converted. The actual number of events will depend on the number of events received by the workflow as well as the values of the other options below involving event selection based on the presence of digits and/or tracks. The workflow will process as many events as possible in an attempt to reach this limit.
- `bool noTracks (false)`. This specifies whether the workflow should expect to receive tracks or not. If not, then the event display will perform its own custom tracking.
- `bool withTracks (true)`. This option specifies whether the event display should only display events for which tracks were received (if noTracks is set to false) or fitted with the custom tracking algorithm (if noTracks is set to true).
- `bool withDigits (true)`. This specifies whether the event display should only display events for which digits were received. This is useful since most runs are done where digits are only written for a small fraction of events. This fraction is given by the inverse of the `calrate` run parameter [44].

Operation
For each timeframe of objects received, the workflow will loop over all events, but only write a maximum of `nEventsMax` to be displayed. The events are accumulated across time frames. The JSON library `nlohmann:json` was used to assist with reading and writing the JSON files.

Tracks are processed first. These will either be read from input or fitted using the custom tracking algorithm depending on the value of `noTracks`. In the case of input tracks, sector and stack number are determined by the detector number of the first matched tracklet that is found where the search is done in order of layer number from 0 to 5. Track parameters are extracted directly from the `TrackTRD` class and track λ is given in degrees. The quantity λ refers to the track inclination in the $x-z$ plane. The track `path` key contains a set of points on the trajectory of the track which are used to draw the track in the event display. These points are calculated using the `TrackTRD::getXYZGloAt()` method which accepts arguments of magnetic field and x position. Using this information as well as other track parameters in the class, the method calculates a global (x, y, z) point based on the x position given. The calculation is done such that iterating through values of $x \geq 0$ gives the track trajectory regardless of whether it propagates in positive or negative x direction. At the time of writing, the loop has bounds of $0 \leq x \leq 470$ cm in the Event Display code. This range can be extended to higher values which will result in longer tracks drawn in the event display with an upper limit determined by the dimensions of the view port.

If `noTracks` is set to true, the workflow will apply its own tracking algorithm in an attempt to find tracks. This algorithm takes a very basic approach to finding tracks that is simply directed toward the goal of visualizing interesting events. These tracks are not meant to be used for any kind of physics analysis. The algorithm begins by reorganizing input tracklets by sector, stack, and layer to make them easier to search through. All tracklets in the first three layers (moving from the inside of the TRD barrel towards the outside) are analyzed to determine if they could potentially be the starting point of a track. If a particular layer has more than 10 tracklets in it, it is skipped as it becomes difficult to separate signal from noise in these high multiplicity cases. To determine whether a particular tracklet in one of the

first three layers could be the starting point of a track, the algorithm first selects a starting tracklet in one of these layers and then moves sequentially outwards through the proceeding layers to look for candidate tracklets that may be matched with. For each of tracklets in each consecutive layer, the difference in the local y with respect to the closest previous tracklet is measured. To calculate the local y position, eq. 7.2 is used via its implementation in the `Tracklet64` class. The acceptance window is 8 cm multiplied by a factor equal to the number of layers between this tracklet and the previous. This is to take into account a larger expected deviation over a further distance. The angle of the tracklet is not taken into account resulting in poor tracking efficiency at low p_T. The candidate tracklet that is found to be the closest in local y position to the previous tracklet and within the specified limit is selected and added to the candidate track. Once the algorithm has progressed through all layers, the candidate track, if one was found, will be added to the JSON track array if it has a length of at least 4 tracklets. In the case of custom tracks, the track parameters are not determined and a path is not drawn in the event display. However, the IDs of the matched tracklets are recorded in the JSON output and therefore the track will be visible in the event display by the red colouring of the matched tracklets.

Tracklets are processed next. Incoming tracklets are calibrated using the `TrackletTransformer::transformTracklet()` method described in section 7.2. Relevant tracklet information is extracted from both raw and calibrated tracklets. In the JSON output, `localY` refers to the y position of the tracklet in the local chamber coordinate system. This is given at the intersection of the tracklet with time bin 0. Both the "raw reconstructed" and "Lorentz corrected" (see Figure 7.5) tracklets in the event display are drawn from this point. The JSON keys `dyDxAN` and `dyDx` store the slopes of the raw and calibrated tracklets respectively. The slopes are calculated using dy/h where h is the height of the drift region since it is over this distance that the deflection length, dy, is calculated from the raw tracklet slope.

Next, the digits are written. For each stack, the workflow checks whether a track was found in it. If at least one was, digit data is written for each layer in the stack. All 30 time bins are written to the output file. One file is created per stack.

Finally, the workflow applies some de-noising to the output data. This is done by first counting the number of tracklets in each pad across all events. If the ratio of tracklets to events is above a threshold, the tracklet will be assumed to be noise and deleted from the data. The threshold is set by the `float noiseLimitRatio` variable in the code and, at the time of writing, is set to 0.85. Once the noisy tracklets have been removed, the workflow will re-process the saved tracks and any matched tracklets that were found to be noise will be removed from the tracks. If the length of the track remains greater than or equal to 4, the track will be kept, otherwise it will be removed from the output as well. The output path for the digits as well as the `events.json` file is hard coded into the `TRDEventDisplayFeedSpec` class.

7.5.2 Frontend

The TRD event display hosted on the alicetrd website is shown in Figure 7.5. The page is served from a directory named `eventdisplay` in the top level of the website folder on EOS. The corresponding repository is hosted in the author's name space at [45]. Some minor modifications related to the drawing of tracklets were also made to the frontend JavaScript code [46].

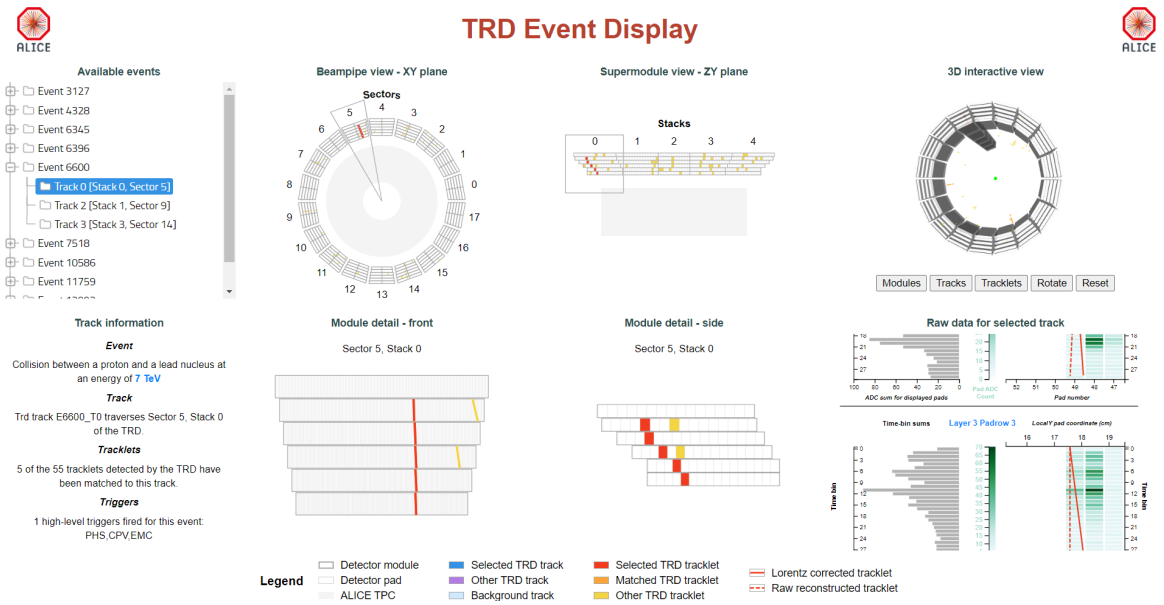

Figure 7.5: The TRD event display showing pilot beam data from run 505673. Several view ports are involved offering different views of the data at different scales. A menu is included on the left-hand side of the screen providing a selection of events and tracks.

In the digits view of the event display, tracklets are drawn over the entire 30 time bins from time bin 0 to time bin 29. As mentioned in section 7.5.1, the starting point of both the raw and calibrated tracklets is drawn at coordinates of $x_1 = (y, 0)$ with a time bin on the vertical axis. The end points of the tracklets are determined by their respective slopes. This is calculated as

$$x_2 = \left(y + s\left(\frac{dy_{\text{raw}}}{dx}dx\right), 29\right) \tag{7.7}$$

where dy/dx is the slope and dx is the height of the drift region. The factor $s = 30/t_{\text{drift}}$ is a scaling factor equal to the ratio of the full 30 time bins over which the tracklet is being drawn to the number of time bins in the drift region over which dy was calculated. This denominator depends on the drift velocity.

Chapter 8

Pilot Beam

In the time building up to the end of LS2 during late 2021, ALICE started to take data. This was first done using cosmics, followed by the first beams in the LHC since the end of Run 2 which began in November 2021 during a short testing phase known as the pilot beam. These runs afforded the opportunity to test a variety of components including the reconstruction workflow. In this chapter, first, a brief section covers hardware at P2 (section 8.1) followed by some comments regarding analysis of cosmic and pilot beam data in sections 8.2 and 8.3 respectively. Lastly, the status of an investigation into read out noise is reported in section 8.4.

8.1 Hardware / DCS

Hardware upgrades and maintenance were conducted at P2. The temporary AnaGate controllers, which were replacing the old PEAK controllers as a means to control the high-voltage power supplies, were disconnected and replaced by the permanent ones [44]. One of the high-voltage crates was also removed and replaced by one with a UPS battery. A few of the modules were replaced too in two of the crates.

In addition to the high-voltage, some of the Linux nodes in CR4 were moved into their new racks.

8.2 Analysis of Cosmic Data

Before the pilot beam in November, the TRD was tested by recording cosmic events. Cosmic rays are the result of interactions between high-energy protons with the Earth's upper atmosphere. Sources of the protons include solar events like supernovae as well as quasars and gamma-ray bursts from distant galaxies. These interactions bring about the production of pions which decay into high energy muons. Muons are weakly-interacting and will often penetrate the atmosphere and reach the surface of the Earth. At ground-level, approximately 100 muons pass through every square metre per second. Where ALICE resides, at a depth of approximately 60 m, this number is a lot lower. Nonetheless, those with significant-enough energy, have a probability to penetrate to that depth and, since they are also charged, they can be detected by the TRD.

On the 24th September 2021, cosmic data taking run number 502238 was performed using configuration 42201 with the Time Of Flight (TOF) trigger at 60Hz. The analysis shown in this section was performed on this data set. Some modifications were made to the event

display in order to have it find cosmic tracks which are expected to enter and exit the TRD barrel at opposite ends while moving in the $-y$ direction, that is from top to bottom. A simple strategy was employed that involved finding a track in one of the top five sectors (2, 3, 4, 5, or 6) and then another track in one of the bottom five (11, 12, 13, 14, or 15). If an event had at least one track in both a top sector and a bottom sector, the event display was configured to display it. The result was several well defined cosmic tracks. Two examples are shown in Figure 8.1. This phase of data taking and analysis was vital in terms of testing software on live Run 3 data and identifying and resolving a number of bugs in the raw data reader.

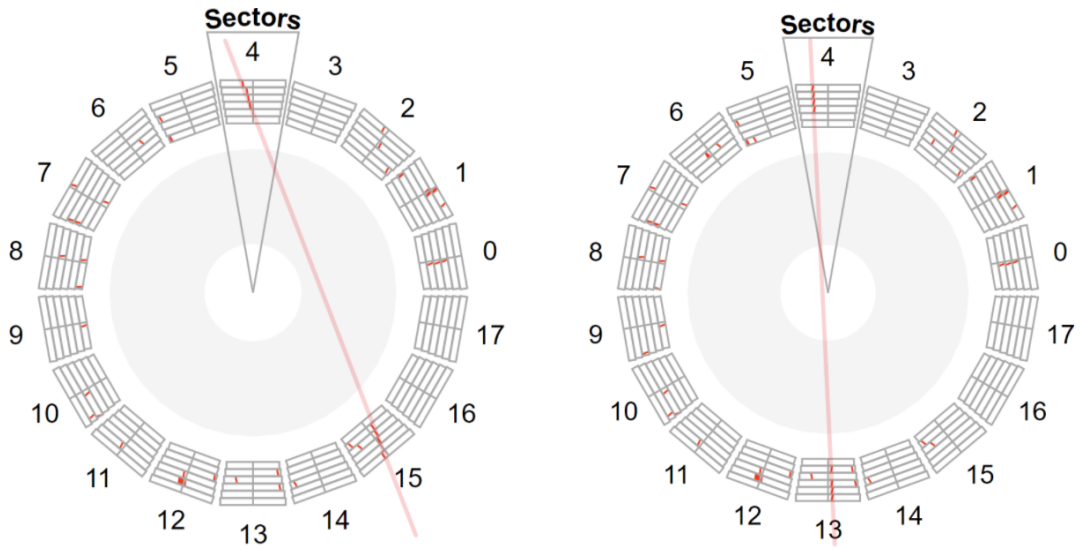

Figure 8.1: Cosmic tracks from run 502238 shown using the TRD event display. The red line joining tracks in the top and bottom sectors is drawn by hand and added only as a visual guide for a proposed cosmic track.

A study was conducted into the rate of energy loss of cosmic muons through the TRD. In theory, it should be found that tracks in the bottom sectors deposit slightly more energy into the detector due to transition radiation. The energy of the transition radiation is proportional to the γ of the cosmic muon beyond a certain threshold of around 10^3 [47]. Only a small subset of the incident muons will have energy above this threshold and therefore the difference between energy deposits in the top and bottom layers is expected to be small. The presence of transition radiation will not be found at all in the tracks that pass through any of the top sectors since the muon will pass through the drift chamber before it passes through the radiator. It is essentially moving in the wrong direction relative to chambers in the top group of sectors. Code was adapted from [48] to plot pulse height spectra for the top and bottom groups of sectors. Only digits from tracklets matched to tracks were used. Tracking was done using the modified event display discussed above.

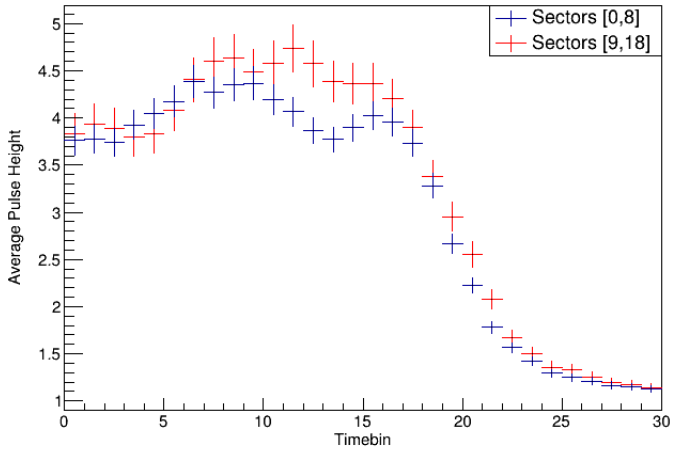

Figure 8.2: Pulse height of cosmic tracks through the TRD. Pulse heights are shown for tracks in the top and bottom groups of sectors separately. Transition radiation is only expected in the bottom group of sectors (red) due to the improper orientation of chambers in the top group (blue) of sectors relative to incident muons.

In Figure 8.2, the difference in energy loss in the top and bottom groups of sectors can be seen. A small rise in the average pulse height of the bottom-sector tracks (red) can be seen between timebins 10 and 17 when compared to the top-sector tracks (blue). This bears a minor resemblance to the transition radiation plot shown in Figure 1.7. However, the difference is minimal and other unexpected features are observed as well such as the rise in both plots around timebins 5 to 10. Furthermore, the amplification peak is not observed due to slow trigger timing from the TOF detector. Therefore the preliminary result is compatible with expectation given the issues mentioned, but further investigation is required after those issues have been resolved.

8.3 Analysis of Pilot Beam Data

With the updated TRD event display, it was possible to get a first look at the pilot beam data. Specifically, run 505413 which was a *pp* run done with 0 B field. This run was chosen to better accommodate the rudimentary tracking algorithm of the event display which does not account for any curvature of the track. First tracks were successfully visualized and when looking at the data, it was observed that the position of the tracklets did not match very well with the digit data in the lower right window of the event display. There was frequently a significant offset along the local y axis. By reviewing a C++ implementation of the TRAP code [49], it was discovered that the pad position granularity in the `Tracklet64` word was not recorded correctly in O^2 and therefore, did not match with the TRAPs calculation. The granularity in the TRAP code was 40 steps per pad, whereas in O^2, it was 80. This resulted in the large offset being observed after reconstruction. The plots also aided in discovering that it was necessary to xor the slope values with 0x80 during the calculations and this was not being done. Once corrected, the tracklets began showing significantly better agreement with the digit data. In Figure 8.3 a typical example of the offset issue is shown. The digit data can be seen to be offset to the right of the tracklet although it is expected that the tracklet be drawn directly on top of the digit data. By eye, the expected position of the tracklet can be approximated to be about pad 11. Instead the tracklet is found at pad 8. The slope of the tracklet also appears to be incorrect, but this was to be expected since no

calibration had been done at this point.

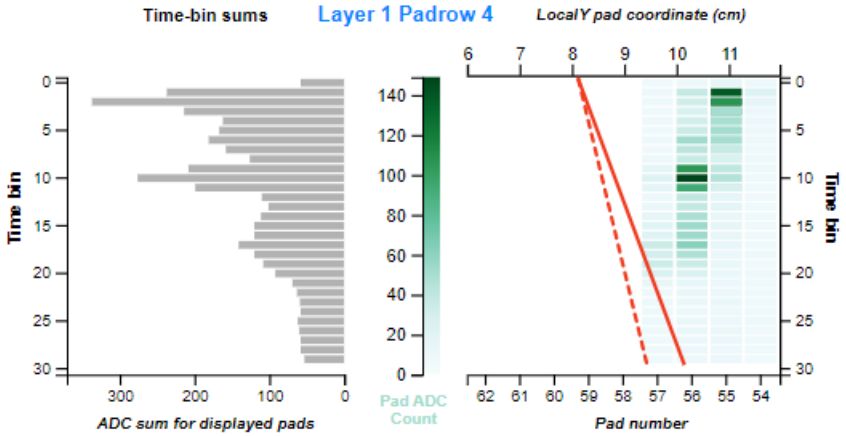

Figure 8.3: An offset is observed between the tracklet position and the digits. Above is a typical case where the tracklet position is observed to be approximately 3 pads to the right of where it should be. In general, offsets were observed in both directions. Some disagreement in tracklet slope was expected since no calibration had yet been done. Progress was made in resolving this offset.

Histograms were plotted showing the offsets between tracklet and digit position for a number of events before and after the fix. This can be seen in Figure 8.4. The histogram is filled with Δy which is the calculated distance between the tracklet position and the digit position. The digit position is calculated by taking the ADC weighted mean of the digit positions for the first five time bins using

$$\begin{aligned} \Delta y &= y_{\text{digit}} - y_{\text{trklt}} \\ &= \frac{\sum_p \sum_{t=0}^{5} p q_t}{\sum_{t=0}^{5} q_t} - y_{\text{trklt}} \end{aligned} \tag{8.1}$$

Where p refers to the pad number of the digit and q_t refers to the charge deposit (ADC) in the corresponding timebin t. The tracklet position is calculated in the usual way. Digits are only matched to tracklets if they are within a range of 8 cm on either side of the tracklet position. Since inner pads have a width between 6.35 mm and 7.85 mm, the histogram should be able to display at least 10.2 pads in either direction. In Figure 8.4, it can be seen that the standard deviation of 2.685 pads is higher than expected and, in general, the tracklets and digits do not match up correctly. After changing only the tracklet position granularity from 80 steps per pad to 40 steps per pad, the histogram changes shape into what can be seen on the right-hand side of Figure 8.4. The peak is significantly more pronounced and the standard deviation has decreased to 1.239 pads. It is clear that the change in the tracklet position granularity has significantly improved the agreement between tracklets and digits. It was observed that the peak is shifted from 0 slightly to the left with a mean of -0.769 pads. At a later time to this work, a bug was discovered in O^2 that had been causing a shift of 0.5 pads so the error above can largely be attributed to that. Δy (pads)

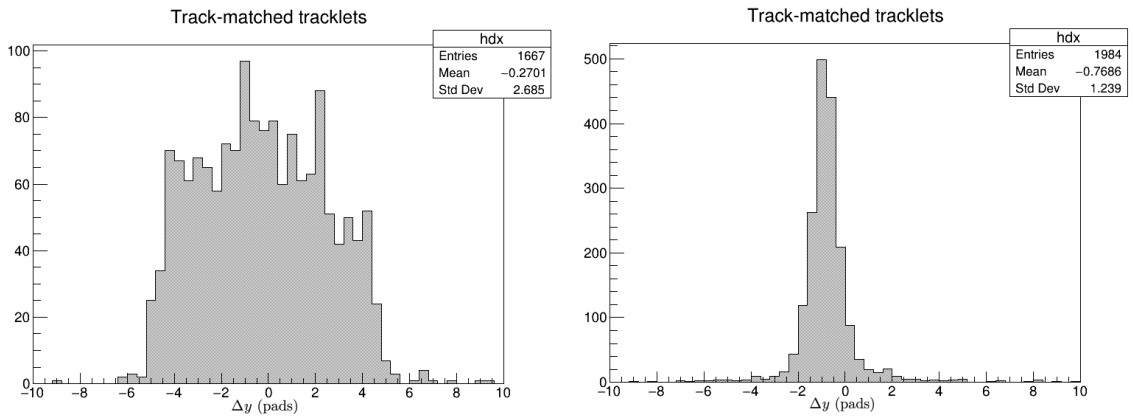

Figure 8.4: Histograms of the measured offsets, Δy, between tracklet and digit positions before (left) and after (right) reconciling the pad granularity in O^2 with the TRAP code. Only tracklets that were matched to tracks are used.

8.4 Noise

An investigation into pad noise was halted by a number of irregularities in the TRD data that prevented the study from reaching a conclusion. The purpose of the study was to identify and mask noisy pads which would otherwise contribute unreliable data to the read out stream. The data examined was from noise run 504419 of October 2021. This was recorded at a rate of 100 Hz with with no beam, a random trigger, and no zero suppression. This ensures that the data is largely void of any particle interactions aside from the occasionally cosmic muon. The remaining signal is caused only by the read out electronics of the TRD itself. The status of the investigation is reported below.

The primary plot that was examined was a collection of 2D histograms showing the distribution of the RMS of the ADC counts for each pad in sector 0. A total of 1000 time frames are parsed and the square of the ADC counts, q_t, in each time bin are recorded for all digits. The number of ADC counts, n, is also stored. Then the RMS is calculated in the usual way with

$$\text{RMS} = \sqrt{\frac{1}{n}\sum_t {q_t}^2 - \quad \langle q_t \rangle^2} \tag{8.2}$$

The results are shown in Figure 8.5.

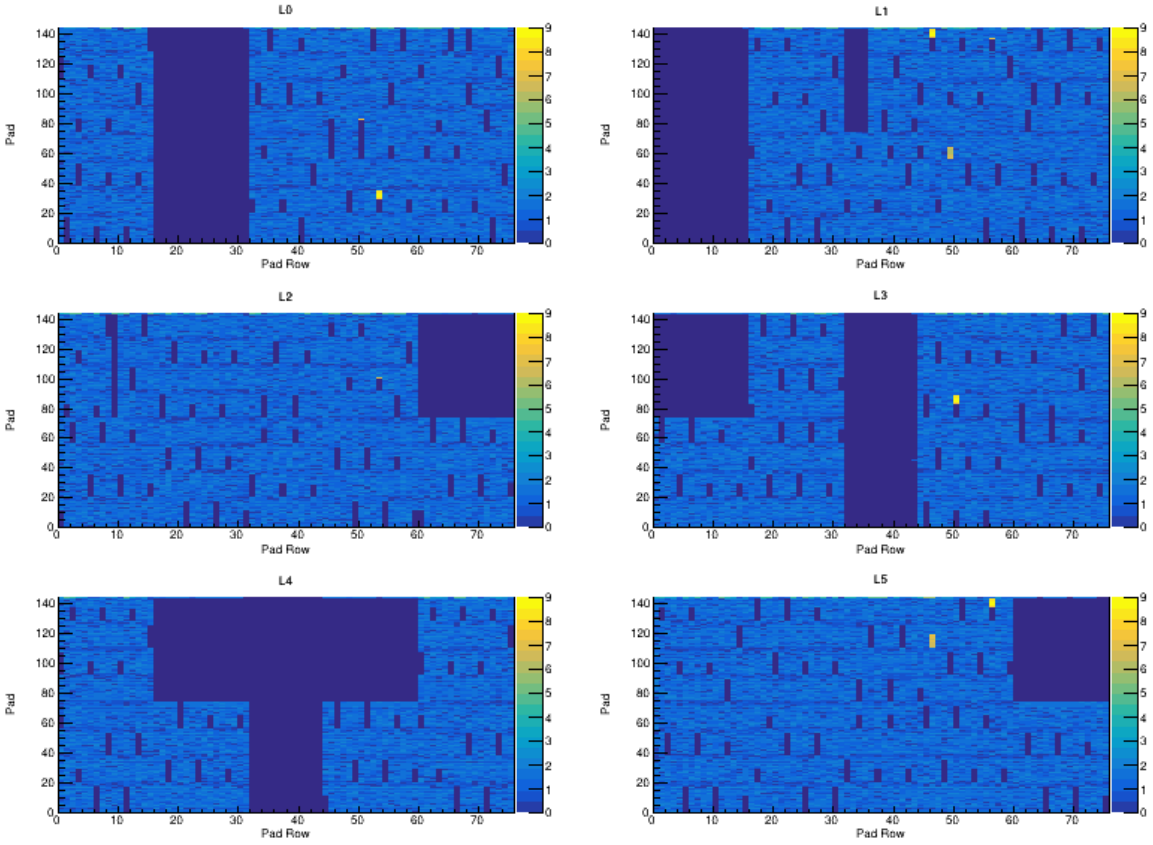

Figure 8.5: The RMS of the ADC counts in each pad of sector 0. Each sub plot displays the same layer in all 5 stacks. Therefore, there are 6 sub plots and each plot has a pad row range that extends up to $16 \times 4 + 12 = 76$.

A number of features are noted in Figure 8.5. Firstly, the RMS takes values approximately between 0 and 8. This is higher than expected. In previous studies, noisy pads have been identified with an RMS close to 3 [25][15]. In some cases, for example in layer 2, a singular noisy pad can be identified, but more generally speaking, the noisy pads appear in groups of several pads as indicated by the longer strips of yellow in other layers. Layer 4 appears to have a much narrower distribution of noise in the range of 0 to 1.8 with no obvious outliers able to be recognized. The second obvious feature is the presence of a multitude of holes of varying sizes indicated by the purple regions of 0 RMS. Loosely speaking, the holes appear to come in two different sizes. Both are caused by a lack of any digit data in these regions. The large holes were discovered to correlate perfectly with a map of TRD Link Monitor Errors (LME). This comparison is shown in Figure 8.6. A LME is relative to a particular half-chamber and is an indication that the fibre optic data transmission line to that half-chamber has gone dead. This can be caused by a number of factors including the occurrence of too many readout errors or a decision to leave the link in the ”off” state for the duration of the run. The result is that no data is received from the half-chamber. In Figure 8.6, each large empty square would correspond to a single half-chamber and a hole covering the full vertical range corresponds with a chamber. Demonstrating this correlation proves that the presence of the large holes is understood. The large hole left of the centre of the layer 1 plot in Figure 8.6 is assumed to be a single ROB given its size.

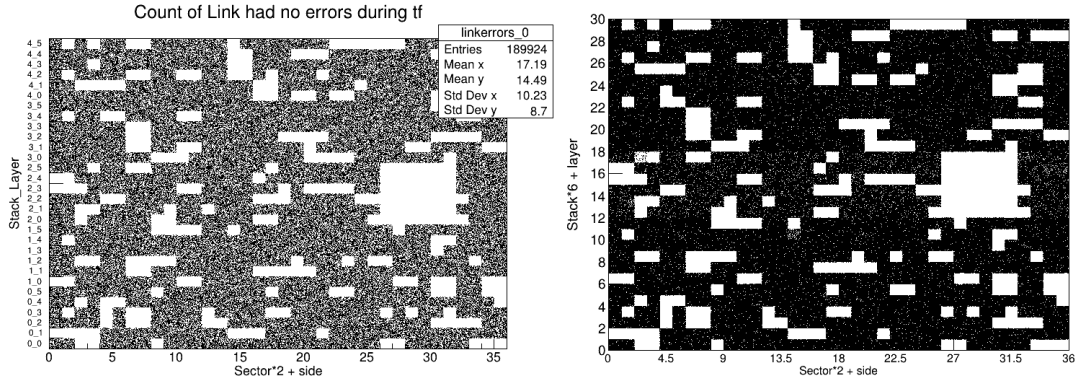

Figure 8.6: A correlation is demonstrated between the large holes shown in Figure 8.5 caused by missing half-chambers and the distribution of LMEs. The plots are 2D histograms with a range of the entire TRD ordered by half-sector in the horizontal and layer in the vertical. On the left is shown the distribution of link errors which result in the failure of a chamber to send data to the Common Readout Unit (CRU). On the right is shown the distribution of digits. The correlation is exact with regard to the distribution of holes.

In addition to the several large holes, an abundance of smaller holes is observed. These correspond with groups of pads. The sizes of the holes are not consistently 18, indicating that the problem is not local to particular MCMs. Additional plots were made of pad number as a function of detector and pad row and the missing pads were seen again. Through these visualizations, it was eventually discovered that the CRU was skipping a few dwords at the end of each 8 KB raw data header page.

Chapter 9

Conclusion

In this thesis, a significant contribution to the recommissioning of the ALICE TRD has been documented. It has involved work done most notably in the areas of calibration, core software maintenance, and various implementations within the O^2 framework.

In chapter 3, the migration of core TRD software from subversion to git was documented. All packages were successfully ported while preserving full authorship history. Continuous integration was configured on all relevant GitLab repositories and secure and public yum repositories have been established in the DCS network at CERN as well as on the TRD website respectively. Similarly, ipkg repositories were also set up which can be accessed by any machine connected to the internet via the public repository as well as by the DCS boards via the repository at CERN. The software was modified where necessary and has been shown to run on CentOS 7 machines. Using Ansible, the software was configured on the new Linux nodes at P2 although this remains a work in progress.

In chapter 4, the development of the `AliRoot` v_D and $E \times B$ calibration software was described. This involved gaining a deep understanding of the geometry of the TRD as well as the way in which tracklets transform between different coordinate systems and through varying v_D and α_L. The calibration software was tested on Run 2 data and found to perform well although the performance varied across the TRD. In chapter 5, quality control was done to assist in interpreting the results from the test calibration. These results were examined in more detail in chapter 6. The angular resolution achieved was represented by the Gaussian σ from fits done to distributions of $\Delta\alpha$. The mean value of this quantity was measured to be $\bar{\sigma} = 2.6°$. This was compared to a previous result from 2020 and found to be a factor 1.5 to 2 times higher. Reasons were offered to explain this discrepancy. Position resolution was also investigated and found to be approximately 1.2 mm which fell well within the range of reason.

Chapter 7 consisted of the documentation of three primary projects in the O^2 framework. The first was the TRD `TrackletTransformer` which was written to transform the coordinates of raw tracklets into the tracking frame and apply calibration parameters accordingly. The second project involved the addition of two new methods to the `PadPlane` class. The new methods accommodated transformations from pad coordinates into the local chamber reference frame. The functions were novel in that they returned floats instead of integers in order to preserve full information during the transformation. This became useful in testing the implementation of the pad tilting effect in O^2 as well as other efforts towards debugging and analysis. As a result of this investigation, a bug in one of the `PadPlane` methods related to pad tilting was identified and resolved. The final project involved improvements to the TRD event display and integrating it with O^2. Improvements included noise reduc-

tion as well as a new tracking algorithm, allowing events with tracks to be isolated. The improved TRD event display was made live on the TRD website with a selection of data sets.

The final chapter 8 was the most recent chronologically and documented the cosmic and pilot beam runs of late 2021. Analysis was done on both sets of data and a number of issues were successfully debugged. Cosmic tracks were discovered and visualized with the event display. Visualizations of pilot beam data contributed majorly to a discovery of a severe discrepancy between tracklet calculations in the TRAP and calculations done in O^2. The issue was resolved through reconciliation of the tracklet position granularity. An investigation was conducted into pad noise, but the results were inconclusive due to further issues with the data. The issue was later resolved by the collaboration.

Appendix A

SVN Authors

The list of authors used during the migration of the core TRD packages from SVN to git is shown below.

bbathen = Bastian Bathen <bastian.bathen@uni-muenster.de>
angelov = Venelin Angelov <angelov@physi.uni-heidelberg.de>
demscher = David Emschermann <david.emschermann@cern.ch>
dietel = Tom Dietel <tom@dietel.net>
doenigus = Benjamin Donigus <benjamin.doenigus@cern.ch>
fkramer = Frederick Kramer <frederick.kramer@gmx.de>
gunji = Taku Gunji <Taku.Gunji@cern.ch>
jkl = Jochen Klein <Jochen.Klein@cern.ch>
kschweda = Kai Schweda <kschweda@physi.uni-heidelberg.de>
lippmann = Christian Lippmann <christian.lippmann@cern.ch>
mwalt = Matthias Walter <m_walt06@uni-muenster.de>
oyama = Ken Oyama <ken.oyama@cern.ch>
sicking = Eva Sicking <eva.sicking@cern.ch>
sschmied = Stefan Schmiederer <stefan.schmiederer@cern.ch>
tbird = Thomas Bird <thomas.bird@cern.ch>
uwest = Uwe Westerhoff <uwesterhoff@uni-muenster.de>
weiser = Dennis Weiser <dennis.weiser@gmx.net>
helge = Helge Grimm <hinfo@helgegrimm.de>
tanabe = Daisuke Watanabe <daisuke.watanabe@cern.ch>
cbaumann = Christoph Heinrich Baumann <unimail@baumann.es>
obusch = Oliver Busch <o.busch@gsi.de>
pconst = Paul Constantin <pconst@physi.uni-heidelberg.de>
vallero = Sara Vallero <svallero@to.infn.it>
watanabe = Daisuke Watanabe <daisuke.watanabe@cern.ch>

Appendix B

CERN CentOS 7 on WSL2

With Windows' new Windows Subsystem for Linux (WSL) feature, it is possible to run an optimized Linux virtual machine inside the Windows operating system. With WSL2, this virtual machine operates a full Linux kernel with improved file system performance and native GUI application support. Access to a Linux shell is immediate via a terminal application like Windows Terminal making transitioning between the Windows desktop and Linux shell seamless. For users with Windows machines, this may be an appealing option. The user can benefit from optimized drivers and Windows proprietary software while being able to work inside a genuine Linux environment. Alternatives like configuring a dual-boot or running a Linux virtual machine using software like VirtualBox feel cumbersome in comparison.

With this in mind, the author created a GitHub repository with a few files and instructions that assist in setting up a custom WSL distribution as well a pre-built CERN CentOS 7 (CC7) distribution. The root file system for this distribution was extracted from the official CC7 Docker image. After testing other distributions including Ubuntu 18.04, Ubuntu 20.04, and stock CentOS 7, CC7 was found to be the most compatible with the ALICE O^2 software stack and made building it much easier and more consistent.

Appendix C

TRD Homepage

Previously, a TRD website had existed at the GSI research facility here. For centrality, it was preferable to move this website into its own CERN sub-domain on the EOS web hosting service. The entire website was cloned from the GSI page and placed in a GitLab repository. It was then published at alicetrd.web.cern.ch from `/eos/project/a/alice-trd/www/` on `lxplus`. A few minor edits were made. The GSI logo on the homepage was replaced by the official ALICE logo and several pages had defunct "data privacy" links removed. Additionally, authentication was configured. The EOS web hosting service includes an OpenID Connect module that makes securing pages simple. Access to a web folder which may contain any number of HTML pages may be restricted by adding a `.htacces` file. The file should contain the two lines

```
AuthType openid-connect
Require valid-user
```

Any user who attempts to access restricted material will be required to log in through the CERN Single-Sign On (SSO) portal. After logging in, they will be redirected back to the secure page with full access. Access is controlled recursively to all sub folders and pages from where the `.htaccess` file is located. The above two lines allow any CERN user to gain access. Alternatively the second line may be substituted for something of the form `Require claim cern_roles:<role_identifier>` in order to grant access only to users of particular e-groups. More information is available at the official EOS web documentation page. To edit the website, the repository should be cloned to the local working machine where edits can then be made. To transfer the edits to lxplus and publish them, an ssh connection to lxplus should be established and then `git pull` may be run from the web directory mentioned above.

Appendix D

TRD Kalman Filter Seeding

Work was done by a Heidelberg university student called Sven Hopner supervised by Alex Schmah on a Kalman filter implementation of TRD self-tracking.

Kalman filtering requires the use of an initial seed which is 2-3 starting tracklets from which a track can be constructed. Two methods of track finding will be discussed here.

D.1 Linear Fitting

By this method, a random pair of tracklets is selected in layers 5 and 4. A straight line is then drawn between these two and the point of intersection with the next chamber (in layer 3) is determined. Then, the program searches for any tracklets which are near this point of intersection. The program checks a number of conditions, including the proximity of the tracklet to the point of intersection, the direction of the tracklet compared to the directions of the preceding tracklet, and the number of tracklets which are within the acceptance radius. If a suitable tracklet is found, a new straight line is drawn between this tracklet and the previous tracklet in layer 4. The process is then repeated iteratively until all layers have been explored.

In more detail: The program is written to loop over all chambers first in layer 5 and then in layer 4 in order of increasing chamber number (i.e. from sector 0 through to 18). It will select the first chamber in layer 5 and then begin iterating over all the tracklets in this chamber. For each tracklet it selects in the chamber in layer 5, it will iterate over all tracklets in the chamber directly below it (in the radial direction) in layer 4. For each of these pairs, a straight line is drawn in 3D space connecting the two tracklets. At this point, the geometry needs to be well understood. The TRD layers are approximated as concentric cylinders with one cylinder corresponding to one layer. The cylinders will have some thickness determined by the width of the chamber, but this is not important in the geometry of this problem. What is important is where the surface of the cylinder is. The surface of each cylinder whose normal vector points radially outward from the beam axis will begin approximately where the anode plane of the chambers is found. This geometry is not thought of as static across each solution, but rather is constructed and solved individually for each case. A general case is described below.

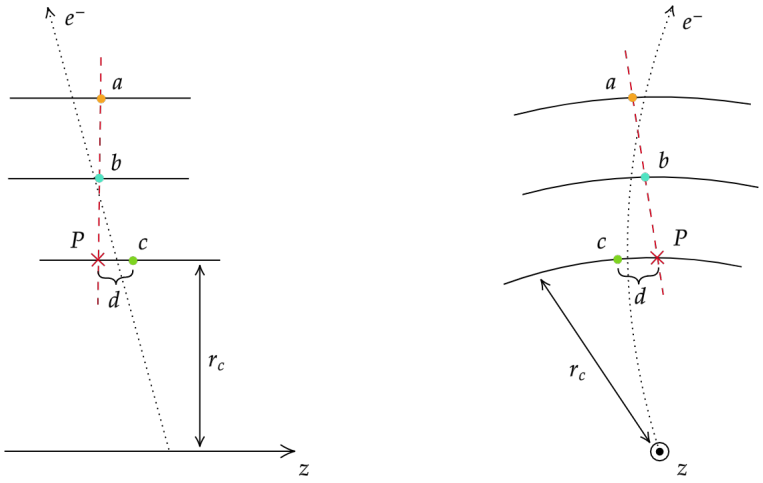

Figure D.1: TRD line fit geometry from a front-on view down the beam axis (right) and a side-on view along the beam axis (left). a, b, and c represent tracklets in 3 successive layers in descending order

Tracklet a is selected in layer n (5 or 6) and a second tracklet, b, is selected in layer $n-1$. The difference in angle between tracklets a and b is measured which we call $\Delta\theta_{ab}$. If $\Delta\theta_{ab}$ is too large, then a new b is selected. Then, for every possible third tracklet, c, in layer $n-2$, the distance, d, is measured in 3D space between c and P which is the point of intersection of the straight line connecting a and b with the layer in which tracklet c resides. P is determined by finding the point along the line connecting a and b which has the same radius relative to the beam axis as tracklet c. Every possible tracklet c in layer $n-2$ is iterated over and then the tracklet, c corresponding to the minimum d measurement is selected. A final check is done to ensure that d_{min} is within the acceptance threshold and that $\Delta\theta_{bc_{\text{min}}}$ is also within the acceptance threshold. The number of candidate c tracklets which fall within the offset acceptance radius is also counted. If this number is greater than 3, the tracklet c is discarded. This is to avoid overlapping tracks and mis-matches. If tracklet c_{min} passes the tests, then it is selected and a new straight line is drawn joining b and c. Then the whole processes is repeated in layer $n-3$. If d is not below the specified threshold, then c is replaced by another tracklet in layer $n-2$ until either a match is found or all possible candidates in this layer are exhausted. A track is found when enough tracklets have been matched together in this simple way. Once the track has been saved, all tracklets of which it consists will be ignore by track finding algorithm as it continues searching. An example of some of the located tracks are show below in figure D.2 a 3D visualization of the TRD.

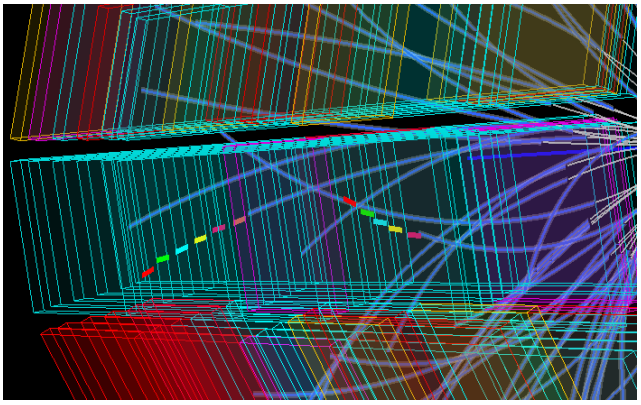

Figure D.2: Two tracks located by the straight line seeding algorithm. Each colour represents a different layer. The track to the left consists of 6 tracklets (layers 5-0) while the track on the right-hand side consists of only 5 (layers 5-1). These two tracks also have matching TPC tracks shown as faded blue curves. The colours of the detectors indicate information relating to QC.

All code can be found on the TRD self tracking repository [50].

D.2 Tracklet Path Fitting

This method was found to be significantly less effective in locating good tracks, but could perhaps become more useful as various other improvements are made. This method functions similarly to the previous method, with one crucial difference. Instead of drawing a straight line between tracklets a and b and finding the intersection of this line with the next layer in which a tracklet, c is located, this method draws a line from tracklet a in the direction of tracklet a and finds the point of intersection with the layer in which a tracklet b is located. Then d is measured between tracklet b and this intersection point. If tracklet b is within the acceptance radius, a new line is drawn from b in the direction of b and a matching c in the next layer down is searched for.

This method showed some promise, but did not work well since the tracklet directions were not found to be reliable enough. Many tracks could be found with the human eye which did not have well aligned tracklets, but were still observed to be valid tracks. This method was tested before the final calibration model was put in place and may have become more effective after this, but was not tested again.

For either of these methods, the acceptance radii for both the

Bibliography

[1] A. M. Helmenstine, "A brief history of atomic theory," ThoughtCo, Aug 2020. [Online]. Available: www.thoughtco.com/history-of-atomic-theory-4129185

[2] K. S. Schmitz, "Chapter 1 - philosophy of science," in *Physical Chemistry*, K. S. Schmitz, Ed. Boston: Elsevier, 2018, pp. 183–367. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128005132000012

[3] MissMJ, Cush, "Standard model of elementary particles," Wikimedia Commons, September 2019. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/0/00/Standard_Model_of_Elementary_Particles.svg

[4] R. A. Meyers, *Encyclopedia of physical science and technology*. Academic, 2002.

[5] A. Kalweit, "The physics of heavy-ion collisions," 2017. [Online]. Available: https://indico.cern.ch/event/598530/contributions/2547248/attachments/1517618/2369646/HeavyIonLecture_Lecture3.pdf

[6] ALICE Collaboration, "A large ion collider experiment: Physics," 2021.

[7] W. Busza, K. Rajagopal, and W. Van Der Schee, "Heavy ion collisions: the big picture and the big questions," *Annual Review of Nuclear and Particle Science*, vol. 68, pp. 339–376, 2018.

[8] CERN, "ALICE," 2022. [Online]. Available: https://home.cern/science/experiments/alice

[9] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. A. Khalek, A. A. Abdelalim, R. Aben, B. Abi, M. Abolins, O. AbouZeid *et al.*, "Observation of a new particle in the search for the standard model higgs boson with the ATLAS detector at the LHC," *Physics Letters B*, vol. 716, no. 1, pp. 1–29, 2012.

[10] X. Vidal and R. Manzano, "LHC pb collisions," 2019. [Online]. Available: http://w3techs.com/technologies/overview/content_language/all

[11] CERN, "CERN's accelerator complex," 2021. [Online]. Available: https://home.cern/science/accelerators/accelerator-complex

[12] "LHC layout: Taking a closer look at the LHC," LHC Closer, 2020. [Online]. Available: https://www.lhc-closer.es/taking_a_closer_look_at_lhc/0.lhc_layout

[13] L. Betev and P. Chochula, "Definition of the ALICE coordinate system and basic rules for sub-detector components numbering," *ALICE Offline, Geneva*, 2003.

[14] ALICE Collaboration and others, "ALICE Technical Design Report of the Transition Radiation Detector," *CERN/LHCC*, vol. 21, 2001.

[15] D. Emschermann, "Construction and performance of the ALICE transition radiation detector," Ph.D. dissertation, University of Heidelberg, 2010.

[16] ALICE Collaboration, "TRD offline software writeup," 2009.

[17] ALICE Collaboration and others, "The ALICE transition radiation detector: construction, operation, and performance," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 881, pp. 88–127, 2018.

[18] S. Myers, "Large hadron collider commissioning and first operation," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, no. 1961, pp. 859–875, 2012.

[19] E. Martelli, "Lhcopn update," March 2021. [Online]. Available: https://indico.cern.ch/event/983436/contributions/4221337/attachments/2213678/3747087/LHCOPNE-20210324-46-LHCOPN-update.pdf

[20] S. M. Panebianco, "ALICE detector upgrade for run 3," ALICE Collaboration, July 2020. [Online]. Available: https://indico.cern.ch/event/868940/contributions/3813685/attachments/2081068/3495497/Panebianco_ICHEP2020.pdf

[21] B. Abelev, ALICE Collaboration *et al.*, "Upgrade of the ALICE experiment: letter of intent," *Journal of Physics G: Nuclear and Particle Physics*, vol. 41, no. 8, p. 087001, 2014.

[22] M. Fasel, "Upgrade of the ALICE experiment for LHC run 3 and beyond," ALICE Collaboration, April 2021. [Online]. Available: https://indico.bnl.gov/event/9726/contributions/46030/attachments/33844/54487/upgradeALICE.pdf

[23] T. Dietel, private communication, March 2021.

[24] G. Eulisse, P. Konopka, M. Krzewicki, M. Richter, D. Rohr, and S. Wenzel, "Evolution of the ALICE software framework for run 3," in *EPJ Web of Conferences*, vol. 214. EDP Sciences, 2019, p. 05010.

[25] J. Klein, "Commissioning of and preparations for physics with the transition radiation detector in a large ion collider experiment at cern," Master's thesis, Heidelberg University, 2008.

[26] V. Angelov *et al.*, "ALICE TRAP user manual," University of Heidelberg and University of Mannheim and University of Kaiserslautern and GSI Darmstadt, Tech. Rep., 2005.

[27] U. Westerhoff, "The FEE Server control engine of the ALICE-TRD," Master's thesis, Westfälische Wilhelms-Universität Münster, 2009.

[28] TRD Group, "alitrd GitLab," 2020, URL: https://gitlab.cern.ch/alitrd.

[29] CERN Collaboration, "ci-web-deployer," 2020, URL: https://gitlab.cern.ch/ci-tools/ci-web-deployer.

[30] O. Hartkopp *et al.*, "SocketCAN - controller area network," 2022, URL: https://www.kernel.org/doc/html/latest/networking/can.html.

[31] ANAGATE, "Anagate: Software Downloads," 2022, URL: https://www.anagate.de/support/download.php.

[32] J. Wiechula, "Everything you wanted to know about the TPC but were afraid to ask," 2013. [Online]. Available: https://alice-analysis.web.cern.ch/sites/default/files/documents/Analysis/JensJD.pdf

[33] R. Bailhache, "Calibration of the ALICE transition radiation detector and a study of z^0 and heavy quark production in *pp* collisions at the LHC," Master's thesis, Technical University of Darmstadt, 2008, URL: http://cds.cern.ch/record/1320722/files/CERN-THESIS-2008-143.pdf.

[34] A. Schmah, A. Berdnikova, and J. Barrella, "TRD-Run3-Calibration," 2020, URL: https://github.com/aschmah/TRD-Run3-Calibration.

[35] O. Schmidt, "Space point calibration of the ALICE TPC with track residuals," Ph.D. dissertation, University of Heidelberg, 2020.

[36] A. B. Alexander Schmah, "Trd drift velocity calibration," March 2020.

[37] "xyscan distribution page," 2020, URL: https://rhig.physics.yale.edu/~ullrich/software/xyscan/.

[38] J. Klein, "Jet physics with a large ion collider experiment at the large hadron collider," Ph.D. dissertation, Heidelberg University, 2014, URL: https://cds.cern.ch/record/1973326/files/CERN-THESIS-2014-186.pdf.

[39] ALICE Collaboration, "Offical QA results run 265338," 2016. [Online]. Available: http://aliqatrd.web.cern.ch/aliqatrd/data/2016/LHC16q/pass1_CENT_wSDD/000265338/

[40] A. Schmah, private communication, 2020.

[41] R. Shahoyan, "ALICE continuous readout and data reduction strategy for run3 and 4," ALICE Collaboration, November 2019. [Online]. Available: https://indico.cern.ch/event/773049/contributions/3581368/attachments/1935979/3211474/chep19_rs_final.pdf

[42] S. Perumal, "Designing an event display for the Transition Radiation Detector in ALICE," Master's thesis, University of Cape Town, 2020, URL: https://cds.cern.ch/record/2753183/files/CERN-THESIS-2020-286.pdf.

[43] S. Perumal, "msc cpp," 2019, URL: https://github.com/samperumal/msc-cpp.

[44] A. Berdnikova, private communication, November 2021.

[45] J. Barrella, "alice-trd-event-display," 2022, URL: https://github.com/jbarrella/alice-trd-event-display.

[46] S. Perumal, "alice-trd-event-display," 2021, URL: https://github.com/samperumal/alice-trd-event-display.

[47] B. Dolgoshein, "Transition radiation detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 326, no. 3, pp. 434–469, 1993.

[48] ALICE Collaboration, "QualityControl," 2021, URL: https://github.com/AliceO2Group/QualityControl/blob/master/Modules/TRD/src/PulseHeight.cxx.

[49] V. Angelov, "C++ simulation of trap," 2021, URL: https://alice.physi.uni-heidelberg.de/viewvc/trd/wconfigurations/trunk/C/.

[50] A. Schmah, S. Hopner *et al.*, "TRD-Self-Tracking," 2020, URL: https://github.com/aschmah/TRD-Self-Tracking.