Characterisation of Silicon Pixel Sensors

For High-Energy Physics and beyond

Author: Jens Kröger, Universität Heidelberg

Manuel A. Del Rio Viera, DESY
Håkan Wennlöf, DESY

August 21, 2023

Contents

Contents

1 Overview

This lab course is an introduction to data analysis in the field of silicon pixel detector development for High-Energy Physics (HEP). The objectives are 1) to understand the working principle of silicon pixel detectors and 2) to analyse a set of test-beam data in order to characterise a pixel sensor prototype and investigate its performance.

Of course, actual data taking in a real test-beam environment at CERN or DESY is not possible within this course due to cost and time limitations. Therefore, this lab course is a pure analysis project, but a set of real data recorded at the Super Proton Synchrotron (SPS) at CERN, Switzerland, is provided.

The analysis is carried out with the help of the software packages $ROOT$ [1, 2] and *Corryvreckan* [3, 4], with which you will become familiar throughout the course. Step-by-step, you will build up an entire test-beam data reconstruction and analysis chain, starting with reading in the raw data, performing a quality check on the raw data, which under 'real circumstances' would be done during – or right after – data taking. This is followed by a more in-depth analysis of the performance of a sensor prototype.

1.1 Prerequisites

This lab course requires some basic knowledge and interest in the field of semiconductor physics and electronics. A particular preknowledge in particle physics is not necessary. A list of important keywords you should be able to explain is provided in Section 2.

This course can be carried out without prior knowledge in the area of data analysis or programming.

The analysis and data visualisation is carried out using $ROOT$ [2] and *Corryvreckan* [3]. These software frameworks are written in C++ but they are ready-to-use and well documented, i.e. the user does not need to go through the source code to understand how it works 'behind the scenes'. A user manual and online help are available and should cover most questions.

2 Introduction and Theory

When students think of data analysis in the context of particle physics, they will first and foremost think of the analysis of "fundamental" physics data such as data acquired at the Large Hadron Collider (LHC) at CERN, in order to examine the fundamental laws of nature or probe the particles of the standard model such as the Z boson or the Higgs boson [5, 6] and beyond. As you can imagine, this data is only available after many years of R&D in detector development and other fields such as accelerator physics.

This course focuses on one very important aspect of detector development: test-beam data analysis, a crucial ingredient to facilitate large-scale experiments as they are performed at the LHC.

Throughout this course, you will become familiar with the analysis and visualisation of test-beam data. You will acquire the following knowledge and skills:

- understanding of the functionality of modern silicon pixel sensors

- basic data analysis and data visualisation skills

2.1 A Short Introduction to Silicon Pixel Sensors

Pixel sensors are solid state particle detectors widely used in particle physics detector systems. Prominent examples from high-energy physics comprise the large experiments at LHC (ATLAS, CMS, ALICE, LHCb). In addition, pixel sensors are developed for smaller experiments like Mu3e [7] as well as in medical imaging applications like at the Heidelberg Ion Beam Therapy Centre (HIT) [8].

The fundamental working principle of modern pixel sensors is based on the properties of semiconductors, mostly silicon or germanium. Please make yourself familiar with the relevant topics discussed below not only by reading this manual but also in literature.

A very good introduction into the topic of detector physics is the (German) textbook **"Teilchendetektoren - Grundlagen und Anwendungen"** by Kolanoski and Wermes [9], which is also available in Englsih as **"Particle Detectors, fundamentals and applications"** [10]. Another English alternative is **"Pixel Detectors: From Fundamentals to Applications"** by Rossi et al. [11].

2.1.1 Semiconductor Physics

Pixel sensors make use of the properties and characteristics of semiconductors. To understand and interpret the results of the analysis during the lab course, it is important to have an understanding of the basic principles of the underlying physics.

Answering the following questions might help you to understand the relevant concepts for this laboratory experiment:

- What is a semiconductor?

- What is an electron-hole pair?

- What is doping of a semiconductor?

- What happens if a p-doped and an n-doped semiconductor are brought into contact?

- What is a depletion region?

- What happens when a voltage is applied across a diode?

- What is forward and reverse bias?

2.1.2 Interaction of Particles with Matter

As mentioned previously, a basic understanding of the interaction of particles with matter is necessary. Please refresh your knowledge on the following concepts and be able to explain them by answering the following questions:

- What is ionisation? What is the difference in gases and semiconductors?

- What does the Bethe-Bloch formula describe?

- What are minimum-ionising particles?

2.1.3 Silicon Pixel Sensors

Silicon pixel sensors (see Figures 1 and 2) are based on pn-junctions operated in reverse bias. When a charged particle traverses through the silicon, electron-hole pairs are created along the trajectory of the particle due to ionisation.

Usually, not the entire silicon is depleted. The electron-hole pairs created outside of the depleted volume of the diode diffuse in a random direction because no electric field is present. They either recombine after a certain time or they diffuse into the depletion region. Inside the depletion zone, the electron-hole pairs are separated by the electric field and drift towards the electrodes of the diode, the so-called collection electrodes. According to the **Shockley Ramo theorem** [9], these drifting charges induce a signal on the collection electrodes, which can be amplified and thus create a measurable electrical signal.

A pixel sensor consists of a 2-dimensional array of diodes. Each pixel corresponds to one diode with a small electronic signal-processing circuit. As illustrated in Figure 3, the induced signal is typically fed into an amplifier to make the signals large enough to be processed further. The amplified signal is then fed into a comparator. If the signal is larger than the (user-configurable) threshold of the comparator, the traversing particle is

detected as a hit. In many modern silicon pixel sensors, not only the 2-dimensional pixel address (column and row of the hit) are sent out, but in addition a hit timestamp or time-of-arrival (ToA) and the amount of charge (i.e. the size of the signal) is measured. The concept of the charge measurement as the time-over-threshold (ToT) is illustrated in Figure 18.
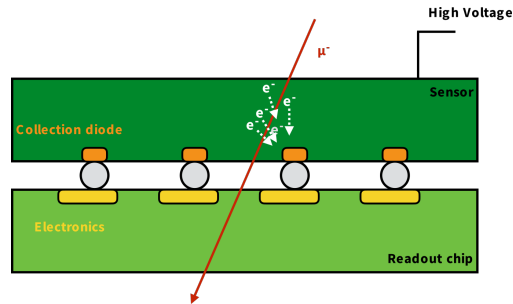
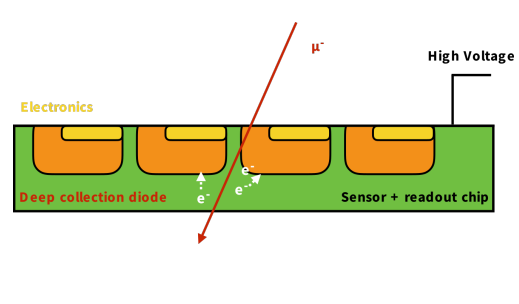Figure 1: Schematic drawing of a hybrid pixel sensor.

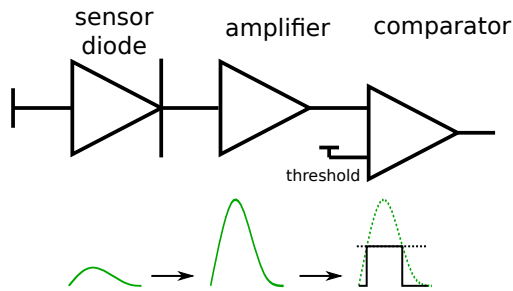Figure 2: Schematic drawing of a high-voltage monolithic sensor.

Figure 3: Illustration of the signal amplification and discrimination by the pixel electronics.
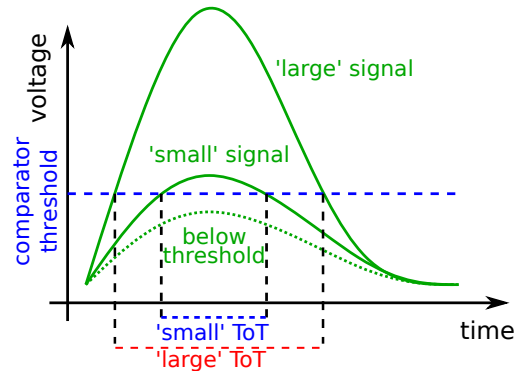
Figure 4: Illustration of the detection threshold and the **time-over-threshold (ToT)** measurement for different pulse heights.

Note: The ToT value can be translated into deposited charge by means of an energy calibration. However, this exceeds the scope of this course. Here, it suffices to understand that a large ToT value corresponds to a large induced signal in the pixel and a small ToT value to a smaller signal. In other words, the ToT spectrum corresponds to the spectrum of deposited energy by the ionising particles passing through the sensor.

A pixel hit cannot only be caused by a traversing ionising particle. 'Fake hits' can be caused by accidental crossings of the threshold due to (thermal) fluctuations of the amplifier output signal or the threshold or other external factors (e.g. bad solder connections etc.). This effect is called **noise**.

The readout electronics of the sensor contain logic to generate a digital data stream

('ones and zeros'), which is read out by the data acquisition system and stored for offline analysis.

Prepare to answer the following questions:

- How is a signal generated in a pixel sensor?

- What is a ToT measurement and how can it be translated into energy?

- Which spectrum is expected for energy loss of ionising particles in thin layers? What does this have to do with the time-over-threshold?

- What is noise? What can cause it?

2.1.4 Hybrid sensors

The "classical" design of solid state pixel detectors are **hybrid pixel sensors**. As depicted in Figure 1, they consist of two layers. The sensor layer acts as the active detection region. When a traversing ionising particle creates electron-hole pairs, the electrons drift in the electric field created by the applied high voltage towards a collection electrode. Usually, it is connected to a readout chip via small conductive balls, called bump-bonds. The readout chip contains the electronics such as amplifiers, discriminators, etc. and processes the signal induced on the collection electrode to form data packages, which can be sent out to the data acquisition system.

The *Timepix3* chips, which are used in the reference telescope in this exercise (see section 2.4), are hybrid sensors.

2.1.5 Monolithic pixel sensors

In contrast to hybrid sensors, monolithic pixel sensors consist only of one single layer of silicon, in which the detection volume and the readout electronics are combined (see Figure 2).

In **monolithic active pixel sensors (MAPS)**, a signal amplifier is located in each pixel cell. In **HV-MAPS**, an additional high voltage (HV) in the order of 100 V is applied as a reverse bias. This creates a strong electric field across the collection diode to allow for a faster charge collection and thus a larger signal and an improved time resolution.

Signal digitisation and data serialisation are usually implemented in a small dedicated area next to the active matrix of the chip, the so-called periphery.

The *ATLASpix_Simple* (see section 2.5), which will be the device-under-test in this lab course, is an example of a HV-MAPS.

Prepare to answer the following questions:

- What is the difference between hybrid and monolithic detectors? Can you think of advantages and disadvantages of the two technologies?

2.2 Sensor Characterisation

The design process of new pixel sensor prototypes is accompanied by an extensive design verification and simulation programme. However, this does not guarantee that they can be taken into operation by plug-and-play as they are often pushing the limits of technology well beyond what has proven to work in the past. Therefore, sensor prototypes need to be characterised and understood in detail once they have been produced.

Usually, the sensor characterisation comprises two complementary parts: laboratory testing and test-beam campaigns. In the laboratory, the sensors are first taken into operation. Typical steps of lab testing and measurements comprise:

- biasing tests (current vs. high-voltage characteristics)

- configuration tests

- optimisation of sensor settings (amplifier, voltages, etc.)

- measurement of the power consumption

- measurement of the noise rate

- energy calibration with radioactive sources or X-ray tubes (photons with well-known energies)

However, not all crucial parameters can be determined in the lab. X-ray tubes or portable radioactive source do not yield the necessary energies to penetrate multiple detector layers. However, this is necessary to achieve a high spatial and time resolution on the reference measurement of the incident particle. For this reason, high-energy particle beams and reference telescopes are used at large-scale test-beam facilities.

Measurements in test-beam campaigns comprise:

- spatial and time resolution

- hit detection efficiency

- detailed studies, such as hit detection efficiency across pixel matrix or within pixels

- comparison and optimisation of different operation parameters

- system integration tests, such as simultaneous operation and readout of many sensors in parallel or multiple subsystems

2.3 Test-Beam Facilities

Test-beam facilities are large-scale scientific user facilities, which allow relativistic particle beams to be used for the testing and characterisation of electronics, biological samples or new materials.

Due to the size and cost of such infrastructure, there are only a few places in Europe where test-beam campaigns can be conducted. Examples of institutes operating test-beam facilities in Europe are [12]:

- DESY in Hamburg, Germany

- CERN near Geneva, Switzerland

- Paul-Scherrer Institute near Zurich, Switzerland

2.3.1 The Super-Proton Synchrotron at CERN

The data provided for this lab course have been recorded at the Super-Proton Synchrotron (SPS) at CERN. As shown in Figure 5, the SPS is not only used as a pre-accelerator to fill the Large Hadron Collider, but it also provides particles for a range of smaller experiments and it serves as a test-beam facility. Figure 6 shows a photograph of the experimental hall at CERN, in which a number of experiments and test-beam setups are located.

The SPS has a circumference of around 7 km and accelerates protons to energies of up 450 GeV [13]. Some of these protons are directed onto a target in order to produce secondary particles. The particle beam provided to the test-beam setup shown in the red circle in Figure 6 normally consists of charged pions with a momentum of 120 GeV.

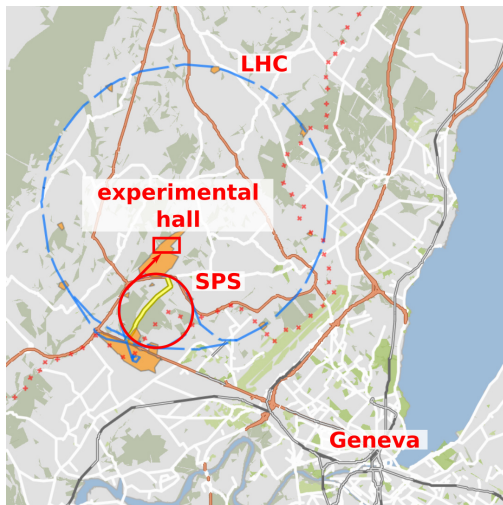Figure 5: Illustration of the CERN accelerator complex. The red arrow indicates the direction of the particle beam towards the experimental hall (red box) in the North Area at CERN. Screenshot from [14] (edited).
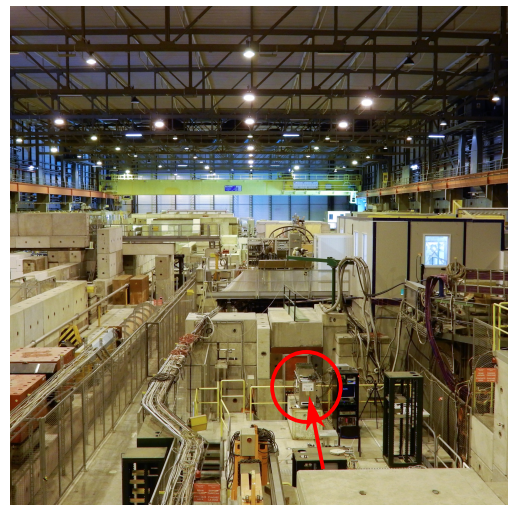

Figure 6: Photograph of the experimental hall in the North Area of CERN. The red line indicates the direction of the beam and the red circle marks the position of the reference telescope.

2.4 The Reference Telescope

The data provided for the lab course were taken with the *Timepix3* telescope of the CLICdp collaboration [15]. It is located in the experimental hall in the North Area at CERN as indicated in Figures 5 and 6. The telescope consists of seven planes of *Timepix3* readout chips [16] bump-bonded to silicon sensors with a thickness of 300 µm. Each Timepix3 consists of a matrix of 256×256 pixels with a pitch of 55×55 µm^2. In addition to the pixel address (column and row), the *Timepix3* provides a pixel timestamp (time-of-arrival) with a binning of 1.5625 ns (640 MHz) and a time-over-threshold (ToT) measurement with a range of 10-bit (0 - 1023) for each hit.

In short, the information we store for each *Timepix3* hit is:

- pixel address (column, row)

- pixel timestamp: time-of-arrival (ToA)

- pixel charge: time-over-threshold (ToT)

As can be seen in Figures 7 and 8, the telescope planes are rotated by ±9° around the x and y axes (the beam defines the z axis) so that particles pass through more than one pixel. This improves the achievable spatial resolution by increasing the amount of charge sharing. You will learn more about charge sharing in Section 3.2.2.

The hits on the planes of the reference telescope will be connected into tracks fitted by a straight line in the analysis. These tracks describe the trajectory of the ionising particles through all sensor planes. The telescope reaches a tracking resolution of ∼1 µm to 2 µm at the position of the **device-under-test (DUT)** with a timing precision of ∼1 ns [15].
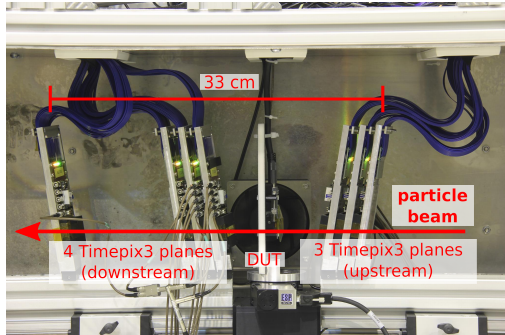

Figure 7: Photograph of the Timepix3 telescope at the SPS, CERN.
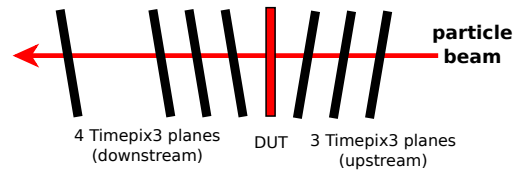

Figure 8: Schematic drawing of the Timepix3 telescope with a device-under-test (DUT) at the SPS, CERN.

2.5 The Device-Under-Test

The device-under-test (DUT) for this lab course is an *ATLASpix_Simple*. It is a High-Voltage Monolithic Active Pixel Sensor, which was designed as a test chip for the ATLAS

ITk upgrade [17] as well as the CLIC tracking detector [15]. After production, the sensors can be thinned down to $50\,\mu m$ by removing undepleted bulk material from the backside. A high bias voltage of up to $\mathcal{O}(100\,V)$ leads to a large depleted volume with a high electric field resulting in a fast charge collection via drift.

A photograph is shown in Figure 9 and a drawing is shown in Figure 10. The sensor has an active matrix consisting of 25 columns and 400 rows of pixels with a pitch of $130{\times}40\,\mu m^2$. As depicted in Figures 2 and 3, each pixel consists of a collection diode, which is connected to an amplifier and a comparator. After the comparator, the signal is sent to the digital periphery, where the pixel timestamp (ToA) is added. In addition, the digital periphery houses the readout electronics, which creates the data stream of 'ones and zeros' that is sent out from the chip.

For each hit, the time-of-arrival (ToA) is recorded with a binning of 16 ns. In addition, the signal charge is determined with a time-over-threshold (ToT) measurement with a 6-bit precision (0-63).

In short, the information we get from the *ATLASpix_Simple* is:

- pixel address (column, row)

- pixel timestamp: time-of-arrival (ToA)
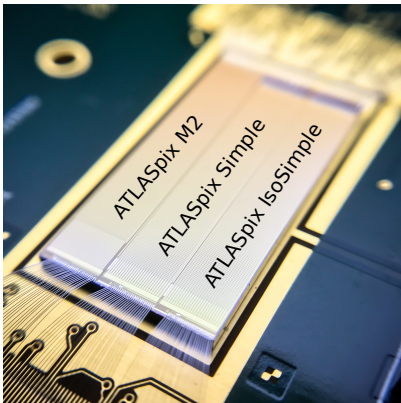
- pixel charge: time-over-threshold (ToT)

Figure 9: Photograph of an *ATLASpix_Simple* next to similar sensors glued and wire-bonded to a printed circuit board.
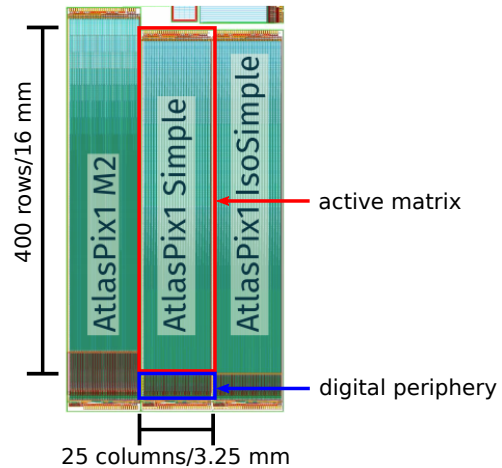

Figure 10: Micrograph of the *ATLASpix* sensors showing the geometric dimensions and the division into active matrix and digital periphery.

3 Basics of Test-beam Analysis

3.1 Overview of the Software

This section provides a brief overview of the software packages *ROOT* and *Corryvreckan*, which will be used during this course.

Figure 11: *ROOT* Data Analysis Framework. From [18].
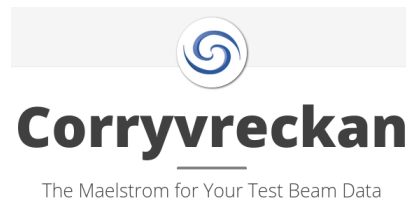
Figure 12: *Corryvreckan* – The Maelstrom for Your Test Beam Data. From [3].

3.1.1 ROOT

ROOT is a data analysis software package developed at CERN for usage in high-energy particle physics [2]. It provides all the functionalities needed for 'big data' and statistical analysis, as well as visualisation ('plotting') and data storage. Consequently, it is not only used in particle physics but also other fields of science and even industry, and many plots that you may have seen in lectures or publications, such as the famous Higgs discovery plots, have been made with *ROOT* (see Figures 13 and 14). More examples can be found here:
https://root.cern.ch/gallery
During this lab course, you will get familiar with *ROOT* and use it for the analysis and visualisation of your results.

3.1.2 Corryvreckan

Corryvreckan is a flexible and highly configurable software for the reconstruction and analysis of test-beam (and laboratory) data [3]. It was initially developed in the *CLIC Detector & Physics (CLICdp)* collaboration at CERN [19]. The software is capable of performing all steps of a test-beam analysis as described in the following (section 3.2). Each step of the reconstruction is performed by a dedicated module, which can be
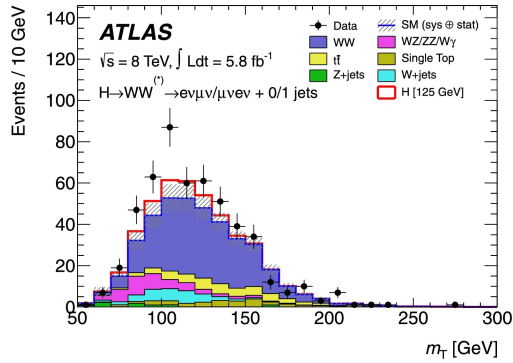
Figure 13: Example plot from the 'AT-LAS Higgs discovery paper' [5] created with *ROOT*.
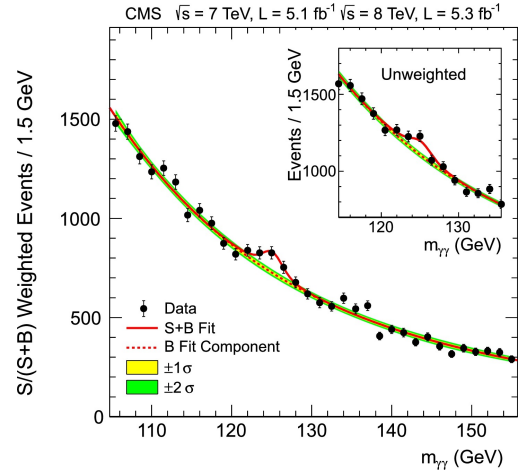
Figure 14: Example plot from the 'CMS Higgs discovery paper' [6] created with *ROOT*.

configured by the user. An illustration of the reconstruction chain for this lab course with the involved modules is shown in Figure 15.

The individuals steps of the analysis are described in the following. You will learn how to use *Corryvreckan* and how to configure the necessary modules in Section 4.

3.2 The Test-beam Reconstruction Chain

This section is a short introduction to the fundamentals of test-beam data reconstruction and pixel sensor characterisation. It covers the steps illustrated in the flow chart in Figure 15. As you follow the instructions in Section 4, you will recognize all the steps discussed here and understand them in more detail.
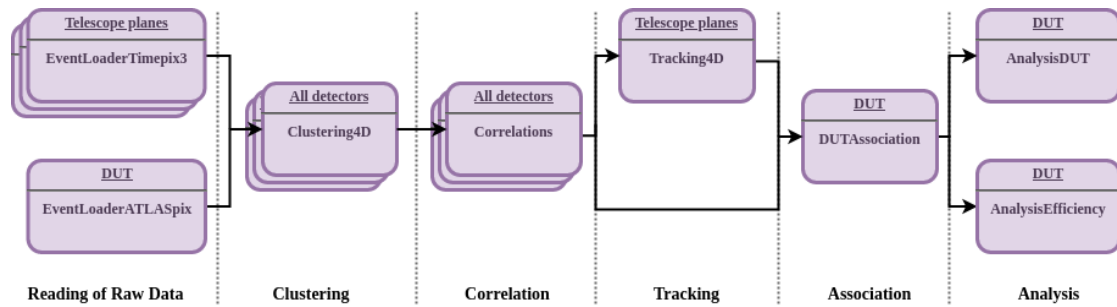
Figure 15: Flow chart of the *Corryvreckan* reconstruction chain for this lab course.

3.2.1 Reading in of Raw Data

The data acquired during a test-beam campaign are usually not stored in tables but in binary files, i.e. in a format which is not human-readable. Therefore, the first step of the reconstructions is reading the raw data from file and interpreting chunks of the data stream as the pixel hit information. In *Corryvreckan*, this is done in the modules called `EventLoader<SomeDetector>`.

The pixel hit information for the *Timepix3* and the *ATLASpix_Simple* comprises:

- pixel address (column and row)

- pixel timestamp: time-of-arrival (ToA)

- pixel charge: time-over-threshold (ToT)

3.2.2 Clustering

Depending on the incident point and angle of a traversing particle, it may create electron-hole pairs in two or more adjacent pixels. In addition, charge can "leak" from one pixel into neighbouring cells by lateral diffusion. This effect is called **charge sharing**.

As a consequence, a clustering algorithm is needed to group such pixels into one cluster. Since both the *ATLASpix_Simple* and the *Timepix3* measure a pixel-by-pixel timestamp (ToA), the clustering can be performed using both spatial and time information. The corresponding module in *Corryvreckan* is called `Clustering4D`.

Charge sharing can be beneficial for the spatial resolution of a detector: If multiple pixels fire and the energy deposition is measured by means of the time-over-threshold (ToT), a ToT-weighted mean of the cluster positions can be calculated.

3.2.3 Tracking

The beam at SPS consists of pions with an energy of around $120\,\text{GeV}$. This energy is sufficiently high to assume that the particles traverse all planes of the telescope in a straight line and scattering effects in the material of the detector planes are negligible.

For the tracking, clusters within a certain spatial and time interval on the planes of the reference telescope are combined into a straight-line track. The DUT plane should be excluded from the tracking to avoid a bias of the analysis.

In order to make use of the spatial and time information provided by the *Timepix3*, the *Corryvreckan* module `Tracking4D` is used.

3.2.4 Alignment

The raw data only contain basic pixel hit information (column, row, timestamp, ToT) for all sensors. Consequently, the reconstruction software is not aware of where exactly the sensor planes are positioned. Therefore, a detector geometry file is needed to describe the position and rotation of all detectors.

In addition, the positions and rotations of each sensor can only be measured by hand to a precision of $\mathcal{O}(1\,\mathrm{mm})$ and they differ from the ideal position with perfect overlap of all sensors as illustrated in Figure 16. However, for a precise tracking, the geometry needs to be known with a higher precison because even a misalignment well below 1 mm corresponds to an offset of multiple pixels. Therefore, a software alignment needs to be performed to find the exact position of each plane. This way, the intercept of the track with all planes can be determined accurately.
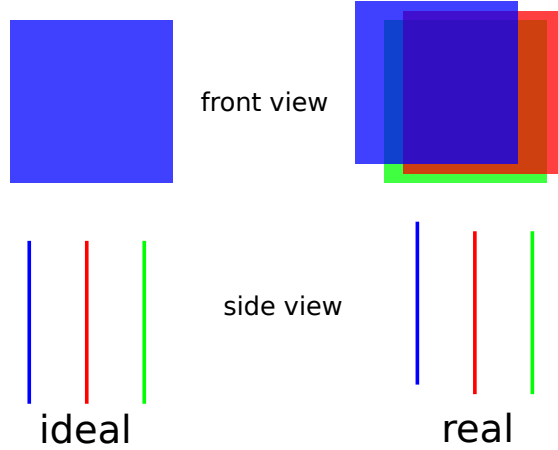
Figure 16: Illustration of the ideal and real (mis-)alignment of the sensor planes (exaggerated). Different telescope planes are indicated in different colours.

3.2.5 Correlations

Spatial correlations in the context of test-beam reconstruction can be defined on pixel or cluster level. In the latter case, the spatial correlations are calculated and histogrammed as:

$$x_{\text{correlation}} = x_{\text{cluster on \textbf{reference} detector}} - x_{\text{cluster on \textbf{this} detector}} \tag{3.1}$$

$$y_{\text{correlation}} = y_{\text{cluster on \textbf{reference} detector}} - y_{\text{cluster on \textbf{this} detector}} \tag{3.2}$$

for all clusters of a particular detector vs. all pixels on the reference detector. Any detector can be chosen as the reference.

Calculating correlations is a very powerful tool in particular in the early stage of the analysis or even during data taking as a first quality check of the data and a confirmation of its coherence. This is because correlations can be calculated from the raw data **without a working track reconstruction** – which is a bit more work as you will see below.

3.2.6 DUT Association

Since the DUT was excluded from the tracking (see section 3.2.3), its clusters need to be related ('associated') to the reference tracks.

To this end the *Corryvreckan* module `DUTAssociation` is used. It loops over all tracks and associates matching clusters on the DUT within a defined spatial and time interval to the track for further analysis as shown in Figure 17.
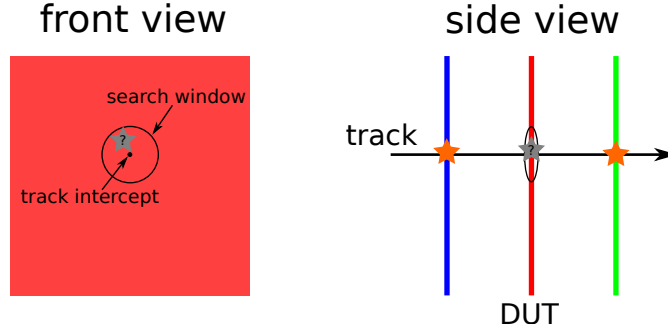
Figure 17: Illustration of the DUT association.

3.2.7 DUT Analysis

In the DUT analysis, properties of DUT clusters associated to a track in the reference telescope are studied. Spatial resolution, time resolution, and hit detection efficiency are of special interest.

3.2.8 Spatial and Time Resolution

In order to determine the spatial resolution of the DUT, the positional difference between the intercept of a track with the DUT and the associated cluster are calculated and histogrammed:

$$x_{\text{residual}} = x_{\text{track intercept}} - x_{\text{associated cluster}} \tag{3.3}$$

$$y_{\text{residual}} = y_{\text{track intercept}} - y_{\text{associated cluster}} \tag{3.4}$$

The root mean square (RMS) of this distribution gives an estimate on the spatial resolution of the DUT.

A simple calculation (see e.g. Appendix E.1 of [9]) shows, that for pixel sensors with little or no charge sharing (i.e. mostly clusters of size 1) the so-called 'binary resolution' is determined by

$$RMS_{\text{binary, x/y}} = \frac{\text{pixel pitch in x/y}}{\sqrt{12}} \tag{3.5}$$

Important: The measured resolution deduced from a histogram contains the combined resolutions of both the reference telescope and the DUT. Only if the telescope resolution

is much smaller than the DUT resolution can it be neglected in the calculation:

$$\sigma_{\text{measured}} = \sqrt{\sigma_{\text{telescope}}^2 + \sigma_{\text{DUT}}^2} \tag{3.6}$$

$$\sigma_{\text{measured}} \approx \sigma_{\text{DUT}} \tag{3.7}$$

In analogy to the spatial resolution, the time resolution can be determined from the width of the time residual between the reference track and the associated cluster on the DUT:

$$t_{\text{residual}} = t_{\text{track intercept}} - t_{\text{associated cluster}} \tag{3.8}$$

Both spatial and time residuals might have non-Gaussian shapes, for example tails. In the case of time residuals, these tails are typically not symmetric. As illustrated in Figure 18, a 'small' signal will cross the detection threshold later than a 'large' signal. This signal-height dependent delay is called **timewalk**, and can cause said non-symmetric tails. If a sensor does not only measure the arrival time of a pixel hit, but also the signal size (time-over-threshold), then the timewalk effect can be corrected for in the analysis. However, this exceeds the complexity of this course.

Both the spatial and the time resolution are determined using the *Corryvreckan* module `AnalysisDUT`.
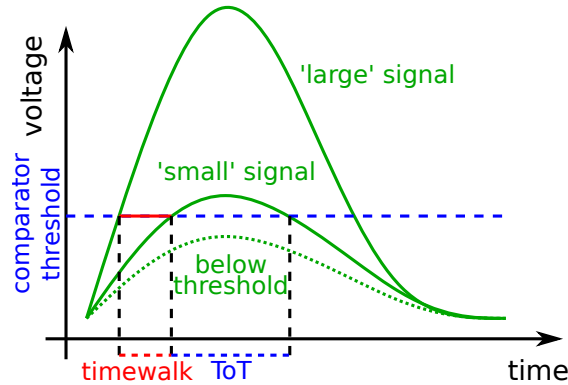
Figure 18: Illustration of the **time-over-threshold (ToT)** and the effect of **timewalk** for pulses of different heights.

3.2.9 Hit Detection Efficiency

The hit detection efficiency ϵ corresponds to the probability of detecting a particle passing through the sensor. It can be determined as follows:

$$\epsilon = \frac{\text{\# tracks } \textbf{with} \text{ an associated cluster}}{\text{\# tracks } \textbf{with} + \textbf{without} \text{ an associated cluster}} \tag{3.9}$$

If the spatial resolution of the reference telescope is smaller than the size of pixels on the DUT, not only the global efficiency of the sensor can be measured. In addition, the hit detection efficiency can be determined in dependence on where the track intercepts with the sensor within a pixel, i.e. close to the center, edge, or corner of the pixel. This is called the in-pixel efficiency.

The efficiency is determined with the dedicated *Corryvreckan* module `AnalysisEfficiency`.

4 Instructions

This section contains the instructions, which should be followed step-by-step during the lab course.

Important: All the steps you will perform here have been discussed in the previous section. You are encouraged to scroll back and read Section 3.2 again before getting started.

4.1 Getting Familiar with Corryvreckan

As described in section 3.1.2, *Corryvreckan* is a flexible and highly configurable software for the reconstruction and analysis of test-beam data. In the following, you will learn how to configure and use it step-by-step.

Corryvreckan needs to be provided with a configuration file as an input parameter to know what it is supposed to do. This is referred to as the **main configuration file**. An example can be found in the `configs` directory. Open `01_example.conf` with a text editor and inspect it.

As described in section 3.1.2, *Corryvreckan* is built in a modular way, such that each module performs a separate task such as reading in raw data, clustering or tracking. The main configuration file defines the reconstruction chain by listing all modules to be used and contains all relevant parameters. If a parameter is not specified, its internal default value is used. Each module is identified by a section header in square brackets "`[]`", followed by parameters and values, separated by an equal sign "`=`".

If you want to know more about a particular module and its functionality, all available modules and their parameters are described in detail in the user manual [4]

Each configuration file needs to start with a **global** section labelled `Corryvreckan` in which the **detector geometry file** and the name of the **output *ROOT* file** are defined (we'll look at both in a moment).

The global section of the configuration file is followed by a module called `Metronome`, which defines time slices of 20 µs (0 µs to 20 µs, 20 µs to 40 µs, ...). These are called **events**. The subsequent analysis will be performed event-by-event (time-slice by time-slice), i.e. the entire analysis chain (all listed modules in the main configuration file) will be executed sequentially for one event and then start again at the top. The reason for using the `Metronome` is that not the entire data set, which can easily be multiple gigabytes, can be loaded into the RAM simultaneously. In addition, it is not necessary to have pixel data with a timestamp of 15 min in the RAM, when processing the first few seconds of the data set or vice versa.

The following module `EventLoaderTimepix3` loads all data (pixel column/row, timestamp, time-over-threshold) of the detector named `Timepix3_0` matching the current
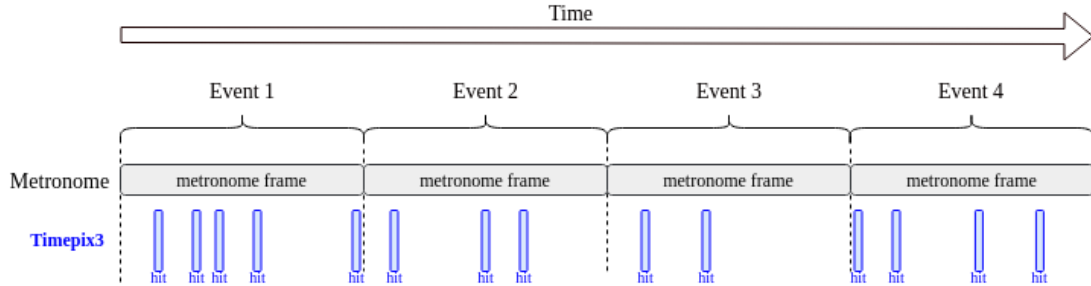
Figure 19: Pictoral representation of the event building mechanism. The `Metronome` defines the extent of the event and the `EventLoaderTimepix3` adds all pixels with a suitable timestamp to it.

event, i.e. suiting into the time-slice defined by the `Metronome` as depicted in Figure 19. After this, *Corryvreckan* moves on to the next event and repeats the procedure.

When the end-of-file of the data set is reached or the user interrupts the analysis, the **output *ROOT* file** is created and saved as specified in the global section of the configuration file. It can be found in the `output/` directory.

The input raw data files only contain pixel hit addresses (column/row), timestamps and ToT values. Hence, *Corryvreckan* does not know how the sensors are located relative to each other in the test-beam setup. Consequently, the above-mentioned **detector geometry file** is needed to specify the properties of all detectors as well as their position and orientation in space. It can be found in the `geometries/` directory. Open `01_example.geo` with a text editor and inspect it. In this file, each section describes one detector. The section header corresponds to the name of the detector, for which the number of pixels and the pixel pitch are provided. In addition, geometrical information such as the position and orientation of the sensor in space are specified.

Now let's run the example by typing:

```
$ corry -c 01_example.conf
```

Note: A warning will be shown that no material budget was specified for the detectors. The material budget can be deduced from the type of material and thickness of the sensor (e.g. 100 µm of silicon). This is not needed for the analysis of the data set at hand due to the high beam momentum at SPS. It is, however, relevant for a particle beam with a lower momentum such as at DESY (5.4 GeV at DESY vs. 120 GeV at SPS), where a more complex track model needs to be used to take the scattering of the particles in the material of the sensors into account. It is also displayed that no calibration file was provided. Calibration files are needed for a conversion of the ToT values into electrons for each detector. Also this is not relevant for the analysis within this lab course, so both messages can be safely ignored. A third type of warning might occur during the event loop and indicates power pulsing of the first telescope plane. This is unexpected and should be dealt with on the side of the developers. Since it affects only one event of the run, we can safely ignore also this warning.

In addition, you will see updated information about the ongoing analysis:

```
(STATUS) =======================| Event loop |=======================
(STATUS) Ev: 599.9k Px: 2.68M Tr: 0.0k (0/ev) t = 11.998s
```

It displays the number of processed events ("Ev"), the sum of pixel hits in all detectors ("Px"), the number of found tracks ("Tr"), the ratio of tracks per event, and the run time of the data taking. Since we haven't used the tracking module yet, the number of tracks remains zero for now.

Once *Corryvreckan* is finished, go to the `output/` directory and inspect the output file:

```
$ root -l 01_example.root
```

In the *ROOT* file, you will find the following histograms for `Timepix3_0` in `EventLoaderTimepix3`:

- `pixelToT`: This is a 1D histogram of the ToT values of all pixels.

- `hitMap`: This is a 2D histogram with the size of the pixel matrix. It is filled with the pixel address of each hit.

The `Metronome` does not produce any histograms as it performs only a trivial task.

Tip: The entire data set contains ~24 M events. You don't have to reconstruct the entire data set each time you change one parameter, especially while developing the analysis chain and testing your configuration files. To save time, you can run

```
$ corry -c 01_example.conf -o number_of_events=600000
```

to restrict the reconstruction to a defined number of events. Alternatively, you can press `Ctrl+C` while *Corryvreckan* is running. This way, *Corryvreckan* will finish analysing the current event, write out the *ROOT* file and then stop.

In the following, we'll build up the entire data analysis chain as described previously. For an overview of the steps, go back to Section 3.2 and have a look at Figure 15 and read the corresponding paragraphs again.

4.2 Reading in the Raw Data

Now that we know how to use *Corryvreckan*, we can start to read in the data of multiple detectors. To do this, we need to update both the main configuration file and the detector geometry file.

Open `geometries/02_read_data.geo` with a text editor. Using the geometrical information about the setup shown in Figure 20, complete the detectors geometry file. Since we do not yet know the precise (mis-)alignment in x- and y-direction, our "best bet" is to put all zeros.

Note: One telescope plane of your choice (e.g. the `Timepix3_3`) needs to be marked as `role = "reference"`. The *ATLASpix_Simple* needs to be marked as `role = "DUT"`. Since we don't know the spatial resolution of the *ATLASpix_Simple* yet, we can assume

a binary resolution both in x and y (see Equation 3.5). For its time resolution assume 16 ns.

Next, let's open the main configuration file `02_read_data.conf`. Update the path to the detectors geometry file and choose a reasonable name for the output *ROOT* file. The `Metronome` module is needed for the same reasons as before and the module `EventLoaderTimepix3` will now handle all detectors of type `Timepix3` specified in the geometry file. To load the data of the device-under-test, the *ATLASpix_Simple*, we need to add a new dedicated event loader for this detector type. This is the `EventLoaderATLASpix`, which requires the following configuration:

```
[EventLoaderATLASpix]
type = "ATLASpix"
clock_cycle = 8ns # This is a chip setting, used during data taking.
clkdivend2 = 15   # This is a chip setting, sed during data taking.
input_directory = <path/to/data>
```

This way, all *ATLASpix_Simple* pixel hits with the right timestamps will be added to the event as illustrated in Figure 21.

Once you've updated both the detectors geometry file and the main configuration file, re-run *Corryvreckan* and inspect the resulting output *ROOT* file. There should now be additional histograms for all sensors. If some histograms are empty, you've probably missed something in geometry or main configuration file.

Reminder: Use the option `-o number_of_events=600000` to save time for all further steps unless otherwise stated.

Inspect the following histograms:

- `hitMap` for all sensors: 2D histogram with the size of the pixel matrix, filled with all pixel hits per detector

Write a macro to draw the following plot:

- Divided canvas with the `hitMap` of all *Timepix3* planes and the *ATLASpix_Simple*.
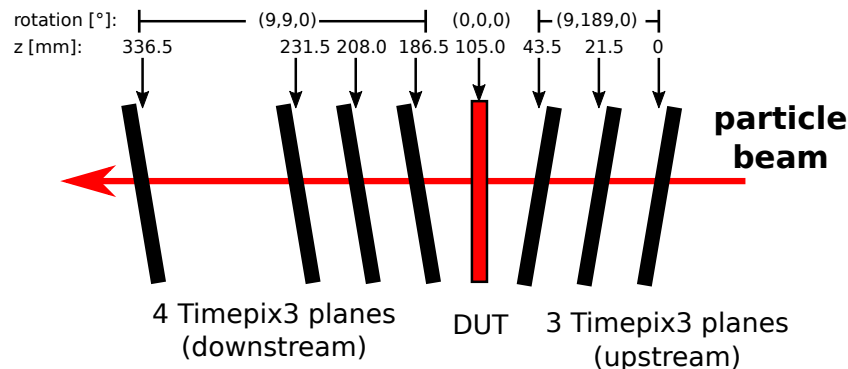

Figure 20: Schematic drawing of the detector setup with geometrical information on the plane distancing and rotations.
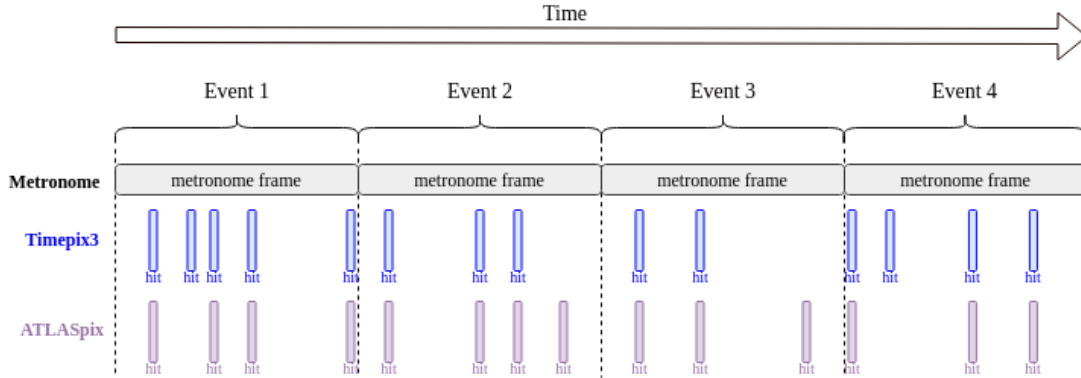
Figure 21: Pictoral representation of the event building mechanism. The `Metronome` defines the extent of the event and the `EventLoaderTimepix3` and `EventLoaderATLASpix` add all pixels with a suitable timestamp to it.

Answer the following questions:

- Do you see any pattern? Can you explain it?

- Does any of the telescope planes look significantly different from the others? What might cause it? **Tip:** Compare the z-ranges of the different hitmaps and use the zoom on the z-axis.

4.3 Clustering

Until now, we have only read in individual pixel hits. To combine these into clusters, we use the module `Clustering4D`, which takes both spatial and time information into account. Add it to your main configuration file `03_clustering.conf` and again, find a reasonable name for the output *ROOT* file. By default, the `Clustering4D` will only search for touching neighbours. Split cluster, i.e. clusters with a gap, as depicted in Figure 22 are not allowed. The ToT-weighted centre-of-gravity is calculated as the cluster position. Note that the ToT values are used instead of the charge because no calibration was performed. The cluster timestamp corresponds to the earliest pixel timestamp within a cluster. For the time cut, it is recommended to start with a large cut of `time_cut_abs = 200ns`.
Re-run *Corryvreckan* and inspect the histograms in the output file. The relevant histograms are:

- `clusterSize`: Number of pixels in one cluster.

- `clusterCharge`: ToT of the cluster (sum of all pixel ToTs). **Note:** *Corryvreckan* expects to receive calibrated charge measurements from the event loader. Hence, the x-axis label is wrong.
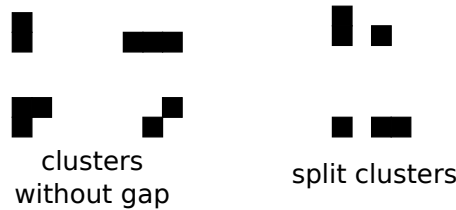
Figure 22: Examples of different clusters with and without a gap.

- `clusterSeedCharge`: ToT of the seed pixel of each cluster.
  Reminder: The seed pixel is the pixel with the largest charge within the cluster.

- `clusterWidthRow` and `clusterWidthColumn`: Extent of the cluster in row or column direction.

Write macros to draw the following plots:

- Overlay the `clusterSize` distribution for the *ATLASpix_Simple* and the *Timepix3* (a plane of your choice). **Tip:** Use the draw option "same" to draw them on top of each other for a better comparison.

- Overlay the `clusterWidthColumn` and `clusterWidthRow` distributions for the *ATLASpix_Simple*.

- Overlay the `clusterSeedTot` and `clusterTot` for both the *ATLASpix_Simple* and the *Timepix3*.

Don't forget to label the axes properly and add a legend.
Answer the following questions:

- Compare the cluster sizes for the *Timepix3* and the *ATLASpix_Simple*. How do they differ? Why?

- Compare the cluster width in column and row for the *ATLASpix_Simple*. How do they differ? Why?

- Compare the seed pixel ToT and the cluster ToT for the *ATLASpix_Simple* (for the *Timepix3*). Which difference can you observe for the *ATLASpix_Simple* and the *Timepix3*? Why?

4.4 Correlations

So far, the alignment specified in the detector geometry file was only measured "by hand". It is (most likely) not good enough to find tracks because even a misalignment below 1 mm already corresponds to an offset of many pixels. To have an initial estimation on how good the alignment is, we can investigate the spatial correlations. For this, add

the module `Correlations` to the main configuration file
`04_correlations_telescope.conf`. Again, use `time_cut_abs = 200ns`.
Re-run *Corryvreckan* and inspect the output file. The following histograms are the most important:

- `correlationX` and `correlationY`: These show the spatial correlations between each sensor and the reference sensor (`type = reference` in the geometry file) on cluster-level.

Peaks should be visible in the spatial correlation plots in both X and Y for all detectors. The peaks are superimposed with a "triangular" background, which comes from non-correlated clusters, either from different tracks or from noise. It is there because we correlate all clusters will all clusters within the entire event (20 µs).
Answer the following questions:

- How can translational misalignments be observed in the correlations? How large are they approximately in your case? How many pixel pitches do they correspond to?

- How can a bad rotational alignment be observed in the correlations?

You can test your answers by modifying the position and rotation in the detector geometry file by hand and checking the result. Make sure to rename the output file to avoid overwriting the one from the previous analysis.

4.5 Alignment and Tracking

As discussed above, the detector geometry file describes all sensor planes present in the setup as well as their position and rotation. The z-positions of all planes **must** be measured by hand in the existing setup and entered into the geometry file. They cannot be reconstructed a posteriori from the raw data. The x- and y-positions as well as the rotations can be measured only rough by hand. However, they have a strong influence on the tracking because even an offset well below 1 mm easily corresponds to a shift by multiple pixel pitches.
Consequently our measured alignment parameters need to be corrected to increase the track quality. Procedures for that are implemented in Corryvreckan, which iteratively shift and rotate the planes relative to the detector marked as role = reference to find these corrections.
At this point, it is important to recall the difference between **correlations** and **residuals**.

- A spatial **correlation** plot is filled with the spatial distance of any cluster on a particular detector and any cluster on the reference plane. No tracking is involved.

- On the contrary, a **spatial residual** plot shows the difference of the interpolated track intercept onto a particular plane and the position of the cluster on this plane.

Here, we need to distinguish between **biased** residual, in which case the cluster on the plane is part of the track and **unbiased** residuals, in which case the cluster has been associated to the track after the track fitting and is not part of the track itself.

$$x_{\text{correlation}} = x_{\text{cluster on reference detector}} - x_{\text{cluster on this detector}} \tag{4.1}$$

$$\text{biased: } x_{\text{residual}} = x_{\text{track intercept on this plane}} - x_{\text{track cluster on this plane}} \tag{4.2}$$

$$\text{unbiased: } x_{\text{residual}} = x_{\text{track intercept on this plane}} - x_{\text{associated cluster on this plane}} \tag{4.3}$$

Ultimately, the goal of the alignment is to force the **residuals** to be centered around zero. The **correlations** do not necessarily need to centered at zero as a possible offset reflects the *physical displacement* of a detector plane in x and y with respect to the reference plane. This is illustrated in Figure 23. For the assumed ideal alignment, which is what we start with, the distance between track intercept and hit on the blue plane is large. If initial alignment is too far away from the 'real' alignment, the track finding and fitting will not work at all. For the fine-tuned alignment, which is what we want to achieve with the alignment procedure, the residuals (i.e. the distances between track intercept and associated hits on each plane) are close to zero. On the contrary, the correlations between the blue and the red plane are 'off' by two rows representing the physical misalignment between the two planes.

However, it is useful to start by inspecting the **correlation** plots in the beginning when the assumed ideal alignment is too different from the real alignment and no or only few tracks can be found such that the residual plots cannot be filled reasonably. Based on these a coarse prealignment can be found, which is later improved in two more steps, telescope and DUT alignment.
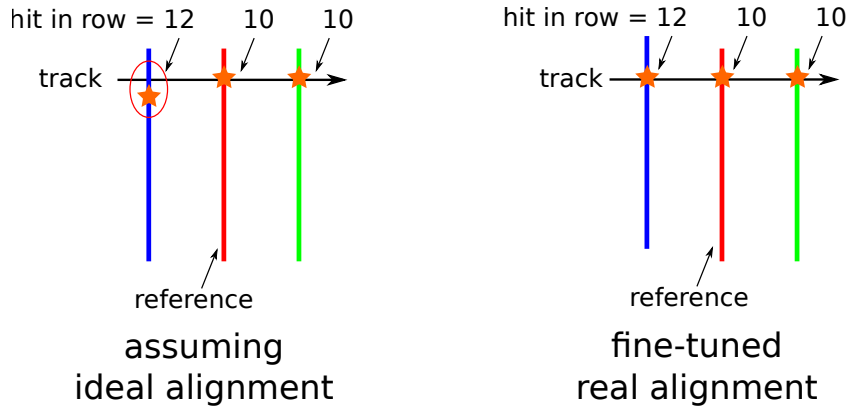

Figure 23: Illustration of a track fit through an assumed ideal and the fine-tuned real alignment of the sensor planes (exaggerated). Different telescope planes are indicated in different colours.

4.5.1 Prealignment

If the misalignment is too large, a reasonable track reconstruction is not yet possible. In this case, the "best bet" is to shift all spatial correlations to zero despite of what was discussed above.

In principle, this can be done by hand by inspecting all correlation plots and editing the detector geometry file. Alternatively, the `Prealignment` in *Corryvreckan* can automatically determine the peak position of the spatial correlations and update the detectors geometry file accordingly.

To use it, edit `05_prealignment.conf` and add

```
[Prealignment]
time_cut_abs = 200ns
method = gauss_fit
fit_range_rel = 10
```

With these settings a Gaussian is fitted to the peak to determine its position with a fit range of $10\times$ the spatial resolution of each detector as specified in the detector geometry file.

You should also add `detectors_file_updated = ...` in the global section of the main configuration file to avoid that the initial geometry file is overwritten.

The output *ROOT* file from this analysis does not show the effect of the updated geometry because the corrections are calculated at the very end of the analysis when the histograms are already filled. Nevertheless, you should have a look at correlation plots in the prealignment module to verify that the peaks were detected correctly and the Gaussian fits look reasonable, i.e. represent the peak position well. If the peaks have not been found, you may need to adjust the fit range or re-run the analysis with a larger number of events.

After running the prealignment, re-use the previous configuration `04_correlations.conf` with the updated detector geometry file (and changed output file name). Compare the correlation plots of all detectors you get with the updated geometry to the initial geometry and verify that the peaks have been shifted towards zero.

Write a macro to draw the following plots:

- Divided canvas with the correlations in x for all *Timepix3* planes before and after the prealignment to show that they are centered at zero now.

- Divided canvas with the correlations in y for all *Timepix3* planes before and after the prealignment to show that they are centered at zero now.

4.5.2 First Tracking

Now that the correlation peaks are close to zero, we can have a first go on the tracking. For this, edit `06_tracking.conf` and add the module `Tracking4D`. As for the previous modules, use a time cut of 200 ns. In addition, we should specify a spatial cut, which defines a "search ellipsoid" around the projected track on the next plane. Try to use

```
time_cut_abs = 200ns
spatial_cut_abs = 200um,200um
```

In addition, we should set

```
exclude_dut = true
```

to make sure that tracks are built with clusters from the reference telescope only and the device-under-test is excluded. The track timestamp is set to be the average timestamp of all clusters along the track.

Run *Corryvreckan* and inspect the output file. The most important histograms for the tracking module are:

- `LocalResidualsX` and `LocalResidualsY` for each detector: These are the biased residuals discussed above.

- `trackChi2ndof`: This is the track χ^2/ndof (number of degrees of freedom) for each track fit. It is calculated from the residuals of the track and the clusters on each plane used for the track, as well as the resolutions specified in the detector geometry file.

Note: there are global and local residuals. The global residuals are calculated in a Cartesian coordinate system given by the orientation of the detector marked as `role = reference`. Local residuals are calculated in the plane of the corresponding detector plane.

Answer the following questions:

- Compare the residuals with the correlations from the previous analysis. What is the difference?

- As you remember, a reasonable fit should have a χ^2/ndof ~ 1. What does the track χ^2/ndof look like for you? Why? **Tip:** Right-click on the statistics box and choose `SetOptStat`. Then add two additional 'ones' to obtain the overflow bin.

4.5.3 Telescope Alignment

With the coarse prealignment and a working track reconstruction (even though the track χ^2/ndof might be bad), we can proceed with the precise alignment – still excluding the DUT. For this, add the module `AlignmentTrackChi2` to the configuration file `07_telescope_alignment.conf`. This module iteratively varies the position and rotation of each sensor, which is used in the tracking, refits the tracks and tries to minimize the track χ^2. Initially, you can run it with all parameters on default.

Note: For a good results more statistics is needed than in the previous steps. However, it is still not necessary to analyse the entire data set. For instance, you can use the option `-o number_of_tracks=250000`.

As for the prealignment, the histograms in the output file show the initial alignment because the updated geometry file is only created at the very end of the analysis, when

all histograms are already filled. To test the new alignment, update the previous config-
uration `06_tracking.conf` with the new geometry file and re-run the analysis.

If the result is not satisfactory, you can re-run the alignment a second time (don't forget
to update `detectors_file` and `detectors_file_updated`). In addition, you can try to
allow only shifts or only rotations by using the following parameters:

```
[AlignmentTrackChi2]
align_position = true    # set to false to disable
align_orientation = true # set to false to disable
```

The alignment is reasonable when the residuals on all telescope planes are well centered
at zero (the mean of the histogram should be smaller than 1 µm) and symmetric. In
addition, the track χ^2/ndof distribution should be peaking around 1.

Write macros to draw the following plots:

- Overlay the χ^2/ndof distributions before and after the alignment.

- Divided canvas with the residuals in x for all *Timepix3* planes before and after the
  alignment.

- Divided canvas with the residuals in y for all *Timepix3* planes before and after the
  alignment.

Tip: Instead of `hist->Draw()` use `hist->DrawNormalized()` to renormalize the his-
tograms. Otherwise the histograms are hard to compare because the numbers of entries
are very different.

Explain what you see.

4.5.4 First DUT Residuals

Now that a good alignment of the reference telescope was achieved, we can go ahead
with the alignment of the device-under-test.

Since the DUT is excluded from the tracking, the clusters on the DUT are not part of
a track. Instead, we need to associate the clusters on the DUT to tracks in a different
way. To do so, we use the module `DUTAssociation`. For each track, it calculates the
track intercept with the DUT and searches for clusters within a defined spatial and time
window, which can be associated with this track. Since the DUT may not yet be well
aligned, use generous cuts like

```
time_cut_abs = 200ns
spatial_cut_abs = 350um, 350um
```

In addition, the module `AnalysisDUT` can be used to calculate the DUT residuals.

Add both modules to `08_dut_residuals.conf` and run *Corryvreckan*. In the output
file, look at the following histograms:

- `residualsX` and `residualsY`: These are the unbiased residuals as discussed above.
  It is again suggested to look at the local residuals.

Answer the following question:

- Why should the DUT be excluded from tracking? For instance, think about the effect on the hit detection efficiency of the DUT.

4.5.5 DUT Alignment

In the prealignment step, the correlations have already been shifted toward zero for the DUT. Consequently, we can proceed with the precise alignment of the DUT. Since it is not part of the tracking, we need to apply a different method than the track χ^2 minimization here. The module `AlignmentDUTResidual` is used, which iteratively varies the position and rotation of the DUT to center the unbiased residuals in x and y around zero and minimize their widths.

To use it, add this module to `09_alignment_dut.conf`. As before, you should also add

```
detectors_file_updated = ...
```

in the global section of the main configuration file to avoid that the initial geometry file is overwritten.

For a good alignment result, it is recommended to use a large number of tracks. As for the telescope alignment described above, the new alignment will be calculated and written out at the very end of the analysis, so it will not yet be seen in the histograms of the output file. To see how the alignment improved, update the previous configuration `08_dut_residuals.conf` with the new geometry file, rename the output *ROOT* file and re-run *Corryvreckan*. The mean of the residuals should be below 1 µm and the residuals should be symmetric.

If the result is not yet satisfactory, try to run the alignment a second time. Again, don't forget to update the configuration file to use the correct geometry.

Write a macro to draw the following plots:

- Overlay the residuals in x/y for the *ATLASpix_Simple* comparing the initial and the final alignment.

Explain what you see.

Note: If you're running out of time or have trouble at this step, ask your supervisor for help. He/she might provide you with the finished alignment such that you can continue with the other tasks.

4.6 DUT Analysis

With a fully aligned setup, we can proceed to the actual DUT analysis, in which we can characterize the sensor and investigate its performance. Update `10_dut_analysis.conf` with the most recent geometry file. We now have the full data analysis chain in place as described in Secion 3 and illustrated in Figure 15 (apart from the efficiency analysis, but we'll get to that later).

For reliable analysis results, it is important to have a high number of tracks for good statistics. Again, you can test your configuration file with a short analysis, but for the final results, the entire data set should be analysed for the best possible statistics.

Tip: To save time, have a quick look at Section 4.6.3 and add the corresponding module already now.

Note: If you are running on a machine with limited memory resources, restrict your analysis to 10 million events by using the option `-o number_of_events=10000000`. Otherwise, the virtual machine or your PC may crash.

After running *Corryvreckan*, inspect the output file and compare the following histograms:

- `Correlations/ATLASpix_0/hitmap_clusters`: 2D histograms of all clusters on the DUT

- `AnalysisDUT/ATLASpix_0/clusterMapAssoc`: 2D histogram of all clusters on the DUT that have been associated to a track

Write a macro to compare the above-mentioned histograms. Answer the following question:

- In one case, not the entire matrix is filled. Can you explain why? Think about the geometrical dimensions of the *Timepix3* sensors and the *ATLASpix_Simple*.

4.6.1 Spatial Resolution

Next, we can have a closer look at the spatial resolution. As described in Section 3, the width (RMS) of the spatial residuals can be used to estimate the spatial resolution. Look at the following histograms:

- `residualsX` and `residualsY` in `AnalysisDUT` (again local)

Write a macro to draw the following plot:

- Overlay spatial residuals in x and y for the *ATLASpix_Simple*.

Answer the following questions:

- Are these biased or unbiased residuals? Why?

- Can you quantify the spatial resolutions?

- Are the spatial resolutions what you expected? Why?
  Tip: Look at the cluster size distribution for the *ATLASpix_Simple*.

- Why are the spatial residuals non-gaussian?
  Tip: Compare the pixel size of the *ATLASpix_Simple* with the precision of the telescope.

4.6.2 Time Resolution

Similarly, the time resolution can be inferred from the track-cluster time residual. Find the following histograms:

- `residualsTime`: This is the time residual between the reference tracks and the associated clusters on the DUT.

- `residualsTimeVsTot`: This is the same time residual between the reference tracks and the associated clusters on the DUT as in the previous histogram plotted against the seed pixel ToT of the associated cluster.

Write a macro to draw the following plots:

- Draw the time residual for the *ATLASpix_Simple* and fit a Gaussian to the distribution. Make sure to choose an appropriate axis range.

- Draw the `residualsTimeVsTot`.

Answer the following questions:

- Is the peak position (offset from zero) relevant for the time resolution? What may cause it?

- Describe the shape of the distribution and compare it to the 2D histogram. Why is there a strong non-gaussian tail? Does it make sense on which side it is?

- Can you quantify the time resolution? Compare the Gaussian fit with the RMS of the distribution. Which one is more appropriate?

4.6.3 Hit Detection Efficiency

For efficiency measurements, a dedicated module called `AnalysisEfficiency` exists in *Corryvreckan*. It can be placed behind the other analysis module `AnalysisDUT` in the main configuration file. Add it to `10_dut_analysis.conf` and re-run *Corryvreckan*.
The hit detection efficiency can be read off from the terminal or the *ROOT* file. In addition, look at the following histograms:

- `chipEfficiencyMap_trackPos`: 2D map of hit detection efficiency across the whole matrix of the sensor.

- `pixelEfficiencyMap_trackPos`: 2D map of the efficiency distribution inside a pixel cell. All pixels are superimposed.

Write macros to draw the two histograms mentioned above.
Answer the following questions:

- How large is the hit detection efficiency?

- Why is the chip efficiency map not filled entirely?

- Do you recognize any pattern in the pixel efficiency map? Can you explain why?

Optional: Repeat the efficiency analysis multiple times while systematically scanning the spatial cuts in `DUTAssociation`. Plot the measured efficiency versus the association cut. What do you observe? Can you explain why?

Bibliography

1. The ROOT Authors. *ROOT Data Analysis Framework* [Online; Last accessed August 21, 2023]. https://root.cern.ch/.

2. Brun, R. & Rademakers, F. ROOT — An object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **389.** New Computing Techniques in Physics Research V, 81–86. ISSN: 0168-9002. http://www.sciencedirect.com/science/article/pii/S016890029700048X (1997).

3. The Corryvreckan Authors. *Corryvreckan Project Website* [Online; Last accessed August 21, 2023]. https://cern.ch/corryvreckan.

4. Kröger, J., Spannagel, S. & Williams, M. User Manual for the Corryvreckan Test Beam Data Reconstruction Framework, Version 1.0. *2019.* arXiv: 1912.00856 [physics.ins-det].

5. Aad, G. *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* **716,** 1–29. ISSN: 0370-2693. http://www.sciencedirect.com/science/article/pii/S037026931200857X (2012).

6. Chatrchyan, S. *et al.* Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B* **716,** 30–61. ISSN: 0370-2693. http://www.sciencedirect.com/science/article/pii/S0370269312008581 (2012).

7. *The Mu3e Experiment* [Online; Last accessed August 21, 2023]. https://www.physi.uni-heidelberg.de/Forschung/he/mu3e/.

8. Gehrke, T., Amato, C., Berke, S. & Martisikova, M. Theoretical and experimental comparison of proton and helium-beam radiography using silicon pixel detectors. *Physics in Medicine and Biology* **63** (Jan. 2018).

9. Kolanoski, H. & Wermes, N. *Teilchendetektoren – Grundlagen und Anwendungen* (Springer, 2016).

10. Kolanoski, H. & Wermes, N. *Particle Detectors, fundamentals and applications* ISBN: 9780198858362 (Oxford University Press, Sept. 2020).

11. Rossi, L., Fischer, P., Rohe, T. & Wermes, N. *Pixel Detectors: From Fundamentals to Applications* (Springer, 2006).

12. *CERN Test Beam Facility Database* [Online; Last accessed August 21, 2023]. https://www.cern.ch/tbdb.

13. *CERN: The Super Proton Synchrotron* [Online; Last accessed August 21, 2023]. https://home.cern/science/accelerators/super-proton-synchrotron.

14. *MapCERN Mobile App* [Online; Last accessed August 21, 2023]. https://play.google.com/store/apps/details?id=ch.cern.mapcern.

15. *Detector technologies for CLIC* (eds Dannheim, D., Krüger, K., Levy, A., *et al.*) https://cds.cern.ch/record/2673779 (CERN, 2019).

16. Poikela, T. *et al.* Timepix3: a 65K channel hybrid pixel readout chip with simultaneous ToA/ToT and sparse readout. *JINST* **9,** C05013 (2014).

17. Prathapan, M. *et al.* Design of a HVCMOS pixel sensor ASIC with on-chip readout electronics for ATLAS ITk Upgrade. *PoS* **TWEPP2018,** 074 (2019).

18. *Wikipedia: ROOT* [Online; Last accessed August 21, 2023]. https://en.wikipedia.org/wiki/ROOT.

19. *CLIC Detector & Physics Study* [Online; Last accessed August 21, 2023]. https://clicdp.web.cern.ch/.