Commissioning of the CMS-CHROMIE beam telescope - DUT Rotation Control and Detector Resolution Estimation

Peter Wimberger

CERN Summer Student programme 2018

Abstract

A beam telescope is a particle detector with multiple module planes, which is primarily used to test new detectors with unknown properties ('DUT' = device under test) in a particle beam.

As my project in the CERN Summer Student programme 2018 I assisted in commissioning of the CMS-CHROMIE telescope, developing a software solution to control the rotational stage for the DUT and estimate the detector resolution by calculated cross-residuals and track uncertainties.

In this report I will explain the parts of CHROMIE, describe the necessary steps to develop the rotation controller and outline the procedures used for sensor resolution estimation.

Furthermore I will give a summary of my other tasks during this wonderful summer at CERN.

Keywords: CERN, Summer Student, report, CMS, CHROMIE, telescope, rotational stage

1. CERN Summer Student programme

The CERN Summer Student programme allows about 300 students each summer from the whole world to learn more about particle and especially high energy physics and conduct research directly at CERN. I had the chance to attend many lectures on particle physics, visit multiple research sites, take part in workshops, but mainly worked in the team of Stefano Mersi and Nikkie Deelen on the CMS-CHROMIE telescope.

2. CMS-CHROMIE telescope

To accommodate the increased luminosity, pileup and resulting radiation of the high luminosity-LHC (HL-LHC), in operation after long shutdown 3 (LS3) in 2024/25, the CMS experiment needs to replace its tracker detector. [1] The detector modules for this Phase-2 upgrade of the inner tracker need to be tested to ensure production quality and understanding of their properties. This is done also inside so-called beam telescopes, which consist of a full detector environment in which the detector in question is placed and tested. To deal with the special requirements of the next CMS upgrade and scientific research, a new telescope was developed, called CHROMIE (CMS High Rate telescOpe MachInE, figure 2). [2]

2.1. Principal telescope commissioning

The principal procedure of installing and commissioning a beam telescope is the following. After the initial design phase, the telescope's detectors with already known properties and optimal calibration parameters are mounted on or inside the telescope. The detector modules are connected to read-out electronics and a trigger system is implemented. The read-out signals are connected to the data acquisition (DAQ) hard- and software. Once data can be obtained digitally, it can be analyzed. First and foremost the data is analyzed to ensure functionality and integrity of the telescope. This is done i.e. by checking data completeness and the hit coordinate correlations of different detectors.

To transform the raw data (hits in 2D) to a 3D particle track the global detector module position has to be known. Determining the accurate position of the modules is crucial for the optimal telescope resolution. First, the design geometry is prepared for the data analysis framework. As the actual geometry is reconstructed in data analysis after data-taking. This alignment is achieved by using separate data taking runs to construct tracks and minimize the residuals over all valid particle tracks by shifting the detectors in x, y, z and rotating them around all 3 axes. The aligned geometry now accurately represents the reality and is used for all further analysis.

After having full knowledge about the telescope, its geometry and properties, a DUT (= device under test) can be added in the middle or behind the other detectors. This DUT is then connected to the telescope's read-out and DAQ systems or to its own, as required. Finally the data of the telescope and the DUT are compared: precise particle tracks are obtained by linear

fitting the 3D data of the telescope. These tracks are then intersected with the DUT module plane. Comparing these intersection points and the DUT data hits yields residuals to align the DUT. After alignment these residuals can be recalculated and used to statistically estimate the DUTs resolution. Analyzing the detector resolution with respect to the chosen parameter values (noise threshold, high voltage, rotation angle,...) forms understanding of the detectors behaviour and gives access to optimal parameter choices. Degradation effects caused by radiation damages can also be analyzed by comparing new modules to irradiated ones.

2.2. CHROMIE in detail

CHROMIE was built to enable detector tests inside an environment similar to the extreme high particle rate expected in the CMS Outer Tracker in the upcoming high luminosity LHC (up to $50 MHz/cm^2$). To handle these rates, fast and radiation hard detectors are needed. CHROMIE therefore uses 8 CMS Phase-1 Barrel pixel modules (same as currently used inside CMS) embedded in 4 metal frames on each "arm" (figure 1). It also uses the same custom-made triggering, control and readout boards and data acquisition software as the actual CMS pixel detector at LHC point 5.

CHROMIE is currently set up in the CERN North Area beam line H8, which supplies experiments with particles obtaining their energy from the SPS. The beam's particle type does not matter for our requirements, as long as it ionizes particles inside the detector sensors. The beam line's main user can choose the main particle type by inserting different targets into the beam (protons for no target and pions and muons for targets of different material and length).

2.3. Hardware

CHROMIE consists of the following hardware parts: Two overlapping scintillators on each end are attached to photomultipliers, which produce a signal to trigger on. This signal is sent to the NIM (Nuclear Instrumentation Module) Logic, which will be superseded by a digitally programmable Trigger Logic Unit in the near future. The Logic shapes, discriminates and coincides the signals from the scintillators with themselves and a clock signal and sends a trigger to the AMC13 board. The AMC13 (Advanced Mezzanine Card for Slot 13) is a custom-made board for CMS. It distributes the trigger signal from the Logic and the LHC clock ($25 ns$) to the FEC (Front End

Controller) and FED (Front End Driver), which control the detector modules and readout its values. FEC, FED, AMC13 and MCH (communication interface) cards are inside and connected to a μTCA crate. FEC and FEC are connected via optical fibres to the FEROL (Front-End Readout Optical Link), a PCI module inside the computer.

Auxiliary electronics, like the motherboard, voltage and temperature probes are connected via i2c to the computers.

2.4. Software: DAQ and Analysis

As data acquisition software, CHROMIE uses the same framework as the full CMS detector: XDAQ (pronounced as cross-duck). XDAQ is part of the CMS Software (CMSSW). The CHROMIE team uses a specific branch called POS (Pixel Online Software). XDAQ allows the development and usage of distributed, hardware controlling applications called Supervisors. XDAQ Supervisors can be run distributed and independently on different machines and controlled by a main Web-Interface or other Supervisors via SOAP messages. They control every important part in the experiment set-up, like configuration and running of FEDs, FECs, AMC13, Logic and more. (see 2.3). The method of distributed module management is crucial for reliable and remotely accessible monitoring and data-taking.

In our setup every hardware component has its own associated supervisor, which interacts with the hardware. This includes the Front End Drivers, Front End Controllers, AMC13 etc. and my developed Rotational Stage Controller called RotaContSupervisor (see 3.2). Most of these supervisors can be configured and controlled by a 'main' Supervisor called 'PixelSupervisor' with a rich Web-Interface available.

To analyze data we use data quality management (DQM) scripts and the framework provided by CMSSW.

3. Development of a control interface for the rotational stage

Since the particles in a test beam fly all in the same direction, in order to test the response of the Device Under Test (DUT) to different particle incident angles α, it is necessary to rotate the DUT by the desired angle α and repeat the measurement for each desired value of α. The need for a rotational stage inside a telescope arises, if one wants to remotely rotate the DUT located in the beam line precisely and reliably. It is most convenient, if this rotation procedure can be automated and is incorporated inside the

data acquisition software. This method reduces potential human error and allows to add the logging of the rotation angle to an event data record as meta data in an automated way.

3.1. RotaContoller - The Mid-Level library

I was given a Physik Instrumente (PI) Mercury C-863 DC Motor Controller and rotational stage (PI M-062.DG) with an angle resolution of 1.2 μrad ($6.9 \times 10^{-5°}$). PI supplies the controller with a Windows GUI (MikroMove), but as CHROMIE uses XDAQ, which requires Linux as Operating System, as data acquisition framework, a new approach had to be found. So the first step was to understand the bundled (but officially unsupported) Linux installation of the hardware driver. This proved to be quite complex. The installation script was written in the bash scripting language. After figuring out why it did not function out-of-the-box (file permissions and imperfect udev-rules), I could proceed.

The controller installation script installs a GUI ("pi_terminal"), LabView drivers, a common stage parameter database and a C++ library (libpi_pi_gcs2.so). A simple library usage example is also supplied, but did not succeed at meeting our demands. Therefore I developed a stand-alone C++ library (RotaController.so) to wrap the low-level hardware commands of the PI library inside meaningful functions to be able to call them in a more reliable fashion and make them more maintainable. The project repository [3] also includes a RotaControllerTest.cc command line interface, which can be used out-of-the-box or adapted to specific needs.

I hope to be able to make the RotaController project open-source as I came to understand, while researching already existing solutions, that there is a need for an adaptable Linux C++ library for PI controllers from other research institutes, e.g. DESY in Hamburg, Germany for their telescopes and similar applications.

3.2. RotaContSupervisor - The XDAQ Supervisor

Once RotaContoller worked reliably, I could start to write a module for XDAQ (see 2.4), a so-called Supervisor. XDAQ Supervisors can be run distributed and independently on different machines and controlled by a main Web-Interface or other Supervisors via SOAP messages.

The Supervisor I developed is named RotaContSupervisor. [4] It handles calling the mid-level RotaController library and is accessible via a smiliar Web-Interface as the Supervisors for the FEC, FED and AMC13.

XDAQ Supervisors use an internal Finite State Machine (FSM) to express the current state of the hardware it controls. A FSM is a concept from computer science. A FSM has different predefined states and transitions from one state to another. No new states or transitions can be added once the machine started. It always starts in the same given Init-State (‘Disconnected‘ for us). If the machine now receives a state transition request, it will move to a new state immediately depending on the name of the transition. The FSM is therefore, at any point in time, in a predefined state. In its most simple and most common form (deterministic FSM) there exists only one target state for one transition. As FSM state transitions happen instantaneous, a good practice to model hardware as a FSM is to define states, that represent changes in the hardware that take some time (connecting, configuring etc.).

In the RotaContSupervisor's FSM the only valid state transition from ‘Disconnected‘ is ‘connecting‘ leading to the state called ‘Connecting‘. If the lower level libraries now report a successful connection, the machine will transition via ‘ConnectingDone‘ to ‘Connected,Unconfigured‘. If there is an error, the machine will instead transition via ‘ConnectingFailed‘ back to ‘Disconnected‘.

The RotaContSupervisor also provides a Web-Interface inside XDAQ. This interface allows a user to call the necessary functions: ‘Connect‘, ‘Configure‘, ‘Reference‘ and finally ‘Move‘. ‘Connect‘ establishes the connection from the computer to the controller and makes it usable. ‘Configure‘ initializes the axis to enable the controller to talk to the connected axis and loads custom parameters like velocity, acceleration, deceleration, target on-set time and reference point offset. ‘Reference‘ moves the stage to its reference point to determine its absolute position. ‘Move‘ moves the stage to the specified target position.

To allow movement to absolute target position, the stage has to do a reference movement to the reference point (or maybe reference limits for linear stages) after every new connection. The reference point inside our rotational stage is a magnet switch glued to the blue ring at $-1.4°$ (would ideally be a $0°$, but this offset is within limits of production constraints). This ring can be rotated after loosening 3 screws with a 0.9mm Allen key to allow arbitrary global reference positions.

The RotaContSupervisor can also pause the Main Supervisor (PixelSupervisor) to make sure no data is recorded during moving, which would mean an unknown geometry state in the data. This is done via SOAP messages, an XML-formatted network message. In contrary, the PixelSupervisor can

also send SOAP messages like 'RotaContSetUp' (which combines 'Connect', 'Configure' and 'Reference') and 'Move'. In general, Supervisors can send SOAP messages to all other or specific Supervisors in the XDAQ network and act independently according to the message received.

While adding the RotaContSupervisor to the default experiment XDAQ configuration and trying to 'Connect', we encountered an error (segmentation fault). From the stack trace, I learned that it is an error caused by calling 'connect' in RotaController, which itself calls connect in the low-level PI library. This function opens a new thread to connect to the controller itself. In fact, it opens a boost::thread, which cannot exit correctly and causes the segfault. This was unexpected as everything functioned correctly, when RotaContSupervisor was used alone in XDAQ without the other Pixel libraries. After trying to debug this, which proved to be difficult, because, as this library is precompiled, I have no legitimate way to edit the source code. After some time I finally figured out, no segmentation fault happens, if one reorders the other XDAQ (Pixel)Supervisors in the configuration XML file, so they are only imported/included after the RotaContSupervisor. Therefore this problem is caused by the more complex XDAQ Supervisors overwriting the thread class symbol, which the PI low-level library accesses with the boost::thread class.

While developing, I learned many small specifics about C++ and Linux tools and errors, especially C++ compiling and library linking. I also finally learned to use git efficiently and understood its real purpose.

4. Module resolution estimation and track fitting

4.1. Error estimation with residual analysis

In order to fully understand the properties of the telescope's detector modules, one has to determine their resolution. This is most commonly done by calculating the residuals of the assumed particle track and detector hit coordinates. In other words, the most likely particle track is reconstructed or assumed and used to calculate deviations to the data hits. To achieve the maximal telescope resolution the digital 3D geometry has to be aligned to best describe the real geometry. This process involves many steps and was beyond the scope of my summer project. Therefore my task was to estimate the detector resolution and only apply rudimentary alignment.

This analysis was done in two steps:

- Rough estimation by correlating hits in different modules per event
- Building tracks and calculating the residual distribution uncertainty and fitting error

Based on my calculations the CHROMIE team learned that the nominal geometry and the reality agree very well even without software alignment, resulting in a high pre-alignment detector resolution.

For all further calculation, we assume that the particles travel along the -z axis (as seen in figure 5). Modules are referenced by their Plane and Side (see 4.2). A 3D visualization of a typical event can be seen in figure 6.

4.2. Rough estimation by correlating hits in different modules per event

To obtain a good first impression of the order of magnitude to expect of the detector resolution, I computed the correlation of each module's data points in 3D. The data I was given to analyze was pre-processed in CMSSW by combining hits into clusters and transforming these clusters from the detector plane coordinates to 3D using the nominal geometry, while retaining the information of the detector module. As figure 5 shows, two modules are mounted inside one metal frame inside the telescope, discriminated by their Side (sign of the x coordinate) and together describing a so-called Plane (numerical order along the z-axis). They are treated completely independent aside from this mounting detail. In other words, modules are referenced by its Plane and Side in this analysis step by compacting the x and z coordinate information to Side and Plane, respectively. The given data set consisted of one entry per cluster hit with the associated eventID, x, y, z, Plane (1..4 and $-1..-4$), Side (-1 or 1), cId (cluster ID; incremented for each cluster in this event) and modId (incremented for each cluster in this event and module).

The following analysis algorithm is fairly simple, but allows quick access to first qualitative information we seek.

The x- and y-coordinates are treated independently, which means the calculation is the same. As always in physics, individual events are also treated independently.

We assume that particles travel exactly along the z-axis or, in other words, that the deviation of the z-axis and resulting error is small.

For each event, the algorithm now generates each possible combination of cluster IDs (AB,AC,AD,BA,BC,...). For each of these combinations the residual is formed (subtract coordinate of the first cluster minus the second)

and the obtained value is added to a distribution for this specific module combination coordinate(Plane1, Side1)-coordinate(Plane2, Side2). After analyzing all events, the resulting distribution should be describable by noise (actually two convoluted constant noise distribution, which form a near-flat triangle function) and a gaussian.

This gaussian arises from the real particle tracks and can be seen in the graph (figure 7). Its standard deviation can be characterized by the following contributions:

- the misalignment of the real geometry compared to the nominal geometry used for this data (see section 3),
- the intrinsic detector resolution (dominated by the pixel density, calibration parameters and analysis technique),
- the deviation of the particle tracks along the z axis.

Its mean describes the offset of these two modules to another, but was manually aligned for the plot shown in figure 7. The offset of the modules in comparison to the first module of each side is in the order of a few hundred micrometers.

The standard deviation of this gaussian is the first estimation of detector resolution we seek. Of course this estimation is a convolution of the resolutions of both analyzed modules, but gives a good estimate of the order of magnitude and, considering all module combination distribution, misbehaving modules can be identified.

As figure 7 exemplary shows the detector resolution for all detector combinations is in the order of 10 to 100μm.

4.3. Building tracks and calculating the residual distribution uncertainty and fitting error

To improve the quality and reproducibility of the first detector resolution estimation (4.2), additionally I took a second approach by using residuals obtained by linear fits and letting them iteratively converge. In detail, consider the following steps for each event and one Side: Select one hit in each of the 4 detector Planes, which could correspond to a real particle track (is within a certain window in regards to the ones in the other Planes). Then, for each Plane, mask one Plane and make linear least square fits involving the remaining 3 data points weighted by the current best estimate for this

Plane,Side	4,1	3,1	2,1	1,1
$x/\mu m$	114	69	31	56
$y/\mu m$	73	50	33	56

Table 1: Resolution estimation in micrometers for the modules indexed by Plane and Side (compactified z and x, see 4.2) using the method described in 4.3. The correlation in the x and y coordinate can be explained by a tilting angle not considered in the nominal geometry (without alignment).

Plane's resolution. This means that a data point from a module with known resolution worse than the others is weighted less than the others in the fit.

The vector obtained by the fit is intersected with the plane specified (actually a small error is introduced here as the increased distance caused by the tilted module). For each Plane, all residuals are combined to a distribution for which its standard deviation gives a value we shall call σ_{fit}. Using methods of error propagation, the uncertainties caused by the other detectors are calculated with:

$$\sigma^2_{error} = (\sum_i \frac{1}{\sigma_i^2})^{-1} \tag{1}$$

For each Plane, the σ_{fit} and σ_{error} now form a new best estimate for this Planes resolution according to the following equation:

$$\sigma^2_{module} = \sigma^2_{fit} - \sigma^2_{error} \tag{2}$$

This procedure is continued until all resolutions converge. These newly found resolution estimations are and more accurate than the estimation in 4.2.

The results of the modules with decent data quality (on one telescope arm (positive z/Plane) and one side (positive x/Side)) are listed in table 1.

For a result plot of the described procedure, please refer to figure 7.

5. Effect of small unknown rotation misalignment on module resolution

Please refer to figure 5 for our choice of coordinate system.

The detector modules used in our telescope are not perpendicular to the chosen z-axis, but tilted along the x- and afterwards along the y-axis to increase the resolution. The modules are mechanically fixed in a metal frame, but this metal frame is mounted manually and therefore may not be perfectly

aligned in regard to the other detectors. In the following calculation we analyze the effect of a small rotation angle variation on the resolution of the module. Let us only take the x and z-coordinates into account, as the rotation around the y-axis is most error-prone. Its angle is $\alpha = 30°$.

After mounting the modules, the telescope was positioned in a way that the beam (radius of about $1cm$) hits the modules in their center.

Assuming the incoming particles travel along the -z direction, we can set the track's x-coordinate constant. Let's further assume that the modules are rotated around the same x-coordinate relative to the particle beam axis (figure 8). The particle hits will correspond to different x-coordinates in the local coordinate systems of the modules (s_1 and s_2). We can therefore obtain:

$$s_1 \ \cos(\alpha) = x_0, \ s_2 \ \cos(\alpha + \epsilon) = x_0 \tag{3}$$

where $\alpha = 30°$ is the rotational angle and ϵ is the unconsidered angle error. Using the cosine addition rule and Taylor approximation for small ϵ, we obtain:

$$s_1 \ \cos(\alpha) = s_2 \ [\cos(\alpha) - \sin(\alpha)\, \epsilon] \tag{4}$$

$$s_1 = s_2 \ [1 - \tan(30°) \ \epsilon] = s_2 \ (1 - \frac{\epsilon}{\sqrt{3}}) \tag{5}$$

$$\Delta = s_1 - s_2 = \frac{s_2 \ \epsilon}{\sqrt{3}} \tag{6}$$

Let us assume, that the modules are rotated around the point 1 cm off the beam track. ($s_2 = 1cm$). From specification and experience we know that the modules resolution can be as low as 20 μm. The resolution increase by an unconsidered angle in the geometry construction of 1 degree is:

$$\Delta = \frac{10\,000\,\mu m \ 1° \ \pi}{\sqrt{3} \ 180°} \approx 100.8\,\mu m. \tag{7}$$

We see even a small rotation can decrease a module's resolution by a factor of 5 or more, if not taken into account. This justifies the need for alignment procedures, which aim to improve the discrepancy between the digital reconstruction of the geometry and its physical reality.

6. Summary of my other projects in this summer

Over the course of the summer I was able to help Stefano Mersi and Nikkie Deelen many times in the North Area, Prevessin, with small things regarding the telescope such as: transporting, rerouting cables, mounting of hardware, taping, calibrating a Voltage meter, and even rerouting AC waste water hoses :).

I also helped with many small computer related tasks, such as wiping, transferring and setting up PCs and a NAS, installing drivers, general tech support, enhancing Wikipedia articles (CMS and graphite) and writing small plot scripts.

I would like to thank Stefano and Nikkie, the summer student team, my new friends around the world and CERN as a whole for this amazing and unforgettable experience and hope to see many faces again in the challenges ahead.

[1] CMS Collaboration, The Phase-2 Upgrade of the CMS Tracker - Technical Design Report, `https://cds.cern.ch/record/2272264`, 2017.

[2] N. Deelen, Homepage for the CMS-CHROMIE telescope, `http://chromie-telescope.web.cern.ch/chromie-telescope/index.html`, 2018.

[3] P. Wimberger, GitLab repo for RotaController - a C++ library for usage of PI controllers under linux, `https://gitlab.cern.ch/pwimberg/RotaController`, 2018.

[4] P. Wimberger, GitLab repo for RotaContSupervisor, XDAQ, pos (pixel online software, `https://gitlab.cern.ch/pwimberg/pos`, 2018.

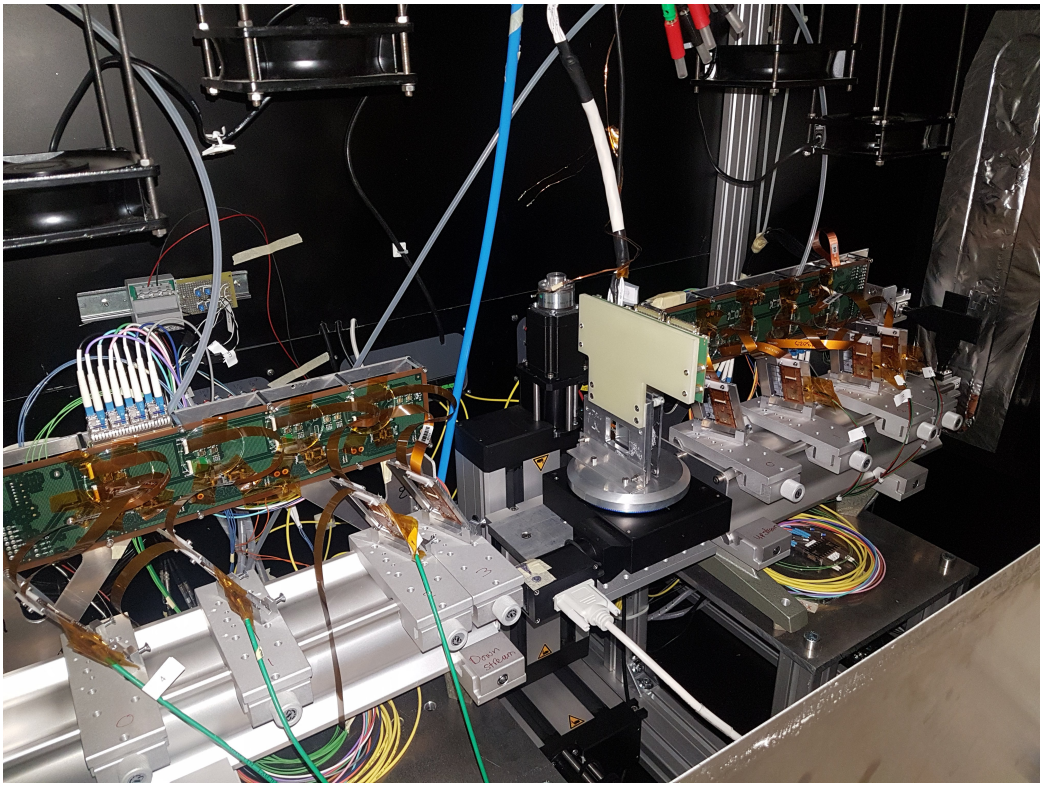

Figure 1: The inside of the CMS-CHROMIE telescope. You can see the two telescope arms (upstream right, where the particles originate from and downstream left) with one motherboard on each side and (most of) the modules connected to it. The motherboard sends the necessary information to the FED via optical fibers. On the back plane are many more necessary cables connected to the outside: Low and High Voltage Supply for the motherboard and modules, power for the cooling fans, which you can see in the upper part of the picture, voltage, temperature and humidity probes, controller cables for the X,Y and rotational stage, power supply for the Arduinos (temperature readout) and the subsystem for the DUT (Low, High voltage, readout,...). The DUT (in the middle of the picture) is mounted on the rotational stage, which itself is mounted on the X,Y stage. There is also a hose coming in from the top inducing nitrogen to keep humidity low in the closed box.

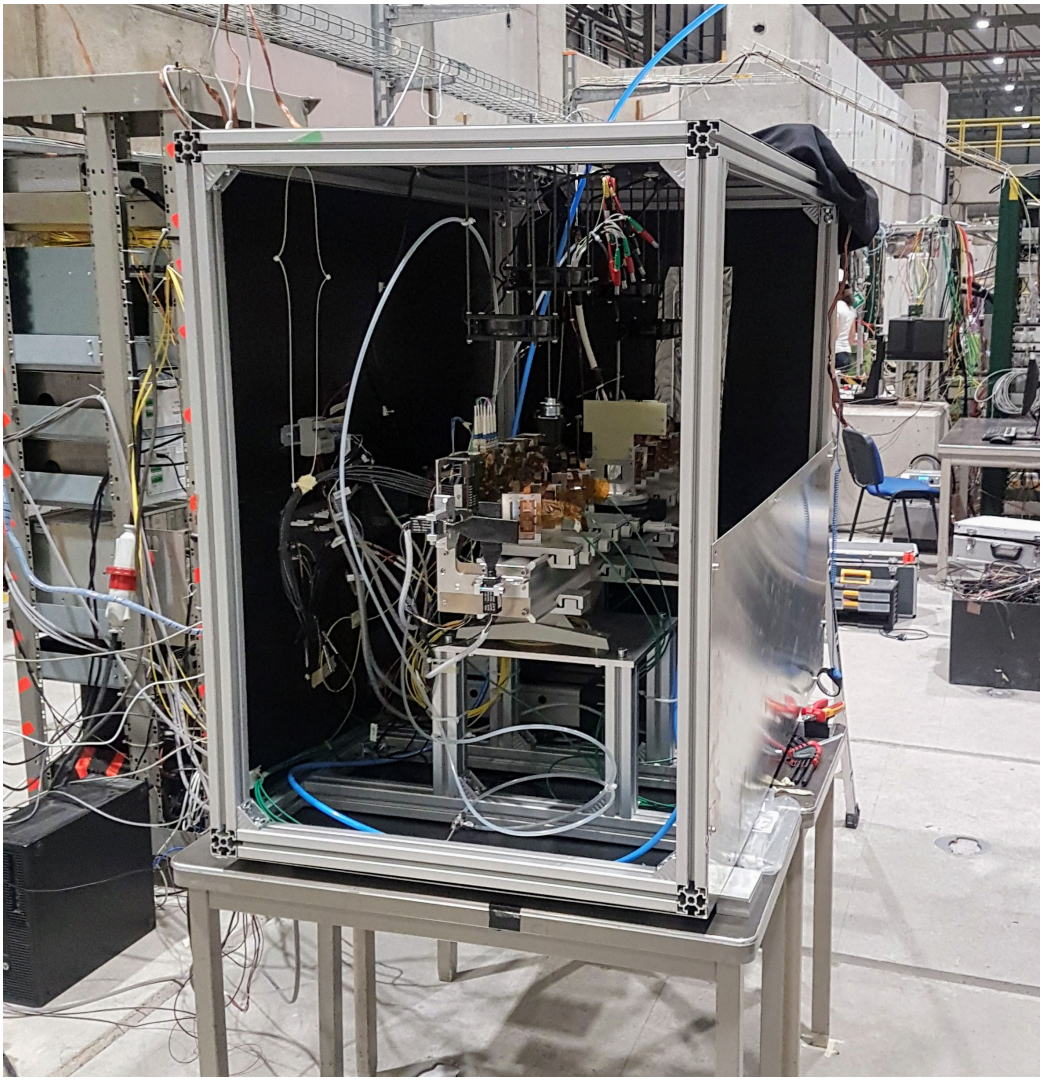

Figure 2: The CMS-CHROMIE telescope consists of many parts, described in detail in figure 1 and section 2. In this picture you can see the (downstream) scintillators attached to PMTs, which are used for triggering. On the left you can see one of the three computers currently in use and the rack (more details in figure 3). CHROMIE is currently located in the CERN North area H8 and supplied with particle beams by SPS.

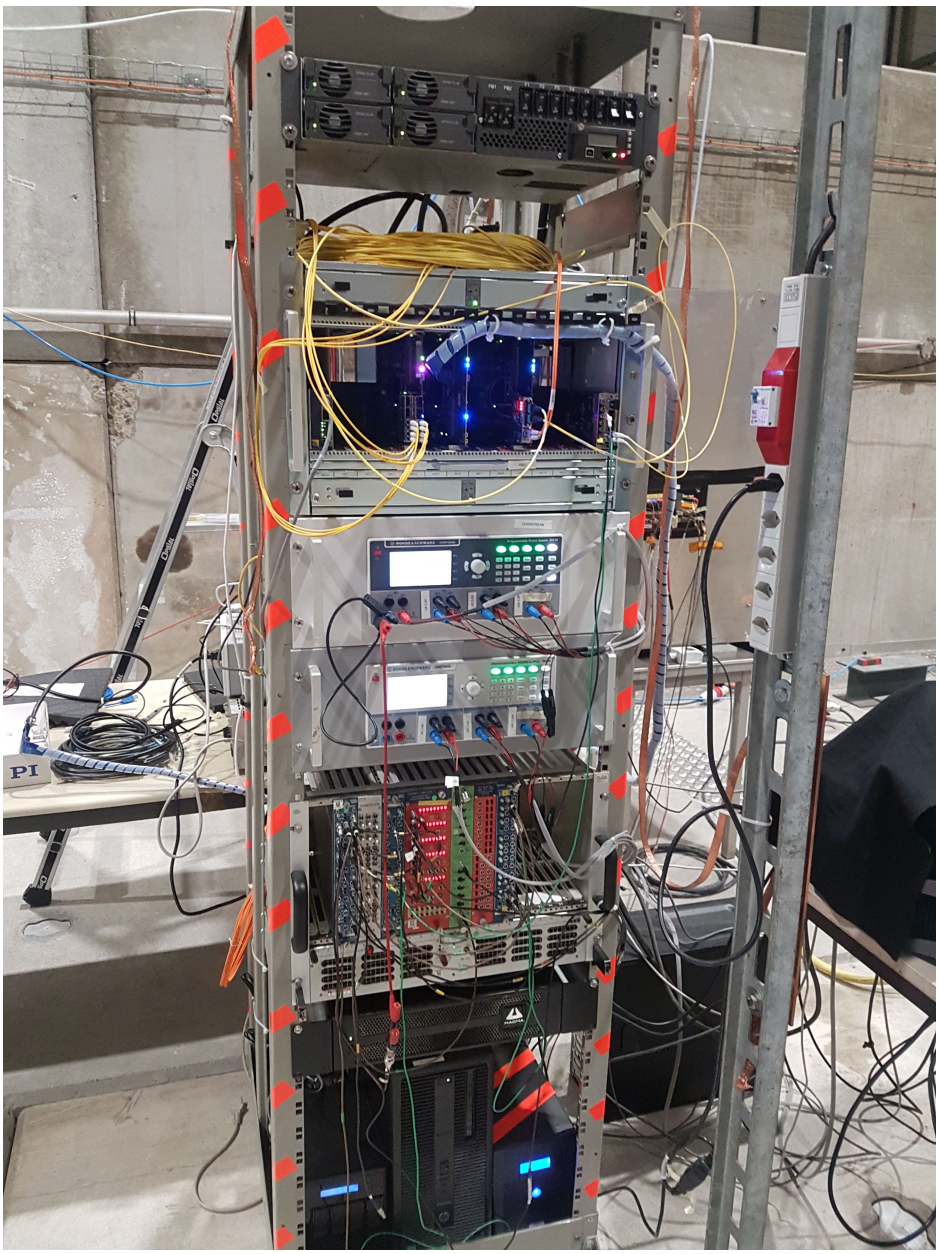

Figure 3: The CMS-CHROMIE telescope rack, which inhibits the μTCA crate (connected to with yellow fibres) with the FEC, FED and AMC13, power supplies (controlled digitally), the trigger logic (discriminator, shaper, delay, coincidence unit) and on the bottom a UPS (backup battery), a NAS and a PC.

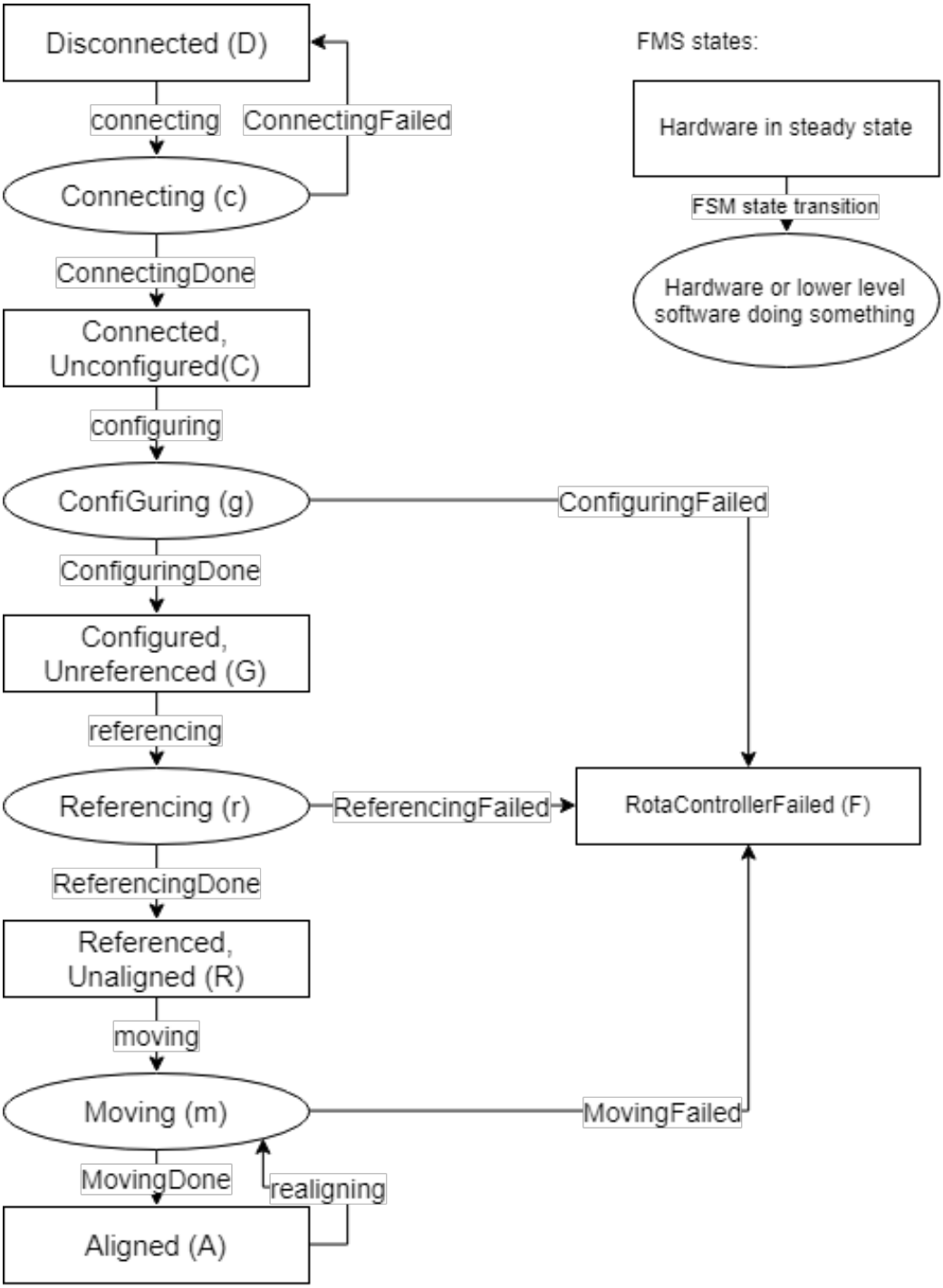

Figure 4: A representaton of the Finite State Machine (FSM) used by the XDAQ RotaContSupervisor to express the current state of the lower level software and hardware it controls. The FSM starts in the Disconnected State. Changing states in the FSM occurs instantly, therefore additional states are needed to represent hardware state changes, which take time (e.g. Connecting, Configuring, Referencing and Moving). The example states in the top right act as a legend.

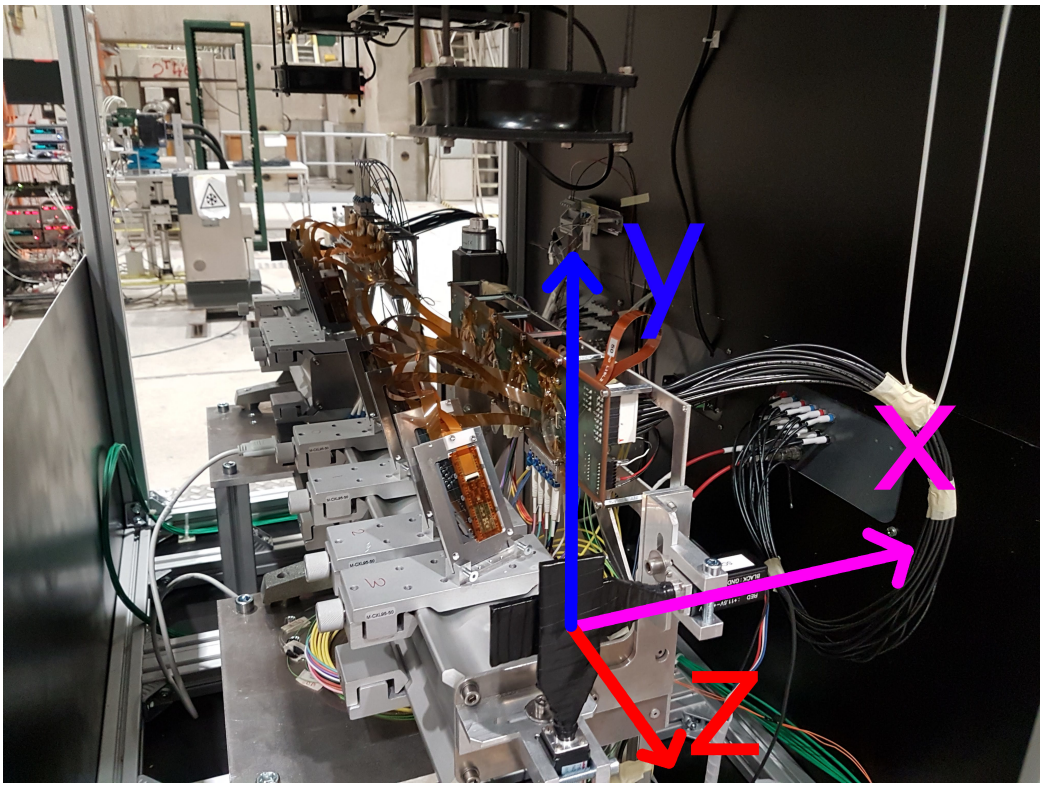

Figure 5: CHROMIE coordinate system convention. Particles are travelling in the -z direction (this convention is used because of many practical purposes). The upstream arm can be seen in the lower part of the picture and the downstream arm in the top part.

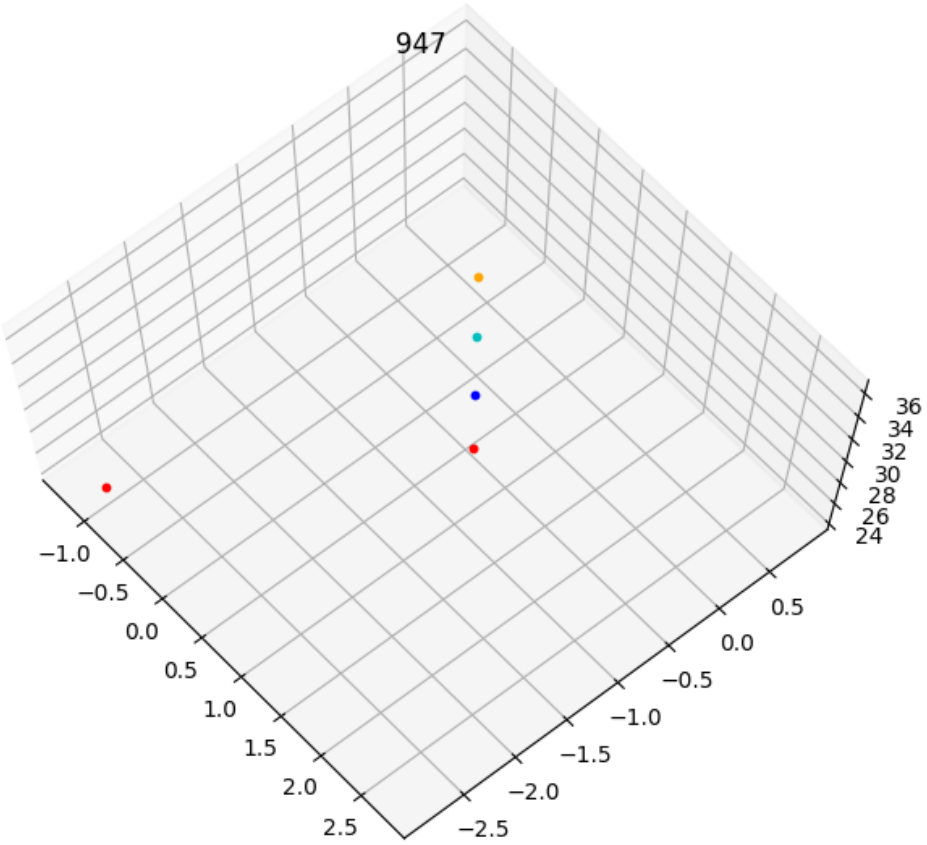

Figure 6: This is a 3D visualization of an ordinary event (number 947 in this specific run). Each plane is color encoded. Z=0 corresponds to the middle of the telescope. Particles travel in -z direction. X is the left axis and Y points up in reality. We see the 4 hits of a particle in the modules on one side of the telescope. We can safely assume, that the particle came from the SPS because of its straightness (travelled along the -z direction) and energy (it passed all detectors). We also see a random hit in one module on the other side. This can have multiple causes: noisy pixels, which send a signal, although there was no particle traveling through; particles scattered somewhere else, maybe even a different event; radioactive decay;...

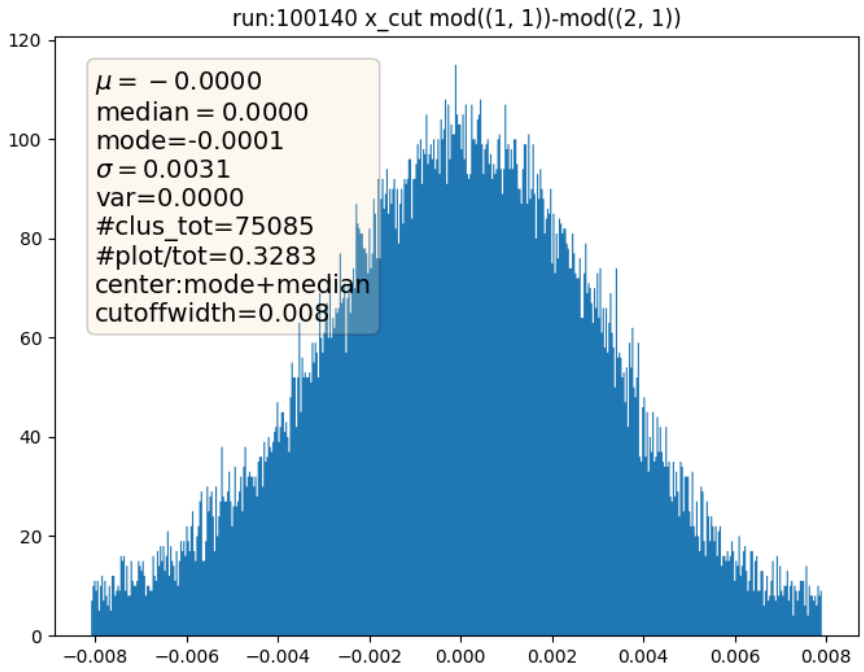

Figure 7: This plot shows the residual x-coordinate (in cm) of module on Side +1 and Planes 1 and 2 for the internal run number 100140. Whenever there is a hit in module 1 and module 2 in the same event, we calculate the residual $(x_1 - x_2)$ and add it to this plot. The existence of a sharp gaussian peak shows us, that we detect the same particle in multiple modules. The mean (center) of the gaussian (μ) indicates an offset in the x-direction perpendicular to the beam axis. For this plot, the modules were already manually aligned beforehand, so μ is 0. The standard deviation (σ) tells us, that the resolution to reconstruct the track from these 2 modules is $31\mu m$. This is very precise even without proper alignment. The resolution is not only dependent on the intrinsic pixel resolution, but also the collimation of the beam and most importantly of the correct rotational angle in the reconstructed geometry (section 5). This plot is cut to fully view the gaussian. We compute about 33% of the total hits for this plot and the obtained values. The curve is commonly modeled as a gaussian and a constant term (or a near-flat triangle caused by the convolution of the two constant noise distributions).

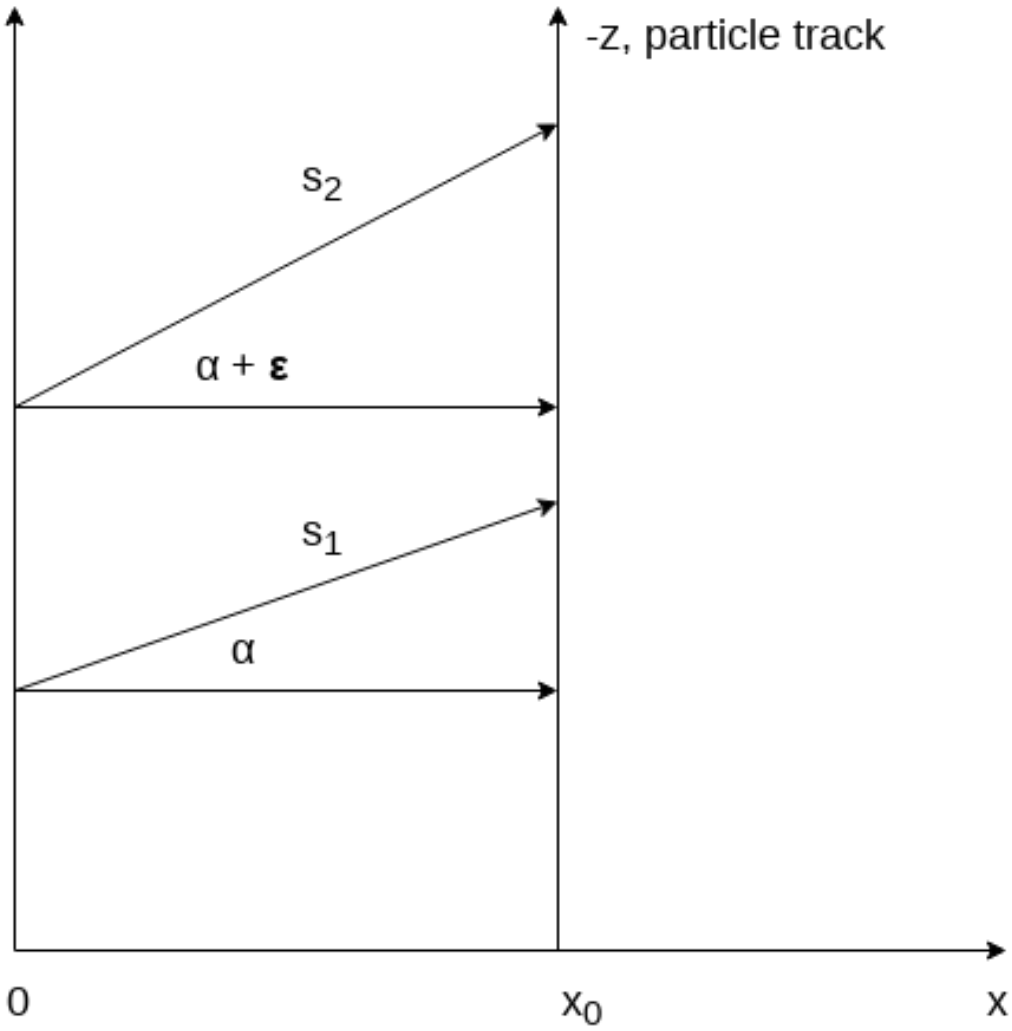

Figure 8: Illustration of the calculation for a small rotation misalignment (ϵ) in section 5. The modules are tilted by $\alpha = 30^\circ$ along the y-axis to be able to analyze resolution effects. x_0 is the distance from the particle track to the point the modules are rotated around. (Be aware that this is not constant for every particle track as the beam has a finite diameter.) s_1 and s_2 are the local coordinates of the module.