

Universidad de Cantabria

Facultad de Ciencias, Grado en Físicas

Estudio del alineamiento en experimentos de test beam a través de una simulación Monte Carlo

Study of the alignment of test beam experiments through a Monte Carlo simulation

Trabajo de Fin de Grado

Autor: Sergio Pérez Ortiz Tutor: Jordi Duarte Campderros Fecha: 5 de septiembre de 2023 A Jordi, por haber sacado tiempo de su apretada agenda para enseñarme y tutorizarme en mi trabajo

A Magdalena por sus tortillas que han servido de energía durante las largas tardes de estudio, a las hermanas Esteban por su férreo liderazgo, a Esqui por su paciente ayuda con el inglés de este trabajo, a Milagros (que no Milagritos) por ser la alegría de la familia, a Leticia por incomparable compromiso con la flora de la tierruca, a Paco, Pipi y su hijo Jesús por enseñarme otra forma de ver la vida, y, en general, a toda la gente de mi tierra natal de Bucaramaga.

Resumen

Aunque a primera vista el alineamiento de un dispositivo experimental puede parecer un problema simple o incluso trivial, en los experimentos de test beam ha demostrado tener una gran complejidad. Este trabajo busca ser una referencia a la que los estudiantes e investigadores puedan recurrir cada vez que se encuentren con este problema en sus propios experimentos.

El objetivo de este trabajo es el estudio sistemático del efecto que produce cada parámetro de desalineamiento sobre los observables relevantes. Con este fin se ha implementado una simulación Monte Carlo que modeliza una versión simplificada de un experimento de test beam. Se ha estudiado el efecto que cada parámetro tiene sobre la simulación, lo que ha servido para determinar la casuística que conecta el desalineamiento con los principales observables.

Una vez cumplido este objetivo, el estudio amplió su alcance para explorar aspectos más complejos. Se ha estudiado el efecto de combinar distintos parámetros de desalineamiento, de utilizar modelos más complejos y de incorporar ruido en la simulación para simular un experimento real.

Palabras Clave: Experimentos de test beam, desalineamiento del DUT, Monte Carlo, simulación de física de partículas.

Abstract

Identifying misalignment in a test beam experiment has proven to be a far more difficult problem that what it may appear at first glance. This project aims to be a reference that students and researchers can come back to whenever they encounter this problem on their own experiments.

The goal of this project is to systematically characterise the effects that each misalignment parameter has over the measured observables. In order to achieve this, a Monte Carlo simulation was used to model a simplified version of a test beam experiment. By varying the misalignment parameters in the simulation, the causal relationship between misalignments and the corresponding changes in key observables was established.

Having successfully achieved the initial goal, the study expanded its scope to explore more intricate aspects. This involved investigating combinations of different misalignment parameters, introducing more complex models, and incorporating simulated noise to enhance the realism of the simulation.

Keywords: Test beam experiments, DUT misalignment, Monte Carlo, particle physics simulation.

Contents

1	Intr	oducti	ion	1	
	1.1	Test b	eam experiments	1	
		1.1.1	State of the art in test beam experiments	3	
	1.2	DUT a	alignment	4	
2	Met	Methodology: Designing the simulation			
	2.1	Appro	ximations	6	
		2.1.1	Negligible telescope error	6	
		2.1.2	2-dimensional DUT	6	
		2.1.3	Particles follow a straight line	7	
	2.2	Simula	ating track data with Monte Carlo	7	
	2.3	Geome	etry	8	
	2.4	Variab	bles definition	8	
	2.5	Simula	ating DUT measurements	10	
	2.6	Code i	implementation	11	
	2.7	Toy m	odel	12	
3	Simulation Results 14				
	3.1	Perfec	t alignment	14	
	3.2	Indivi	dual misalignment parameters	16	
		3.2.1	Translation along X axis: δx parameter	16	
		3.2.2	Translation along Y axis: δy parameter	18	
		3.2.3	Translation along Z axis: δz parameter	19	
		3.2.4	Rotation around X axis: α parameter	20	
		3.2.5	Rotation around Y axis: β parameter	22	
		3.2.6	Rotation around Z axis: γ parameter	23	
	3.3	Combi	ining multiple misalignment parameters	24	
	3.4	Alignn	nent procedure	26	
4	Boyond the Toy Medel				
4	- Б еу			21	
	4.1 4.0	Chan	all distribution	21	
	4.2	Unarg		29	
5	Cor	clusio	n	35	

Chapter 1 Introduction

This first chapter introduces the main theoretical concepts needed to properly understand and contextualize the problem of device under test misalignment on test beam experiments.

The first section covers test beam experiments, as well as the main concepts and terminology that will be used throughout this project. Then it will dive into real world experiments from IFCA and CERN to show how a real world experimental set up may look like, and to give a brief overview of the state-of-the-art on the field. Once all this topics have been covered, it will be time to explain what exactly is DUT misalignment and why it is a common problem in test beam experiments.

1.1 Test beam experiments

A test beam experiment is a particle beam experiment designed to test a particle detector. They have proven to be particularly valuable for examining how a detector behaves and responds under conditions similar to those it will encounter when integrated into the actual experiment [1]. The detector, which is usually a new technology with unclear accuracy, is called the *Device Under Test*, or simply DUT.

This project will be focused on a specific type of test beam experiment, where the goal is to test the spacial resolution of a pixelated silicon detector. As stated in [2, p7], this kind of detectors are "solid-state analogous to gaseous ionization chambers". They are mostly used for reconstructing particles tracks in collider experiments, which can be achieved because the interaction of the passing particles with the semiconductor material creates charge carrier pairs that can be measured.

Designing an experimental set up with the goal of testing the spatial resolution of the DUT is not a trivial task. The initial thought that might come to mind is to place a known detector really close to the DUT, then make a beam pass through both of them and compare the measurements of the two detectors. This approach has several problems. First, if the particle tracks are not perfectly perpendicular to the detectors, which they are not in real experiments, then the same track intersects each detector at a slightly different position. Another problem is that this set up would be limited by the accuracy of the known detector, which is not exactly convenient when the main goal of the experiment is to test a new, potentially more accurate, detector.

In order to solve these problems test beam experiments use what is known as a particle telescope [3]. A *telescope* is an arrangement of multiple particle detectors, called telescope

planes, that allows the trajectory of the particles passing through to be reconstructed with high accuracy. The DUT is placed inside the telescope so that the intersection of the reconstructed tracks with the DUT can be calculated.



Figure 1.1: Diagram displaying a particle telescope. When a single particle goes through the telescope, each telescope plane will record a hit. These hits allow the reconstruction of the particle's track and, assuming the DUT position is known, the intersection of the track with the DUT plane shows which is the correct position that the DUT should measure, according to the telescope. Then, the position of the track when it intersects the DUT plane is compared with the DUT measurement to check the accuracy of the latter.

The telescope solves both of the previously mentioned problems. A track reconstructed

using multiple detectors is more accurate than a single detector hit, and non-perpendicular tracks are not a problem. However this set up introduces some new challenges, the main one being DUT misalignments, which will be discussed on the following section.

1.1.1 State of the art in test beam experiments



Figure 1.2: Scheme of the experimental set up used by Roa M. and Duarte J. on their test beam experiment [2]. Apart from the particle telescope and the DUT, this set up also involves a pair of scintillators, used to start the process of data acquisition, and a reference device (REF) with a higher temporal resolution that checks the hit association in multiple particle measurements. Measuring the passing of a fast moving particle naturally requires taking a lot of measurements on a small time interval. This requirement imposes a complex data treatment process, which involves a TLU that triggers data acquisition, a pair of Digital Test Boards (DTBs) that acquire the data, and the EUDAQ software that stores and manages the data.

Figure 1.2 illustrates the set up used for a real test beam experiment as opposed to the simplified diagram of figure 1.1. Fortunately, the intricacies of the set up do not radically change the behaviour of the system when subjected to a misalignment. The main patterns that appear when studying the relevant observables, and their connections with the alignment parameters, remain almost unchanged from the simplest to the most complex model. In fact, as we'll see later, using a simplistic model often leads to better results, as the patterns become more transparent. Therefore, a clear understanding of the simplified toy version of a test beam experiment shown in figure 1.1 is enough to understand all the concepts relevant to the problem of DUT misalignment.

As stated earlier, the purpose of a test beam experiment is to test the behavior of a piece of equipment, usually a particle detector, when it's subjected to a particle beam. Primarily, these experiments are used to test the precision of new equipment before it goes into production, but they can also be used to calibrate and test the durability of production ready detectors, and to study the behavior of particles in different materials.

Test beam facilities at CERN are heavily used, with bookings being consistently full [4]. Numerous test beam facilities exist at CERN, and each of them accommodates multiple projects on a monthly basis. Because of this it would be impossible to list every single experiment. Instead, this chapter is only going to mention just a handful of experiments, corresponding to the most cited recent papers on this field.

ALICE stands out as one of the best known experiments at CERN. It is specifically designed to study the properties of matter under extreme conditions, such as those that existed shortly after the Big Bang. Central to this endeavor are the ALICE AD modules, which are responsible for triggering data acquisition. A crucial experiment for testing the latest upgrade of this modules was the 2021 study by M. Broz et al [5], which used the CERN PS test beam facilities to examine the position dependence of the modules' efficiency.

The LHCf (Large Hadron Collider forward) experiment is another well known CERN experiment. Unlike the ALICE experiment mentioned earlier, LHCf does not study heavyion collisions but rather focuses on studying the forward region of high-energy particle collisions involving protons and heavy ions. To achieve this, LHCf uses specialized calorimeters. A crucial experiment for testing this calorimeters was the 2012 study by T. Maz et al.[6], which used the CERN SPS test beam facilities to examine the energy resolution and linearity of said detectors.

Finally it is important to mention the paper [1], as it specifically covers the same topic as this project: the characterization of pixelated silicon sensors with a test beam experiment. The study leveraged the electron beam facilities at CERN [7] and DESY [8] to asses the accuracy and efficiency of an experimental 3d pixelated sensor technology when exposed to a high radiation environment, similar to what the detector would experience if it was mounted at the High Luminosity Large Hadron Collider (HL-LHC) project [9].

1.2 DUT alignment

As previously stated, this project is focused on a specific type of test beam experiment that uses a telescope to reconstruct the particle's track, which is then compared to the measurement of the DUT. To meaningfully compare the track measured by the telescope and by the DUT their relative position must be known. The term *perfect alignment* is used to denote the ideal case where the DUT's position with respect to the telescope is measured with no error. On the other hand, *DUT misalignment* is the term used to describe any discrepancies between the measured and real DUT position. A straightforward example of DUT misalignment is shown in figure 1.3.

Each iteration of the alignment process consists of reducing the error in the measurement of the DUT's position and taking this into account to calculate the point where the track hits the device. Alternatively, the DUT could be moved so that its real position gets closer to the measured position. However, this is only possible for the first few iterations. On each iteration the adjustment becomes smaller and it quickly becomes impossible to move the DUT with the required accuracy. This means that aligning the DUT does not mean physically moving the device, as it may appear at first glance. Instead, the core of the alignment process revolves around precisely characterizing the DUT's position so that its measurements of the track can be compared to those of the telescope.

This alignment process must continue until the measured position of the DUT is close enough to its real position. Clearly, the real position of the DUT is unknown on a real experiment. However, as chapter 3 will prove, the effect of the misalignment over the results can be estimated with the residuals, which are defined as the difference between the track's



Figure 1.3: Diagram displaying a simple example of DUT misalignment. The error in the measurement of the DUT's position leads to an incorrect estimation of the point where the track hits the DUT.

position measured by the DUT and the telescope. Therefore, in practice the iterations must continue until the residuals are small enough, after which point the set up is considered to be aligned. In order to achieve alignment the residuals must be tiny, because this project studies experiments that test the spatial resolution of particle detectors, which naturally require great precision in position measurements. The exact precision requirement depends on the experiment. For example, in the previously mentioned experiment [2], the DUT was considered to be aligned when the residuals where less than 0.005mm for every recorded event.

Although not the primary focus of this project, it is worth mentioning some of the existing track alignment methods. In this context, special recognition goes to Milepede II, a software developed by the German research center DESY (Deutsches Elektronen-Synchrotron, which translates to German Electron Synchrotron). As stated in the official documentation [10]: "Detector alignment and calibration based on track fits is one of the problems, where the interest is only in optimal values of the global parameters, the alignment parameters. The method, called Millepede, [is used] to solve the linear least squares problem with a simultaneous fit of all global and local parameters, irrespectively of the number of local parameters."

Chapter 2

Methodology: Designing the simulation

This chapter will be focused on explaining the approximations, methodology, and in general the process that was followed to design the Monte Carlo simulation. The main goal is to create the simplest possible model, the *toy model*, of a test beam that allows the study of DUT alignment.

2.1 Approximations

With the purpose of constructing the toy model, a series of approximations will be made to mimic the real physical phenomena in the simplest way possible.

2.1.1 Negligible telescope error

In most test beam experiments it is possible to achieve a very accurate spatial resolution in the telescope's measurements. This is the result of multiple factors. First, the telescope is using the information from the measurements of multiple planes to reconstruct the track. Through this mechanism the telescope is able to achieve an accuracy much higher than the one from a single plane.

Additionally, it should be taken into account that any DUT has some strict quickness restrictions, i.e. it has to take many measurements in a very small time interval, therefore limiting its accuracy. Therefore it is possible to design the test beam experiment in a way that has less time restrictions than the DUT, allowing the use of slower but more accurate detectors in the telescope.

As a result of this factors, most test beam experiments have a telescope that is significantly more accurate than the DUT. For this reason, the model assumes a negligible telescope error which means that, in the simulation, the telescope perfectly reconstructs the track of every particle.

2.1.2 2-dimensional DUT

This project was inspired by M. Roa and J. Duarte's work on particle sensors [2] where they characterized the behavior of a silicon based detector that consists of multiple connected 3d

pixel sensors. On said project they already encountered the problem of DUT misalignment when trying to measure the 2d position of each particle inside the cross-sectional area of the beam. This proves that the effects of DUT misalignment can be observed even when we are only interested in measuring a 2d position. Therefore, we can safely assume a 2-dimensional DUT and the model will still be useful to study the misalignment effects.

Given this approximation it is natural that from now on we will only consider the 2d position of the particle inside of the DUT as the variable of interest. We will still consider the third coordinate to do the internal calculations but at the end of the simulation we will only be left with the 2d observables.

2.1.3 Particles follow a straight line

In test beam experiments the particles arrive to the telescope from a narrow long cylinder. The particles must stay inside the confines of this geometry, but they could follow a slightly curved path. However given the small cross sectional and the high speed at which the particles travel, we can assume that the curvature radius is much bigger than any distance considered in the experiment. Therefore we will consider that the particles follow a straight line.

2.2 Simulating track data with Monte Carlo

As stated in approximation 2.1.1 the telescope can be considered to have perfect accuracy. This means the track reconstructed by the telescope can effectively be interpreted as the real track followed by the particle. This raises the question of how exactly one could simulate reliable particle track data.

An immediate consequence of approximation 2.1.3 is that in order to simulate the track of a particle we simply need to generate a straight line inside of the simulation.

The plane that contains the 2-d DUT can act as a reference to build the coordinate system. To define a straight line we only need to set its intersection point (x, y) and the angle it forms with the plane (θ_x, θ_y) . After this point it seems like the problem of generating a particle track is already solved. However it's not clear exactly how can we generate this variables.

We could just start with all the variables set to 0 then increment the value of the first variable by Δx amount until we reach a boundary x_{max} then do the same for negative values until we reached x_{min} , do the same for all values and try every single combination. This seems like a cumbersome solution for what it looked like a simple problem. Also using this approach runs into multiple problems, for example, its not possible to pick the number of particle track without introducing a significant bias.

The solution is to use the Monte Carlo approach: just use (pseudo)random numbers. The term *Monte Carlo* is widely used to refer to any computational technique that involves using random numbers to generate a random sample from the desired probability distribution.

For our use case we must set some boundaries for the variables and then pick a probability distribution. If we wanted the same behavior as we had without using Monte Carlo then we should choose a uniform distribution. Once that's done we only need to specify the number of particle tracks desired, n, and use any Random Number Generator to get the desired data points.

It's important to point out that using Monte Carlo for a physics simulation is not exactly a break through idea. Indeed there's a long history of Monte Carlo based particle physics simulations. One in particular, Geant4, deserves a special mention. It is the most sophisticated and comprehensive simulation tool available for particle physics, and its documentation [11] is a great resource to further explore the topic of Monte Carlo simulations.

2.3 Geometry

As stated in 1.2, in order to study DUT alignment on its simplest form, the model only needs to compute the intersection point of the track with the measured and real position of the DUT. Given approximation 2.1.3, the model can neglect any particle interaction, as these interactions cannot modify the particle's track. Therefore we can remove the telescope planes from the simulation without any consequences.

After this simplification it is clear that the only geometry elements required in the model are the track, which is a straight line following approximation 2.1.3, and the measured and real position of the DUT, which is a 2-dimensional polygon following approximation 2.1.2.



Figure 2.1: Diagram displaying the geometry of the simulation. There are only 3 elements: one straight line, representing the track, and two rectangles, representing the measured and real position of the DUT.

2.4 Variables definition

As stated in 2.1.2 this model considers a DUT that measures the two dimensional position of the particles that pass through it. Therefore the main observables of the simulation will be the (x, y) coordinates of a point inside of the DUT.

The true position, $(x, y)_{true}$, is the point of intersection of the track with the real position of the DUT, measured relative to the center of the real DUT. This represents the position that would be measured by a perfectly accurate DUT.

The telescope position, $(x, y)_{telescope}$ is the point of intersection of the track with the measured position of the DUT, measured relative to the center of the measured DUT. This



Figure 2.2: Diagram that illustrates the physical meaning of the key observables. The top section shows the points in the 3D geometric diagram, introduced in section 2.3. The middle section introduces the scattering plots that will be used from now on to display the individual data points, and clarifies the relationship between these plots and the 3D diagram. The bottom section shows the key variables $(x, y)_{telescope}$, $(x, y)_{true}$, and $(x, y)_{dut}$ inside of the scattering plots.

represents the position that would be measured by the telescope. If alignment is perfect then this coincides with the true position.

The *dut position*, $(x, y)_{dut}$ is the position measured by the real DUT, relative to its center point. In the simulation, this is calculated as a function of the true position, $(x, y)_{dut} = f((x, y)_{true})$. How exactly this calculation is performed will be the topic of the following section (2.5).

On a real experiment the true position is unknown, and the only positional data available is the position measured by the telescope, $(x, y)_{telescope}$, and the DUT, $(x, y)_{dut}$. In addition to this, it is also possible to use the telescope to measure the impingement angles, θ_x and θ_y , that were introduced in section 2.2.

In order to extract as much information as possible from these observables, it is useful to define a new variable

$$\Delta x = x_{telescope} - x_{dut} \tag{2.1}$$

$$\Delta y = y_{telescope} - y_{dut} \tag{2.2}$$

where Δx and Δy are known as the *residuals*.

As previously explained, in a real experiment only two positional measurements are taken, one with the DUT and one with the telescope. The residuals, Δx and Δy , quantify the difference between these two measurements.

2.5 Simulating DUT measurements

In this simulation we are trying to mimic a pixelated silicon detector such as the one used in experiments [2] and [1]. As mentioned in Section 1.1, the working principle behind this kind of detector is the creation of charge carrier pairs as a result of the interaction between the passing particle and the semiconductor material. The most simplified description of the measurement process is that whenever a particle hits some pixel of the DUT, some amount of charge carriers will be measured on that pixel.

In order to replicate this process, the simulated DUT is going to be divided into a number of pixels (generally we are going to use a 32x32 pixel DUT to mimic the real device found in [2]). Firstly the true position is calculated, which represents the point where the particle hits the DUT, as discussed in section 2.4. When a particle hits the DUT it will hit some pixel inside of it. That pixel will activate and the position measurement of the DUT will be the position of the center of said pixel. This process is illustrated on figure 2.3.

An important caveat is that the real pixelated device used in experiments [1] and [2] does not simply measure the position of the particle when it hits the device. Instead it can be used to reconstruct the 3D track followed by the particle when it is inside of the geometry of the device. However, because of approximation 2.1.2, the simulation is only considering a 2-dimensional DUT. This means that it can be considered that the DUT measures the 2D position of the particle instead of its 3d track.

In the real device the pixel directly hit by the particle is not the only one that generates charge carriers. Instead, the adjacent pixels will also measure some amount of charge. When the particle hits the DUT in a point close to the boundary between two pixels, it



Figure 2.3:

Diagram that illustrates the process followed to simulate the DUT measurements. First the true position, $(x, y)_{true}$ where the track hits the DUT is computed (1). The true position belongs inside some pixel, that pixel is considered to be activated (2). The position measured by the simulated DUT, $(x, y)_{dut}$, is the center of the activated pixel (3).

is even possible that two different pixels will measure the same amount of charge. This mechanism allows to reconstruct the position of the hit using information from more than one pixel, enabling higher accuracy. This process, known as *cluster sharing*, will be discussed on further detail in chapter 4, where it will be used to build more complex realistic models. However, for the toy model, the single pixel measurement discussed on this section is accurate enough.

2.6 Code implementation

The previous sections introduced the foundations required to pose the toy model. However, the question of which tools can be employed to execute this simulation and analyze its results remains to be addressed.

The project involves a lot of data analysis and visualization, and it could be argued that it is the most important part because it is the one that will be visible to the reader. The programming language python is the industry standard for projects centered around data analysis, making it a natural choice for this project. As stated by google engineer J. VanderPlas on his book *Python Data Science Handbook* [12, p2] : "Python has emerged over the last couple decades as a first-class tool for scientific computing tasks, including the analysis and visualization of large datasets. [...] The usefulness of Python for data science stems primarily from the large and active ecosystem of third-party package" This project used 4 of these third-party packages: *NumPy*, *Matplotlib*, *Seaborn* and *Click*.

Many different resources discuss in detail why vanilla python, i.e. python without any third-party libraries, is slow and not suited for performance critical applications, such as the chapter *Performance* from the book *Python in a Nutshell* [13] or the preface of *High Performance Python* [14]. Critically, python is very inefficient at array operations, which are the core of the calculations needed to implement the simulation. The tool that enables the simulation to be run without suffering a severe lack of performance is NumPy, which is a library that allows an "efficient implementation of the [array] structure together with basic arithmetic and linear algebra" [15].

Matplotlib is the plotting library that enables data visualization in python [16, p253]. Seaborn is a library built on top of Matplotlib that provides built in functions to easily replicate the most common plots used in a scientific article, such as scattering plots with a linear fit [16, p268]. In this project the latter is used for standard scattering graphs and histograms, such as the ones from figure 2.4, while the former is used for graphs that require more customization, such as figure 2.3 or figure 3.1.

As stated on the official website [17], "Click is a Python package for creating beautiful command line interfaces [CLIs] in a composable way with as little code as necessary.". This library was used to build a CLI, enabling quick adjustments of simulation parameters, e.g. number of events, without having to manually modify the source code.

We published the final version of the source code [18] so that anyone can experiment with the simulation if they wish to do so. Further information on where to access the simulation as well as how to use it appears at the end of this project, in Chapter 5.

2.7 Toy model

As stated at the beginning of this chapter, the goal was to create the simplest working simulation that allows the study of DUT misalignment, which we named the toy model. After all the previous considerations we are now able to achieve this goal.

First the simulation employs the geometry explained in section 2.3, building an empty world with a straight track intersecting two rectangles that symbolize the measured and real position of the DUT.

The track is defined by its incidence angle, θ_x and θ_y and the point where it intersects the real position of the DUT, $(x, y)_{true}$. As stated in section 2.2 the Monte Carlo method will be used to generate a sample of this variables. In order to do this, we first need to specify the probability distribution and the boundaries.

A reasonable boundary for the incidence angle is $\pm 0.5^{\circ}$, that is $\theta_x, \theta_y \in [-0.5^{\circ}, +0.5^{\circ}]$, which is a value that you may expect on a real experiment. Similarly, the boundary for the true position of the hits is ± 16 pixels, that is $x_{true}, y_{true} \in [-10 \text{px}, +10 \text{px}]$, meaning the track must hit the DUT on its central 20x20 pixel region. The toy model considers that these 4 variables follow a uniform distribution, meaning that any value in the specified boundaries is equally likely. The more complex models that will be introduced in chapter 4 use a Gaussian instead of a uniform distribution, which is closer to the measurements of a real experiments.



Figure 2.4: Plot showing the histograms for the values of θ_x , θ_y , x_{true} and y_{true} obtained in a test run of the toy model using n = 10000 events. The plots clearly show the use of a uniform distribution, the boundaries at $\pm 0.5^{\circ}$ and ± 10 pixels, and the random noise due to a finite sample.

To initialize the simulation the user must specify the number of events simulated, n, and the misalignment parameters (these parameters will be explained on detail in chapter 3). Once this values have been specified, the simulation will generate a sample of n tracks with random θ_x , θ_y , x_{true} and y_{true} values.

The misalignment parameters determine the measured position of the DUT relative to its true position. The DUT then computes its intersection point with each track, which corresponds to the position measurements of the telescope, $(x, y)_{telescope}$. Finally, the simulation computes the position measurements of the DUT, $(x, y)_{dut}$, following the simple procedure outlined in section 2.5.

Once all the measurements of the telescope and DUT have been simulated the data is plotted into multiple graphs for later analysis. First the data is displayed into the scattering plots introduced in figure 2.2. These plots show the true position of the track as well as the position measured by the telescope and the DUT. The black grid represents the pixels of the DUT. The data is also plotted into multiple different graphs, such as Δx vs. x or Δx vs. y, with the goal of identifying any correlations that may arise between different variables. These graphs will be discussed in detail on chapter 3.

Chapter 3

Simulation Results

The preceding chapter explained the design process behind the toy model, the simplest model that allows the study of DUT alignment. This chapter is focused on the data analysis of the results of said model. First, it will examine the results of a run with perfect alignment. Then, it will introduce the meaning of each misalignment parameter and the effects of increasing that parameter in the simulation. Following this, multiple parameters will be combined to observe emergent effects and identify an approach to streamline this scenario into that of individual misalignment parameters. Ultimately, everything will be brought together to deduce the best alignment procedure, addressing the realistic situation of combining all misalignment parameters.

3.1 Perfect alignment

This section analyses the results of a simulation run in the toy model with perfect alignment.



Figure 3.1: Plots showing the true position as well as the position measured by the telescope and DUT for all the n = 1000 events simulated. Each point corresponds to a single event, *i*.

On the case of perfect alignment, the measured and real position of the DUT coincide. Therefore the true position of the hits equals the position measured by the telescope, $x_{telescope}^{(i)} = x_{true}^{(i)}, y_{telescope}^{(i)} = y_{true}^{(i)}$. This fact can be observed in figure 3.1 where the plots for the true and telescope positions have the exact same data points. Because of



Figure 3.2: This figure consist of 2 histograms and 6 scattering plots. The histograms are used to display the distribution of the residuals, while the scattering plots are used to identify any correlations that may arise between different variables.

this, the telescope measurements have no uncertainty, and the only source of error is the intrinsic uncertainty of the DUT measurement, which corresponds to the variance of a binary distribution with the width of the pixels' pitch. This uncertainty is a direct consequence of the binary resolution, which is the name given to the spatial resolution of the pixelated sensor.

Figure 3.1 clearly shows the way in which the true position translates to the position measured by the DUT. Any point in the true position plot moves to the closest pixel center in the DUT plot, which means that the latter only displays positions that sit exactly in the middle of a pixel. The holes in the DUT plot correspond to the pixels that did not contain any data points in the true position plot.

The residuals histograms in figure 3.2, show that Δx and Δy follow a uniform distribution ranging from -0.5 to +0.5 pixels. As previously stated, this variance is entirely explained by the intrinsic uncertainty of the DUT measurement. In the best case scenario, the true position falls right in the center of a pixel, coinciding with the corresponding DUT measurement, and resulting in zero residuals Δx , $\Delta y = 0$. In the worst case scenario, the true position falls in the corner of a pixel, at a distance of Δx , $\Delta y = 0.5$ px from the closest pixel center. Because the true position in sampled from a uniform distribution, any value that falls between this extremes is equally likely, and therefore the residuals will follow a uniform distribution ranging from -0.5 to +0.5 pixels.

The Δx vs. $x_{telescope}$ and Δy vs. $y_{telescope}$ plots in figure 3.2 display a curious pattern. Examining just the region corresponding to one pixel, such as the range $x \in [0, 1)$ px there is a perfect correlation between the variables. The underlying cause of this pattern is that the DUT measurement always corresponds to the center of a pixel. This means that multiple $(x, y)_{true}$ points are mapped to a single $(x, y)_{dut}$ point, which leads to a correlation between $x_{dut} - x_{true}$ and x_{true} inside an individual pixel.

If the full range of $x_{telescope}$ and $y_{telescope}$ is considered, then the slope of the linear fit is 0. If noise > 0.5px was introduced into the data, the diagonal lines would disappear and these plots would be identical to all the other scattering graphs. This observation, among

with other insights obtained from the more complex models of the fourth chapter allow us to deduce that the diagonal lines of the Δx vs. $x_{telescope}$ and Δy vs. $y_{telescope}$ plots will not appear in real world data.

As a conclusion, in the case of perfect alignment the Δx and Δy histograms show a uniform distribution with a zero mean and the scattering plots have a zero slope. In both cases the variance in Δx and Δy is determined by the intrinsic uncertainty of the DUT measurements.

Finally, a key point is that figure 3.2 shows the results of a run with n = 10000 events, while the figure 3.1 corresponds to a run of just n = 1000. This is a deliberate choice that will be consistent across this whole chapter. A graph that plots variables against each other, such as the one in 3.2, will always benefit from more data points because the correlation between different variables will appear more clearly with a higher n. The opposite is true for the graphs in figure 3.1. If this graphs plotted n = 10000 events then there would be so many data points that the graphs would just show a 20x20 pixel square.

3.2 Individual misalignment parameters

This section will introduce each misalignment parameter individually. First, a diagram will be used to explain the physical meaning of each parameter, then a simulation will be run with that misalignment parameter set to a non-zero value. The resulting data will be plotted in the same graphs that we used in the previous section in order to show the differences between each case and the case of perfect alignment.

The misalignment parameters determine the difference between the real and measured placement of the DUT. In the simulation the real and measured positions of the DUT are represented by 2-dimensional polygons occupying 3D space. In order to specify the relative position of these polygons 6 different parameters must be set, corresponding to the 6 degrees of freedom of the system.

3.2.1 Translation along X axis: δx parameter

The first misalignment parameter is denoted as δx and quantifies translation along the X axis. More precisely, δx parameter determines the distance along the X axis that separates the center point of the measured position of the DUT from the center point of the real position of the DUT.

Based on the information presented in figures 3.4 and 3.5 it can be deduced that the primary indicator of a δx misalignment is the Δx histogram. The mean of the histogram, for example +4.99px in figure 3.5, is a highly accurate approximation of δx .

The true position of the track as well as the position measured by the DUT are independent of the misalignment parameters. To illustrate this fact the simulation used the same same random seed in 3.4 as it did in the run with perfect alignment that is displayed in 3.1. Because of this the middle and right most plots in both figures are the exact same, which shows that $(x, y)_{true}$ and $(x, y)_{dut}$ are not affected by the misalignment parameters. On the other hand, track positions measured by the telescope are indeed dependent on these parameters. Based on this reasoning, for the rest of this section only the telescope plot will be shown, and not the DUT and real plots, in order to avoid unnecessary repetition.



Figure 3.3: Diagram that illustrates translation along the X axis. The graph on the left shows a 3D layout of the coordinate system while the graph on the right displays a projection of the system onto the XY plane. On the latter, the green point symbolizes the track impinging perpendicularly on the plane from above.



Figure 3.4: Plots showing the true position as well as the position measured by the telescope and DUT for all the n = 1000 events simulated in a run with a misalignment of $\delta x = 5$ px. The position measured by the telescope is the only one affected by the misalignment. All the position measurements are shifted 5 px to the right, which corresponds to the positive x direction, as it can clearly be seen in the corresponding plot.



Figure 3.5: This figure consist of 2 histograms and 6 scattering plots showing the results of a run with n = 10000 events and a misalignment of $\delta x = 5$ px. All the scattering plots show no correlation between the variables, just as they did in the case of perfect alignment. Indeed, the only plot that changes with respect to that case is the Δx histogram. The histogram still shows a uniform distribution with a variance of ± 0.5 px, but the mean is equal to 4.99px instead of 0.

In the same fashion, any of the plots from figure 3.4 that display the same patterns as the ones shown in the case of perfect alignment (figure 3.2) will not be shown. In summary, any histogram with a 0 mean, and any scattering plot that shows no correlation will be omitted.

3.2.2 Translation along Y axis: δy parameter

The second misalignment parameter is denoted as δy and quantifies translation along the Y axis. This case is very similar to the first one. Based on the information presented in figure 3.7 it can be deduced that the primary indicator of a δy misalignment is the Δy histogram. The mean of the histogram is a highly accurate approximation of δy .



Figure 3.6: Diagram that illustrates translation along the Y axis showing a 3D layout of the coordinate system and a projection of the XY plane.



st 300 -200 -100 -3 4 5 6 Δy (pixels)

500

400

(a) Positions measured by the telescope. Comparing to the case of perfect alignment, all the positions are shifted upwards (i.e. in the positive y direction) by 5 px.

(b) Δy histogram. The distribution is centered at +5px, instead of being centered at 0 like it was in the case of perfect alignment.

Figure 3.7: Results of a run with n = 10000 events and a misalignment of $\delta y = 5$ px. The plot on the left only displays the first 1000 events.

3.2.3 Translation along Z axis: δz parameter

The third misalignment parameter is denoted as δz and quantifies translation along the Z axis. Based on the information presented in figure 3.9 it can be deduced that the primary indicator of a δy misalignment is a linear correlation between the variables Δx vs. θ_x and Δy vs. θ_y . If these variable pairs are plotted against each other, then the slope can be used to obtain the value of the parameter δz , given that $\alpha = \beta = \gamma = 0$.



Figure 3.8: Diagram that illustrates translation along the Z axis showing a 3D layout of the coordinate system and a projection of the YZ plane. If the impingement angle is zero, then this translation has no effects on the DUT measurements, assuming the real and measured DUT are parallel. However, if either θ_x or θ_y have non-zero values then the telescope measurements will be affected by the misalignment, as shown on the right most graph.

The hits displayed in figure 3.9 (a) are different from the case of perfect alignment because not all the dots are placed inside the central 20x20 pixel region. However there is not a clear pattern like in previous two cases. This is due to the fact that the residuals only depend on the angle and not on the position. This graph does not display any information regarding the impingement angle, and therefore the displacement of each dot with respect to the case of perfect alignment appears random.

The scattering graphs 3.9 (b) and 3.9 (c) clearly show a linear correlation between the



Figure 3.9: Results of a run with n = 10000 events and a misalignment of $\delta z = 30$ px.

variables. Assuming that real and measured positions of the DUT are parallel, which can only occur when $\alpha = \beta = \gamma = 0$, then the displacement of the telescope measurements due to a δz misalignment is

$$x_{true} - x_{telescope} = \delta z \tan \theta_x \implies \Delta x \approx -\delta z \cdot \theta_x$$

which uses $x_{dut} \approx x_{true}$, equation 2.1, and the small angle approximation $(\tan \theta_x \approx \theta_x)$ to get to the final result.

Similarly, it can be obtained that $\Delta y \approx -\delta z \cdot \theta_y$. This shows that the slope of the graphs can be used to estimate the value of the parameter δz . For example the slope of the graphs 3.9 (b) and 3.9 (c) is $m_{(b)} = m_{(c)} = -0.523 \text{px/}^{\circ}$, which converted to radians is -29.97 px, and therefore $\delta z \approx 29.97 \text{px}$ which is remarkably close to the actual value of 30 px.

3.2.4 Rotation around X axis: α parameter

The fourth misalignment parameter is denoted as α and quantifies rotation along the X axis. Based on the information presented in figure 3.11 it can be deduced that the primary indicator of a α misalignment is a linear correlation between the variables Δy and y. If these variables are plotted against each other, then the slope can be used to obtain the value of the parameter α , given that $\gamma = 0$.



Figure 3.10: Diagram that illustrates rotation around the Z axis showing a 3D layout of the coordinate system and a projection of the YZ plane.

When compared to the case of perfect alignment, the hits displayed in figure 3.11 (a) are shifted closer to the center in the Y axis. The shift is greater for the points further away from the center, i.e. the points with a greater y.



Figure 3.11: Results of a run with n = 10000 events and a misalignment of $\alpha = \pm 30^{\circ}$.

The scattering graphs 3.11 (b) and 3.11 (c) show a linear correlation between the variables. Applying the 3D rotation matrix [19, p167] and considering that the error in the DUT measurements is negligible, $(x, y)_{dut}^{(i)} \approx (x, y)_{true}^{(i)}$, one can derive the displacement of telescope measurements caused by an α misalignment

$$y_{telescope} = (\sin \alpha \sin \gamma \sin \gamma + \cos \alpha \cos \gamma) y_{true} + (\cos \beta \sin \gamma) x_{true}$$

if $\gamma = 0 \implies y_{telescope} = \cos \alpha \cdot y_{true} \implies$
 $\Delta y \approx \left(1 - \frac{1}{\cos \alpha}\right) y_{telescope}$

If that approximation was strictly correct, then all the measurements would follow a perfectly thin straight line. Instead there is a variance of ± 0.5 px from the center of that ideal line because of the intrinsic intensity of the DUT measurements, just like in the case of perfect alignment (figure 3.2).

The interpretation of this result is that, if the misalignment in the γ parameter has already been solved, then the slope of the Δy vs. y graph will be approximately equal to $1 - 1/\cos \alpha$. This expression allows the calculation of the absolute value of α , but it does not provide any information about the sign of α . This is the consequence of $1 - 1/\cos \alpha$ being a symmetrical function, meaning that $f(\alpha) = f(-\alpha)$, which explains why graphs (b) and (c) appear to be completely identical. In order to solve the sign, an additional measurement must be taken, changing α by a small positive amount. If the slope increases then α is positive, otherwise it is negative.

For example, the figures 3.11 (b) and 3.11 (c) both have a slope of m = -0.155 that translates to an absolute value of $|\alpha| = 30.0^{\circ}$ which is exactly equal to the actual value of α that we set before starting the simulation. This shows that the formula can give a very precise result in the ideal case. On a real case however, γ cannot be exactly 0 and noise does not allow a 100 % accurate measurement of m. As previously stated, we would need an additional measurement to infer the sign of α .

Finally it is important to note an angle as high as 30° could never appear on a real experiment. Such a misalignment could easily be spotted and corrected visually. The only reason why such a high value is used is so that the patterns in figure 3.11 are immediately apparent. Typically a real experiment deals with misalignment of the order of a few degrees or less.

3.2.5 Rotation around Y axis: β parameter

The fifth misalignment parameter is denoted as β and quantifies rotation along the Y axis. This case is very similar to the previous one. Based on the information presented in figure 3.13 it can be deduced that the primary indicator of a β misalignment is a linear correlation between the variables Δx and x. If these variables are plotted against each other, then the slope can be used to obtain the value of the parameter β , given that $\gamma, \alpha = 0$



Figure 3.12: Diagram that illustrates rotation around the Y axis showing a 3D layout of the coordinate system and a projection of the XZ plane.



Figure 3.13: Results of a run with n = 10000 events and a misalignment of $\beta = \pm 30^{\circ}$.

When compared to the case of perfect alignment, the hits displayed in figure 3.13 (a) are shifted closer to the center in the X axis. The shift is greater for the points further away from the center, i.e. the points with a greater x.

The scattering graphs 3.13 (b) and 3.13 (c) show a clear correlation between the variables. Applying the 3D rotation matrix [19, p167] and considering that the error in the DUT measurements is negligible, $(x, y)_{dut}^{(i)} \approx (x, y)_{true}^{(i)}$, one can derive the displacement of telescope measurements caused by a β misalignment

$$x_{telescope} = (\cos\beta\cos\gamma) x_{true} + (\sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma) y_{true}$$

if $\gamma, \alpha = 0 \implies x_{telescope} = \cos\beta \cdot x_{true} \implies$
$$\Delta x \approx \left(1 - \frac{1}{\cos\beta}\right) x_{telescope}$$

The meaning of this result is that the slope of the Δx vs. x graph is approximately equal to $1 - 1/\cos\beta$, given $\gamma, \alpha = 0$. Similar to the previous case, this expression allows the calculation of the absolute value of β , but it does not give any information about the sign of β .

3.2.6 Rotation around Z axis: γ parameter

The sixth misalignment parameter is denoted as γ and quantifies rotation along the Z axis. Based on the information presented in figure 3.15 it can be deduced that the primary indicator of a γ misalignment is a linear correlation between the variables Δy and $x_{telescope}$. If these variables are plotted against each other, then the slope can be used to obtain the sign the parameter γ .



Figure 3.14: Diagram that illustrates rotation around the Z axis showing a 3D layout of the coordinate system and a projection of the XY plane.



Figure 3.15: Results of a run with n = 10000 events and a misalignment of $\gamma = \pm 10^{\circ}$.

The scattering graphs 3.13 (b) and 3.13 (c) show a linear correlation between the variables. Applying the 3D rotation matrix [19, p167] and considering that the error in the DUT measurements is negligible, $(x, y)_{dut}^{(i)} \approx (x, y)_{true}^{(i)}$, one can derive the displacement of telescope measurements caused by a γ misalignment

 $y_{telescope} = (\sin \alpha \sin \gamma \sin \gamma + \cos \alpha \cos \gamma) y_{true} + (\cos \beta \sin \gamma) x_{true}$ $y_{telescope} - y_{true} = (\sin \alpha \sin \gamma \sin \gamma + \cos \alpha \cos \gamma - 1) y_{true} + (\cos \beta \sin \gamma) x_{true}$ $\Delta y \approx (\cos \beta \sin \gamma) x_{telescope} + f(y_{telescope})$

When studying the relationship between Δy and $x_{telescope}$ the function $f(y_{telescope})$ can be interpreted as random noise introduced into the data, due to the fact that $y_{telescope}$ and $x_{telescope}$ are independent.

The previous result implies that the slope of the Δy vs. x graph is equal to $\cos \beta \sin \gamma$. If β is unknown then the value of γ cannot be solved from just the slope of the graph. On the other hand, the angles will always be smaller than 45° and therefore it is guaranteed that the slope approaches 0 in the limit $\gamma \longrightarrow 0$ and that the sign of the slope is equal to the sign of γ . This properties allow the alignment of γ through iteration. If γ is negative then it can be increased by a small positive amount to reduce the slope, and this process can be repeated until the slope is sufficiently small or until it switches signs. On the case of a positive γ the same process must be followed but using a small negative amount instead.

3.3 Combining multiple misalignment parameters

The previous section determined that each individual parameter is connected to some relationship in the observables. For example, a δz misalignment is connected to the slope of the graphs Δx vs. θ_x and Δy vs. θ_y . In some cases the observed relationship is also dependent on other misalignment parameters. Continuing with the δz example, the slope of the graphs also depends on the parameters α , β and γ . In practice this means that we first need to set $\alpha = \beta = \gamma = 0$ in order to identify a δz misalignment, and therefore we must align the former parameters before attempting to align the latter.



Figure 3.16: Results of a run with n = 10000 events and a misalignment of $\delta x = \delta y = 5$ px, $\delta z = 30$ px, $\alpha = \beta = 30^{\circ}$, $\gamma = 10^{\circ}$.

Figure 3.16 shows a run where all the parameters have non-zero values, meaning no parameter has been aligned. There are 3 parameters that can be evaluated regardless of the value of any other parameters: γ , δx and δy . The parameters δx and δy can be determined with the histograms, and the parameter γ can be aligned using the Δy vs. $x_{telescope}$ plot.

Figure 3.17. shows a run where the 3 previously mentioned parameters (γ , δx and δy) have already been aligned. Once γ is set to 0, the slope of the Δy vs. $y_{telescope}$ plot only depends on the parameter α , enabling its alignment. Moreover, once α has been aligned (figure 3.18) we can evaluate β using the slope of the Δx vs. $x_{telescope}$. After successfully



Figure 3.17: Results of a run with n = 10000 events and a misalignment of $\delta z = 30$ px, $\alpha = \beta = 30^{\circ}$.



Figure 3.18: Results of a run with n = 10000 events and a misalignment of $\delta z = 30$ px, $\beta = 30^{\circ}$, .

aligning β we arrive at the case where there is only one parameter (δz) with a non-zero value, which was already covered in the previous section.

As a conclusion, in the case of a simulation where multiple different misalignment parameters have a non-zero value, we must follow a strict order in the alignment process, namely $\delta x, \delta y, \gamma \rightarrow \alpha \rightarrow \beta \rightarrow \delta z$. Aligning the parameters in this order allows us to simplify the problem to the case of individual misalignment parameters that we studied in the previous section.

3.4 Alignment procedure

A key reminder is that the alignment procedure does not involve physically moving the DUT. Instead the procedure is about analyzing the data to determine the value of the misalignment parameters in order to solve the real position of the DUT.

Section 3.2 determined how we can identify and correct each individual misalignment parameter, and section 3.3 determined the order in which the parameters must be aligned. With this information we can now define the exact procedure that should be followed to align the DUT.

- 1. δx and δy . The mean of the distribution that appears in the Δx and Δy histograms is a good estimation for the value of δx and δy , respectively. Using this property the two parameters can be aligned until both histograms have a mean sufficiently close to 0.
- 2. γ . Plot Δy against $x_{telescope}$ and use a linear regression to determine the slope's value. The sign of this slope coincides with the sign of γ . If γ is negative, then increase it by a small positive amount to reduce the slope. If γ is negative do the opposite. Iterate through this process until the slope is sufficiently close to 0.
- 3. α . Obtain the slope of the Δy vs. $y_{telescope}$ plot. Increase α by a small positive amount and recalculate the slope. Assuming this amount is small enough, an increase in the slope's value indicates that α is positive, and vice-versa. Once the sign of α is determined, its value can be approximated with the expression $m \approx (1 1/\cos \alpha)$, where m denotes the slope. Iterate through this process until the slope is sufficiently close to 0.
- 4. β . Use the same process from the previous step, but using the plot Δx vs. $x_{telescope}$ and the expression $m \approx (1 1/\cos\beta)$, instead.
- 5. δz . Both the Δx vs. θ_x and the Δy vs. θ_y plots can be used to align this parameter. Obtain the slope of either plot and use $m \approx -\delta z$ to estimate the value of the parameter. Iterate until the slope of both plots is sufficiently close to 0.

Chapter 4 Beyond the Toy Model

This chapter will introduce new complications that expand the toy model with the aim to make the simulation more closely resemble the real world experiment that it is trying to model.

4.1 Gaussian distribution



Figure 4.1: Plot showing the histograms of a random sample of n = 10000 events for the variables θ_x , θ_y , x_{true} and y_{true} obtained in a test run of the simulation once the Gaussian distributions have been introduced.

As previously stated, the simulation uses the Monte Carlo method to simulate the track data. In the toy model, the variables x_{true} , y_{true} , θ_x and θ_y are drawn from a uniform probability distribution. This is the simplest case and also the one that leads to the cleanest correlation plots, but it is not likely that a real test beam experiment would ever use a squared uniform particle beam. On the contrary, it is common to have a test beam experiment with a Gaussian particle beam, which is what makes it more realistic.

The boundaries used for the uniform distributions were $\theta_x, \theta_y \in [-0.5^\circ, +0.5^\circ]$ and $x_{truye}, y_{true} \in [-10\text{px}, +10\text{px}]$. To define a Gaussian distribution the standard deviation must be specified, not the boundaries. The chosen values for the standard deviation are $\sigma(\theta_x) = \sigma(\theta_y) = 0.125^\circ$ and $\sigma(x_{true}) = \sigma(y_{true}) = 2.5\text{px}$. This way any value has a probability of 4σ (99.994%) to fall inside of the boundaries of the previous uniform distributions.

The three plots in the top row of figure 4.2 show the true position as well as the position measured by the telescope and DUT for the first n = 1000 events. Everything that was



Figure 4.2: Results of a run with n = 10000 events and a misalignment of $\delta x = \delta y = 5$ px, $\delta z = 30$ px, $\alpha = \beta = 30^{\circ}$, $\gamma = 10^{\circ}$, using a Gaussian distribution to sample the track data. The standard deviation of the Δx and Δy histograms is 0.539 and 0.558px, respectively.

mentioned in the analysis of chapter 3, such as the position of the DUT measurements and how they relate to the true position, still apply here. The fundamental difference lies in the shape of the data points, which now conform to the characteristics of a Gaussian distribution rather than the square pattern seen in the toy model. The change in the probability distribution also introduces outliers. In contrast to the previous scenario where data points were confined within strict boundaries, they can now assume any position. However, it's important to note that data points located far from the center have a significantly lower probability of occurrence.

The rest of the graphs of figure 4.2 are the histograms and scattering graphs that have been used throughout the previous chapter. It is interesting to compare these graphs with the ones from figure 3.16, because both represent the case with $\delta x = \delta y = 5 \text{px}$, $\delta z = 30 \text{px}$, $\alpha = \beta = 30^{\circ}$ and $\gamma = 10^{\circ}$, the only difference being the use of the Gaussian distribution. All the patterns still appear in the new graphs. For example, the histograms still have a mean that is a highly accurate approximation of the parameters δx and δy , and the positive slope of the Δy vs. $x_{telescope}$ plot again indicates a positive value of γ . The main difference is that the histograms have a smaller deviation and that the points in the scattering graphs are concentrated closer to the mean value. This difference makes it harder to identify the patterns, which is why all the in depth analysis of these patterns was done using the toy model with the uniform distribution.

As a conclusion, all the analysis from the previous chapter, including the optimal alignment procedure, still applies in this case. Using a Gaussian distribution changes the simulation results in a way that makes them more closely resemble real world data, but at the same time it blurs the patterns that were clearly visible in the previously used toy model.

4.2 Charge sharing

As stated in the first chapter, this project focuses on a specific type of DUT: a pixelated silicon detector. The working principle behind this kind of device is the interaction of the passing particles with the semiconductor material, which generates charge carrier pairs that can be measured. The position of the particle can be solved knowing which pixel measured the charge carriers. Previously the supposition was that a single particle only activated one pixel at a time, the pixel which center was closer to the particles position, as discussed on detail in section 2.5.

In reality, a particle only activates a single pixel if the particle crosses the sensor perpendicularly and away from the pixel edges. In any other case, more than one pixel measures charge carriers, and the position of the particle must be reconstructed using the measurements from multiple pixels. Therefore, the position of the particle as measured by the device is a function of the position of the activated pixels and the amount of charge measured on each one. The closer the particle passes to the center of a pixel, the more charge it will measure.

The number of pixels activated by a single event is denoted as *cluster size*. *Charge sharing* occurs whenever the cluster size is higher than 1, because the charge generated by the passing of the particle will be shared between more than one pixel.

To facilitate charge sharing modeling one can assume that when a particle strikes the detector, it generates charge carriers within a radius of 0.25px from the impact point, and that the density of generated charge carriers is constant throughout that radius. The

previous models did not implement any mechanism to model the amount of charge created by each particle, but they do calculate the point where the particle hits the DUT, $(x, y)_{real}^{(i)}$. Taking advantage of the aforementioned assumptions, this point can be used as a staring point to mimic the charge measured by each pixel. Finally, it is important to note that the assumptions are closely based on the paper [20], which studies the relationship between the track's position and the charge generation in the detector.

There are many possible methods that can be used to reconstruct the $(x, y)_{dut}$ position, but we decided to use a Center of Gravity (CoG) method [21]. Let the area of a circle with radius 0.5px centered at $(x, y)_{real}^{(i)}$ represent the total charge deposited by the particle, and the charge measured by each pixel equal the fraction of the area that falls into each pixel. The position measured by the DUT can be estimated as the average of the position of all the pixels, weighted by the fraction of the area of the circle that falls into each pixel:

$$x_{dut} = \sum_{j} x^{j} c_{j} \tag{4.1}$$

$$y_{dut} = \sum_{j} y^{j} c_{j} \tag{4.2}$$

where $(x, y)^j$ is the position of the j-th pixel and c_j is the fraction of the area that falls inside it.



Figure 4.3: Graph that illustrates the process followed to simulate charge sharing. First the true track position, $(x, y)_{true}^{(i)}$, is computed. Then a circle centered at that position with a radius of 0.25px is drawn, and the fraction of its area that falls into each pixel, c_j , is calculated. Finally the position measured by the DUT, $(x, y)_{dut}^{(i)}$, is computed using equations 4.1 and 4.2.

Figure 4.3 shows 4 events. The values of the track position, $(x, y)_{true}^{(i)}$, were handpicked so that each event represent one of the possible cluster sizes. The top right point illustrates the case of a size one cluster. The circle falls completely within the central pixel, and the position measured by the DUT is simply the center of that pixel. This result is identical to the one that would be obtained using the toy model before implementing charge sharing.

The top left. bottom left and bottom right points correspond to size 2, 3 and 4 clusters, respectively. The track position measured by the DUT is closer than the pixel center to the true track position, which shows that the use of charge sharing allows more precise DUT measurements, resulting in smaller residuals. This can be exemplified by comparing the histograms from the model with cluster sharing (fig. 4.5) against a simulation without it (4.2). The introduction of cluster sharing has reduced the standard deviation by around 10% for both the Δx and Δy histograms.

The only problem remaining is how to compute equations 4.1 and 4.2, specifically the term c_j . This could be achieved by approximating the required integrals with some numerical method or simply using Monte Carlo. However, when experimenting with this methods in the simulation, they proved to be very inefficient, slowing down the execution time by 2 or 3 orders of magnitude.

Ideally this problem could be solved simply by finding a function that calculates c_j for every pixel based on the true position of the track: $\vec{c} = (c_1, c_2, ..., c_n) = f(x_{true}^{(i)}, y_{true}^{(i)})$. However this would involve finding a universal formula for the overlap area of a circle and a rectangle in \mathbb{R}^2 . There are many books [22, p93] [23, p244] [24, p312] and online resources [25] [26] [27] that discuss this problem on one form or another. However, even after an intensive research we could not find the derivation of a universal formula for the overlap of a circle an rectangle.

Even though none of this sources contained the desired formula, one of them [25] proved to be very useful. It derives the expression for the intersection area A between the unit circle, i.e. a circle with a radius of 1 centered at (0,0), and the half plane $(-\infty, u) \times$ $(-\infty, \infty)$

$$A = f(u) = \begin{cases} \pi, & u \in [1, \infty) \\ \pi - \arccos(u) + u\sqrt{1 - u^2}, & u \in (-1, +1) \\ 0, & u \in (-\infty, -1] \end{cases}$$
(4.3)

Starting from equation 4.3 through a process of trial and error, we were able to find the function $f: f(x_{true}^{(i)}, y_{true}^{(i)}) = (x, y)_{dut}^{(i)}$ that successfully computes the DUT measurements from the true track positions. If the track position is in the range $x_{true}, y_{true} \in [0, 1)$, then the function is

$$f(q_{true}) = \begin{cases} h(q_{true}) + 0.5 & q \in [0, 0.25) \\ 0.5, & q \in [0.25, 0.75) \\ h(q_{true} - 1) + 1.5 & q \in [0.75, 1) \end{cases}$$
(4.4)

where q = x, y and h(q) is

$$h(q) = -\frac{16}{\pi} \left(R^2 \arccos(q/R) - q\sqrt{2R(R+q) - (R+q)^2} \right)^2$$

where R = 0.25 px is the radius of the circle. Using a modern computing library like NumPy, generalizing this function to the full range of x_{true}, y_{true} is trivial.

Figure 4.4 shows that the results using Monte Carlo and numerical integration approach the function 4.4 when the former computes more events or when the latter uses more bins. Apart from this, multiple other test were performed, such as plotting y_{dut} vs. y_{true} , and analyzing the data from figure 4.5, to compare the function f(q) from equation 4.4 to make sure that it leads to the same results as the other methods.



Figure 4.4: x_{dut} vs. x_{true} . Starting with the true track position, x_{true} , Monte Carlo and numerical integration are used to estimate the fraction of the circle's area that falls on each pixel c_j , which is then used to obtain x_{dut} for that particular event. This is repeated for n = 10000 events and compared to the results of using the function f(q) from equation 4.4.

Figure 4.5 show the results after charge sharing has been implemented in the simulation. The misalignment parameters and the number of events are the same as the ones used in figures 3.16 and 4.2, which display the results of the toy model before and after implementing Gaussian distribution to sample the track data. The parameters are kept constant among these three runs so that they can be compared with each other to study the effects of adding more complexity to the simulation.

Looking at figure 4.5, there are a few differences that are visible at first glance. Firstly, the top right plot that shows the position measured by the DUT for all the events, $(x, y)_{dut}^{(i)}$, does not only have points at the pixel center. Instead, a position measurement can fall anywhere within a certain pixel, as a direct result of implementing charge sharing. This fact also has an effect on the Δx vs $y_{telescope}$ and Δy vs $x_{telescope}$ plots, where the data points do not always fall inside diagonal lines, as they did in figures 3.16 and 4.2.

In size one clusters the DUT measurement is always the center of the pixel. This means that multiple $(x, y)_{true}$ points are mapped to a single $(x, y)_{dut}$ point, which leads to a correlation between $x_{dut} - x_{true}$ and x_{true} inside an individual pixel. The lines of the Δx vs $y_{telescope}$ and Δy vs $x_{telescope}$ plots are a consequence of this correlation. Before charge sharing was implemented all the measurements came from size one clusters, and therefore all the points in these plots fell in diagonal lines. After charge sharing was implemented, size one cluster are still present, which is why this lines still appear, but the points coming from size 2,3 and 4 do not follow within this lines, making them harder to see when looking at the hole data set.

If the proportion of size one clusters or the number of events decreases, the number of pixels increases, or some noise due to measurement errors is introduced into the data, then the pattern of diagonal lines in the plots disappears completely. This indicates that the pattern will not be visible in real world data, which is confirmed with the experimental results of the experiment [2].



Figure 4.5: Results of a run after implementing charge sharing into the simulation. The run has n = 10000 events and a misalignment of $\delta x = \delta y = 5$ px, $\delta z = 30$ px, $\alpha = \beta = 30^{\circ}$, $\gamma = 10^{\circ}$. The model still uses a Gaussian distribution to sample the track data. The standard deviation of the Δx and Δy histograms is 0.484 and 0.500, respectively.

Another key insight from figure 4.5 is that the Δx and Δy distributions are slightly narrower, and therefore the scattering plots have less variance in the vertical axis. As explained before, this is due to the fact that the introduction of charge sharing in the simulation leads to more precise measurements by the DUT.

Apart from the differences highlighted earlier, the results still show the same patterns. The misalignment can still be solved using the procedure described in Section 3.4, and all the analysis from the preceding chapter, which correlated each specific misalignment parameter with a crucial indicator in the simulation data, remain applicable to this more intricate model.

Chapter 5

Conclusion

This project has extensively studied the alignment of the DUT in test beam experiments through a Monte Carlo simulation. The toy model, the simplest possible model that allows the simulation of the misalignment effects, was posed and implemented following the methodology outlined in the second chapter. Chapter 3 covered all the data analysis of the results for this model, which is the bulk of this project. First, it focused on the case of perfect alignment, which allowed the study of the effect caused by the intrinsic error of the DUT measurements, which appear independently of alignment. Then, it was determined that there are 6 degrees of freedom in which the DUT can be misaligned with respect to the telescope, and consequently 6 misalignment parameters were defined to characterize every possible state.

Subsequently, this project covered the case of individual misalignment parameters, where only one parameter has a non-zero value. This allowed the establishment of a link between each parameter and a distinct indicator that appears in the simulation results whenever the parameter has a non-zero value. For example, a non zero mean of the residuals Δx indicates a misalignment on the δx parameter. These indicators enabled a way to infer the value of each parameter based on the results, which allows the alignment of that parameter.

The next step was to put multiple parameters together to study the emergent effects. This helped to determine which indicators require the previous alignment another parameter to be accurate. For example, the linear fit of Δy vs. $y_{telescope}$ is an accurate indicator of the parameter α only once the misalignment in γ has already been corrected. This analysis allowed us to determine which order must be followed in order to deconvolute the data in a case where multiple misalignments have a non-zero value.

Putting everything together, we were able to infer the optimal alignment procedure that enables the alignment of any state, regardless of the values of any parameters. This procedure is explained in detail in Section 3.4.

Once this procedure was deduced and all the analysis of the results of the toy model was exhausted we decided to move on to more complex model with the aim to more accurately replicate the real world experiments. With this goal, additional complications were added to the simulation. First, a Gaussian was used to sample the track data, which is more realistic than the uniform distribution used in the toy model. Additionally, we found a way to implement cluster sharing so that the simulation of detector measurements is more realistic. After implementing this complications in the simulation it becomes apparent that, while this effort makes the simulation results more closely resemble experimental data, the underlying indicators are the same as they were in the toy model and that the previous alignment procedure still applies to the more intricate models. Critically, the indicators and relevant patterns appear most clearly in the simple toy model. The complications are useful to check that the simulation is accurate to the real experiment that it is trying to model, however the toy model is the most useful way to study the misalignment effects on their purest form, without any added noise.

As mentioned in Section 2.6, the simulation was implemented in Python. The full source code [18] is published on GitHub [28], the most common tool used to share repositories , with the address https://github.com/PerezSergioTFG/test-beam-alignment-simulation. Alternatively, the repository [18] can also be found in Zenodo [29], a publishing platform that allows the long-term storage and academic citation of code repositories. From both of these sources anyone can access the full script, and we authorize and encourage that the reader downloads it to be able to run this simulation locally whenever they wish to do so. The only requirements needed to run the script are to install Python and all the relevant libraries mentioned in Section 2.6. Alternatively it can also be run online using a tool like Jupyter Notebooks [30].

The user can customize the simulation through a Command Line Interface (CLI). If one where to run the script as it is, it would display the results of a run of the toy model with perfect alignment for n = 1000 events. The CLI allows the user to change all the alignment parameters and the number of events, and it is also possible to choose if they want to add the complications that implement charge sharing and a Gaussian distribution of the track data. To get more information about how to run the simulation, visit the repositories address and read the README.md file or type --help on the CLI.

There are a few potential improvements of this project. Firstly, the development of a Graphical User Interface (GUI) as an alternative to the CLI would make the simulation more accessible for researchers and students that are not familiar with the command line. Furthermore, the GUI could also be used to visualize the geometry of the problem and the possible misalignments.

Additionally, it may also be useful to develop a tool that accepts real world data of a test beam experiment and outputs the predicted values of the misalignment parameters, which could help to accelerate the alignment process in real world experiments. Moreover, it is possible that the alignment process described in Section 3.4 could be optimized further. The extensive analysis of the third chapter does not leave much room for improvement apart from 2 distinct issues: the misalignment parameters must be solved one at a time, and the parameter γ cannot be estimated analytically from the slope of any graph. Machine Learning (ML) has proven to be very efficient for dealing with complex regression problems like these ones, and therefore it is possible that some sort of ML model is capable of solving or mitigating this issues.

Bibliography

- [1] J. Duarte, "Pixelated 3d sensors for tracking in radiation harsh environments," 2020.
- [2] M. Roa and J.Duarte, "Response characterization of 3d pixel particle sensors subjected to extreme radiation conditions," 2021.
- [3] I. Rubinskiy, "An eudet/aida pixel beam telescope for detector development," *Physics Proceedia*, vol. 37, pp. 923–931, 2012. Proceedings of the 2nd International Conference on Technology and Instrumentation in Particle Physics (TIPP 2011).
- [4] E. Gschwendtner, "Overview on cern test beam facilities and plans for tests for noncollider experiments," 2009.
- [5] M. B. et al., "Performance of alice ad modules in the cern ps test beam," Journal of Instrumentation, vol. 16, p. P01017, jan 2021.
- [6] T. M. et al., "Calibration of lhcf calorimeters for photon measurement by cern sps test beam," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 671, pp. 129–136, 2012.
- [7] K. Sjobak, E. Adli, M. Bergamaschi, S. Burger, R. Corsini, A. Curcio, S. Curt, S. Döbert, W. Farabolini, D. Gamba, *et al.*, "Status of the clear electron beam user facility at cern," *Proc. IPAC'19*, pp. 983–986, 2019.
- [8] R. Diener, J. Dreyling-Eschweiler, H. Ehrlichmann, I.-M. Gregor, U. Kötz, U. Krämer, N. Meyners, N. Potylitsina-Kube, A. Schütz, P. Schütze, et al., "The desy ii test beam facility," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 922, pp. 265–286, 2019.
- [9] G. Apollinari, I. Béjar Alonso, O. Brüning, P. Fessia, M. Lamont, L. Rossi, and L. Tavian, "High-luminosity large hadron collider (hl-lhc). technical design report v. 0.1," 2017.
- [10] "Milepede ii documentation." https://www.desy.de/~kleinwrt/MP2/doc/html/ index.html, 2023.
- [11] Geant4Collaboration, "Geant4 user's guide for toolkit developers." https: //geant4-userdoc.web.cern.ch/UsersGuides/ForToolkitDeveloper/fo/ BookForToolkitDevelopers.pdf, 2023.
- [12] J. VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, 2016.

- [13] A. Martelli, Python in a Nutshell: A Desktop Quick Reference. In a Nutshell (O'Reilly), O'Reilly Media, 2006.
- [14] M. Gorelick and I. Ozsvald, High Performance Python: Practical Performant Programming for Humans. O'Reilly Media, 2014.
- [15] A. N. Ziogas, T. Ben-Nun, T. Schneider, and T. Hoefler, "Npbench: A benchmarking suite for high-performance numpy," in *Proceedings of the ACM International Confer*ence on Supercomputing, pp. 63–74, 2021.
- [16] W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media, 2017.
- [17] Pallet, "Click documentation." https://click.palletsprojects.com/, 2023.
- [18] D. J. Pérez Sergio, "Monte Carlo simulation of alignment effects in test beam experiments," Aug. 2023.
- [19] G. Strang, *Linear Algebra and Its Applications*. Academic Press, 1976.
- [20] E. Belau, R. Klanner, G. Lutz, E. Neugebauer, H. Seebrunner, A. Wylie, T. Böhringer, L. Hubbeling, P. Weilhammer, J. Kemmer, U. Kötz, and M. Riebesell, "Charge collection in silicon strip detectors," *Nuclear Instruments and Methods in Physics Research*, vol. 214, no. 2, pp. 253–260, 1983.
- [21] J. Ladell, W. Parrish, and J. Taylor, "Center-of-gravity method of precision lattice parameter determination," Acta Crystallographica, vol. 12, pp. 253–254, Mar 1959.
- [22] G. Chauhan, NDA/NA SSB INTERVIEW GUIDE: Bestseller Book by Gautam Chauhan: NDA/NA SSB INTERVIEW GUIDE. Prabhat Prakashan, 2021.
- [23] C. Van Loan and K. Fan, Insight Through Computing: A MATLAB Introduction to Computational Science and Engineering. Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2010.
- [24] O. Gervasi, V. Kumar, C. Tan, D. Taniar, A. Laganà, Y. Mun, and H. Choo, Computational Science and Its Applications - ICCSA 2006: International Conference, Glasgow, UK, May 8-11, 2006, Proceedings, Part II. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006.
- [25] "Math exchange: Circle areas on squared grid." https://math.stackexchange.com/ questions/823846/circle-areas-on-squared-grid, 2014.
- [26] "Stack overflow: Area of intersection between circle and rectangle." https://stackoverflow.com/questions/622287/ area-of-intersection-between-circle-and-rectangle, 2017.
- [27] "Leet code: Circle and rectangle overlapping." https://leetcode.com/problems/ circle-and-rectangle-overlapping/, 2023.
- [28] "Github official webiste." https://github.com/, 2023.

- [29] "Zenodo official webiste." https://zenodo.org/, 2023.
- [30] "Jupyter notebook official webiste." https://jupyter.org/, 2023.
- [31] MatplotlibDevelopmentTeam, "Matplotlib documentation." https://matplotlib. org/3.7.2, 2023.

List of Figures

1.1	Telecope diagram
1.2	Set up of a real test beam experiment
1.3	Example of DUT misalignment
2.1	Geometry
2.2	Variables definition
2.3	Simulating DUT measurements
2.4	Uniform distributions
3.1	Perfect alignment hits
3.2	Perfect alignment plots
3.3	Translation along X axis
3.4	Hits for a non-zero δx parameter $\ldots \ldots \ldots$
3.5	Plots for a non-zero δx parameter
3.6	Translation along Y axis $\ldots \ldots \ldots$
3.7	Plots for a non-zero δu parameter
3.8	Translation along Z axis $\ldots \ldots 19$
3.9	Plots for a non-zero δz parameter
3.10	Rotation around X axis $\ldots \ldots 20$
3.11	Plots for a non-zero α parameter $\ldots \ldots \ldots$
3.12	Rotation around Y axis
3.13	Plots for a non-zero β parameter $\ldots \ldots 22$
3.14	Rotation around Z axis
3.15	Plots for a non-zero γ parameter $\ldots \ldots 23$
3.16	Combining the parameters δx , δy , δz , α , β , and γ , \ldots , \ldots , 24
3.17	Combining the parameters δz , α and β , \ldots \ldots \ldots \ldots 25
3.18	Combining the parameters δz , and β ,
4 1	Caussian distributions
4.1	Gaussian distributions 27
4.2	Circulation DUT managements with change chaving
4.5	Simulating DUT measurements with charge sharing
4.4	x_{dut} vs. x_{true} with cluster sharing
4.0	Simulation results after implementing charge sharing