

This is the Accepted Manuscript version of an article accepted for publication in Journal of Instrumentation. IOP Publishing Ltd is not responsible for any errors or omissions in this version of the manuscript or any version derived from it. The Version of Record is available online at <https://doi.org/10.1088/1748-0221/15/09/P09020>.

arXiv:2011.10356v1 [physics.ins-det] 20 Nov 2020

# EUTelescope: A modular reconstruction framework for beam telescope data

---

**T. Bisanz<sup>a,1</sup> H. Jansen<sup>b,2</sup> J.-H. Arling<sup>b</sup> A. Bulgheroni<sup>c,3</sup> J. Dreyling-Eschweiler<sup>b,4</sup>  
T. Eichhorn<sup>b,5</sup> I. M. Gregor<sup>b</sup> P. Hamnett<sup>b,6</sup> C. Kleinwort<sup>b</sup> A. Morton<sup>b,6</sup> H. Perrey<sup>b,7</sup>  
M. Queitsch-Maitland<sup>b,8</sup> E. Rossi<sup>b</sup> S. Spannagel<sup>b</sup>**

<sup>a</sup>*Georg August Universität Göttingen, II. Physikalisches Institut, Friedrich Hund Platz 1, Göttingen, 37077 Germany*

<sup>b</sup>*Deutsches Elektronen-Synchrotron, Notkestr. 85, 22607 Hamburg, Germany*

<sup>c</sup>*Università degli Studi dell'Insubria, Via Valleggio 11, 22100 Como, Italy*

*E-mail:* [tobias.bisanz@cern.ch](mailto:tobias.bisanz@cern.ch), [hendrik.jansen@desy.de](mailto:hendrik.jansen@desy.de)

**ABSTRACT:** EUTELESCOPE is a modular, comprehensive software framework for the reconstruction of particle trajectories recorded with beam telescopes. Its modularity allows for a flexible usage of processors each fulfilling separate tasks of the reconstruction chain such as clustering, alignment and track fitting. The framework facilitates the usage of any position sensitive device for both the beam telescope sensors as well as the device under test and supports a wide range of geometric arrangements of the sensors.

In this work, the functionality of the EUTELESCOPE framework as released in v2.2 and its underlying dependencies are discussed. Various use cases with emphasis on the General Broken Lines advanced track fitting methods give examples of the work flow and capabilities of the framework.

**KEYWORDS:** Software architectures; Analysis and statistical methods; Data processing methods; Pattern recognition, Cluster finding, Calibration and fitting methods

---

<sup>1</sup>Corresponding author, now at CERN, Geneva, Switzerland

<sup>2</sup>Corresponding author

<sup>3</sup>now at European Commission Joint Research Centre, Karlsruhe, Germany

<sup>4</sup>now at Silpion IT-Solution GmbH, Hamburg, Germany

<sup>5</sup>now at PANDA GmbH, Hamburg, Germany

<sup>6</sup>now self-employed

<sup>7</sup>now at Division of Nuclear Physics, Lund University, Lund, Sweden

<sup>8</sup>now at CERN, Geneva, Switzerland

---

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Framework architecture</b>	<b>3</b>
2.1	IO-framework and data interface	3
2.2	MARLIN	4
2.3	Geometry	5
2.4	Further external packages	6
2.4.1	MillepedeII	6
2.4.2	GBL	7
<b>3</b>	<b>Reconstruction flow</b>	<b>7</b>
3.1	Raw data conversion	8
3.2	Noisy pixel treatment	8
3.3	Clustering	9
3.4	Hit position derivation	9
3.5	Alignment	10
3.6	Track fit and analysis	10
<b>4</b>	<b>General Broken Lines processor in EUTELESCOPE</b>	<b>10</b>
4.1	Track finding	11
4.2	GBL track fitting	11
<b>5</b>	<b>Reconstruction examples</b>	<b>13</b>
5.1	Empty beam telescope analysis	13
5.1.1	Raw data conversion and noisy pixel treatment	14
5.1.2	Clustering	14
5.1.3	Hitmaker and pre-alignment	15
5.1.4	Alignment	15
5.1.5	Track fit	17
5.2	Passive scatterer analysis	17
5.3	Analysis including the ALiBaVa system	18
5.4	Analysis including a strip detector and a FE-I4 reference detector	20
<b>6</b>	<b>Conclusion</b>	<b>22</b>

---

# 1 Introduction

High-precision tracking devices used at charged particle beam lines are known as *beam telescopes* and present vital tools for the R&D of position-sensitive particle detectors. Beam telescopes provide spatially well-resolved particle trajectories, or tracks, used e.g. for the investigation of novel sensor technologies or detector module testing. Their wide range of use cases include LHC and BelleII detector upgrade programs, detector R&D for future experiments like CLIC, ILC, Mu3e and SHiP as well as generic sensor R&D and outreach programs. EUTELESCOPE [1] is a modular and comprehensive software framework used to reconstruct and analyse the data recorded with such instruments.

The analysis of data recorded at test beam campaigns usually requires the conversion to a standardised data format, the synchronisation of the data, the positional alignment of the detectors based on preliminary tracks and the description of the final trajectories. EUTELESCOPE allows to execute all such tasks in a streamlined manner. Many available custom reconstruction frameworks such as Judith [2, 3], Proteus [4] or TBSW [5] execute similar tasks, but are designed to serve specific use cases in terms of geometry, sensor choice and analysis flow. On the contrary, EUTELESCOPE and Corryvreckan [6] follow a modular, flexible and generic approach covering a broader range of use cases.

Originally developed within the EUDET<sup>1</sup> and the AIDA<sup>2</sup> frameworks and its extensive usage for beam telescopes constructed therein [7], the modular design of EUTELESCOPE also allows for analysis of data acquired with other beam telescopes [8]. Data from any position-sensitive device under test (DUT) can be integrated into the framework for a wide range of geometric arrangements.

Several technical reports on EUTELESCOPE have been published [9–12] over the last decade. One major change in the framework, with respect to the previous publications, is the migration to a new alignment scheme. This scheme is based on incrementing the alignment constants in a human-readable alignment file, moving away from alignment databases stored in a binary format. These databases were applied in a subsequent fashion, i.e. each alignment database held small corrections to the previous step. An additional big change is the implementation of the General Broken Lines (GBL) algorithm for track fitting. The GBL algorithm has become the standard approach for track reconstruction, not only for the final track fit which provides the tracks for analyses, but also in the alignment procedure which is an essential step in a test beam framework, given the short lived nature of beam test campaigns. Moreover, many parts of EUTELESCOPE have been refactored, using newer versions of the utilised libraries and modern programming language features. Due to their technical nature, these modifications are not described in this paper.

The paper describes the reconstruction framework as released in version 2.2 and is structured as follows: section 2 describes the architecture of the framework and a typical reconstruction flow is detailed in section 3. The main processor for track finding and track fitting is discussed in section 4, followed by application examples in section 5.

---

<sup>1</sup>EUDET: The *Detector R&D towards the International Linear Collider* infrastructure initiative funded by the European Union.

<sup>2</sup>AIDA: The *Advanced European Infrastructures for Detectors at Accelerators* programme co-funded by the European Union FP7 Research Infrastructures programme

## 2 Framework architecture

EUTELESCOPE is designed to be a modular, comprehensive and versatile framework and is comprised of a set of independent MARLIN<sup>3</sup> [13] processors written in C++. MARLIN processors, each one reflecting a certain step in the reconstruction chain, act upon a combination of detector data in the LCIO<sup>4</sup> format and a set of databases (GEAR, LCIO, configuration files). By this factorisation of tasks into individual and subsequently executed processors, code duplication is efficiently avoided. LCIO [14, 15] serves as the underlying persistence framework and event data model, storing derivatives of the detector data in so-called *lcio*<sup>5</sup> collections for each event. An event is defined by the entirety of data belonging to a physical particle passage that triggered a read-out of the sensors, packed in a numbered data container. A number of consecutive events are grouped into runs, representing a data-taking period with unchanged detector configurations. EUTELESCOPE uses GEAR [16] for description of the geometrical telescope set-up. External libraries such as the Eigen3 [17] linear algebra library, the MillepedeII [18, 19] package for detector alignment, the General Broken Lines [20–22] track fitting algorithm as well as the ROOT [23] framework are used for additional functionality. Furthermore, custom EUTELESCOPE extensions to GEAR allow for more complex sensor layouts deviating from a regular, equidistant and rectangular pixel grid. The ILCINSTALL package [24] is used in the EUTELESCOPE installation process [25, 26] and optionally provides all necessary dependencies in a self-contained location.

### 2.1 IO-framework and data interface

MARLIN, and thus EUTELESCOPE, use the LCIO persistence framework to store as-recorded as well as processed data. LCIO provides a C++ implementation with a well-documented application programming interface (API). Further implementations in Java as well as a Fortran and a Python interface exist, allowing users to easily process LCIO data in their external programs. This is of special interest for simulation tool-kits, for storing data for subsequent reconstruction in EUTELESCOPE, or for dedicated analysis frameworks operating on reconstructed data.

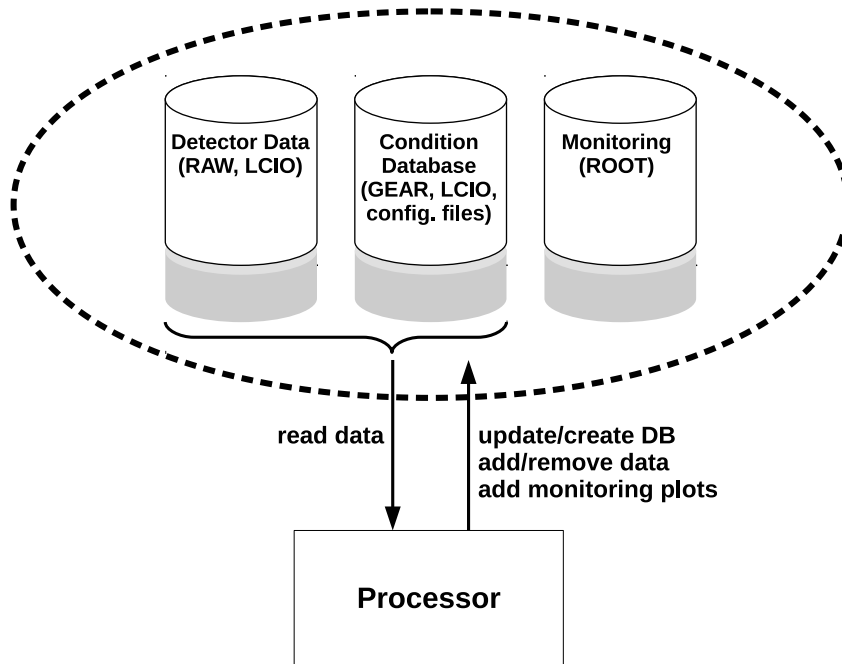
The LCIO format is event-based, with the data associated to a trigger belonging to a given event. Alternatively, an event can be defined as the data belonging to a certain range in time. Exactly one LCIO event, each comprised of a container with data belonging to the event, is accessible at a time. For every event an LCIO file contains an arbitrary amount of LCIO collections. Each collection can store an arbitrary amount of objects. Raw pixel indices, clustered pixels or derived hit positions are examples of such objects. Objects can link to other objects within the same event, e.g. a hit object can link to the pixel hits which were used to derive this hit. This link is important as it allows to trace a hit, i.e. a derived object, back to its original data rendering. For example, from the links between the cluster, the derived hit, and fitted track, an analysis of residual distribution depending on the cluster size is easily possible. Raw data can be converted to the LCIO format using converter plug-ins from the EUDAQ data acquisition framework [27], which requires to link against the EUDAQ library. Conversion can also be performed outside the EUTELESCOPE framework, e.g. by using the EUTELESCOPE library in combination with an interfacing tool to write LCIO data.

---

<sup>3</sup>Marlin: Modular Analysis and Reconstruction for the Linear Collider

<sup>4</sup>GEAR: Geometry API for Reconstruction

<sup>5</sup>LCIO: The Linear Collider Input/Output framework



**Figure 1.** The operating principle of a single processor in EUTELESCOPE with its inputs and outputs.

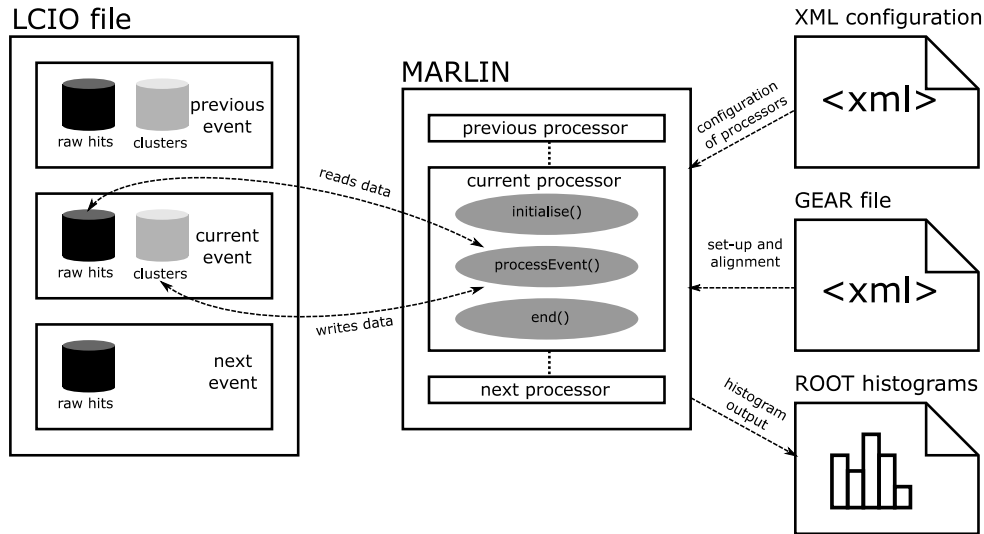
It is straightforward for users to implement their own data types for pixel hits as well as clusters derived from these types in EUTELESCOPE. This is necessary in order to accommodate for particularities of the read-out systems, which differ e.g. in terms of timing capabilities or data output formats.

Data to be stored in persistent memory need to be serialised, i.e. complex objects need to be transformed into a binary stream which can be saved on disk. Object serialisation is performed by LCIO and EUTELESCOPE provides functionality to easily interface LCIO data, hiding much of the workload from the user and providing specific EUTELESCOPE classes which can be stored in the LCIO format. Different pixel types, for example, have a different footprint in the LCIO file. However in EUTELESCOPE, by using C++ templating, an interfacing object can be instantiated by the user to retrieve the pixel from the LCIO collection by calling the same function for every object.

## 2.2 MARLIN

MARLIN is the back-end component providing the entry point for execution of EUTELESCOPE processors and is responsible for loading the LCIO and GEAR files, making them available to the processors. MARLIN itself is configured using an XML steering file in which the order of processors to be executed and their parameters are specified. A single processor operates on a set of input data and provides new or modified output data. It creates or updates a database, adds or removes collections from the detector data and produces monitoring plots for visualisation and verification. The working principle of a single processor is shown in figure 1.

Subsequent processors can then operate on the resulting output. This scheme is depicted



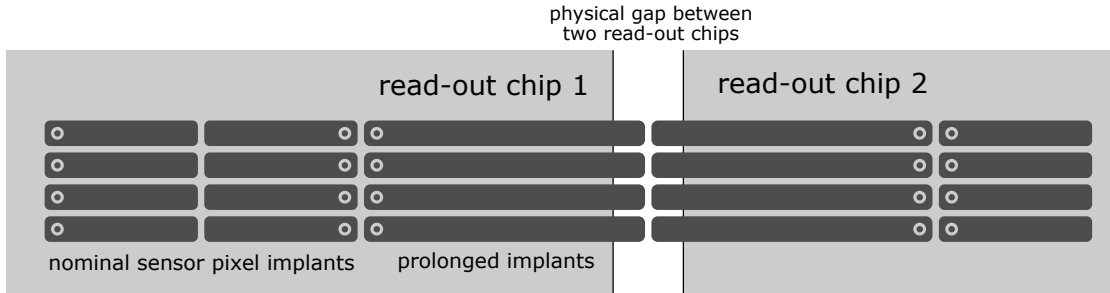
**Figure 2.** Interplay of MARLIN with the LCIO I/O framework. The event-based data structure with its LCIO collections (raw hits, clusters, ...) is depicted, as well as the principle of sequenced execution of processors.

in figure 2, where the interaction between a MARLIN processor and LCIO data is indicated. Three main methods are called during execution of a processor: *initialise()*, *processEvent()* and *end()*. The first method, the initialisation, allows to set variables and create monitoring and analysis objects before processing the first event. The *processEvent* routine is called by MARLIN for each event, carries out the analysis or reconstruction step and publishes its computed data to subsequent processors. The final method, the *end* routine, allows to execute analyses over the entire data set, as well as carry out deallocation actions like closing files or releasing memory after all events have been processed. Not depicted are routines which can be executed at the beginning and at the end of a run. These steps are very similar to the ones at the beginning and end of the EUTELESCOPE execution, but are useful for e.g. intermediate computations or resetting counters if multiple runs are processed in a concatenated batch, allowing to create collections or produce results based on various runs.

The EUTELESCOPE event model is driven by the LCIO data model as well as the usage of LCIO by MARLIN. Via MARLIN each processor can access the data associated with a single event at a time. However, it is not foreseen to access data in the previous or next event, i.e. the data in one event is assumed to be uncorrelated to adjacent events. This has to be guaranteed by the data acquisition (DAQ) system and event building framework. DAQ architectures in which data from one event is (partially) copied into another event are thus not natively compatible with MARLIN and hence EUTELESCOPE. In these cases, additional tools are required to ensure that adjacent events are uncorrelated.

### 2.3 Geometry

MARLIN uses the GEAR framework as a back-end defining an abstract interface for the geometric description of the beam telescope (materials, thicknesses, read-out pitch sizes, channel numbers, etc.). The initial telescope set-up, i.e. the positions, alignment constants and properties of the



**Figure 3.** Example of a sensor layout with differing pixel pitches in a single sensor. Such a layout is used when multiple read-out chips are used together with a single sensor wafer. The prolonged pixels extend the active region of the detector module.

telescope sensors, as well as any additional material or DUTs introduced into the beam, is described via the XML-encoded GEAR file. Typical examples include light shielding or encasings for environmental control, most prominently cooling systems and temperature stabilisation. Any corrections to the alignment are made available by an updated GEAR file, allowing users to easily compare the alignment between runs or between alignment iterations.

Additionally, the geometry package of ROOT, used on top of the GEAR description, has been introduced in EUTELESCOPE to expand its functionality as well as simplify the user interface to the telescope geometry. It allows to overcome restrictions of the GEAR-only description, namely the limitation to same-sized, rectangular and periodically arranged pixel layouts. In the current framework, more complex pixel layouts can be appropriately described. Hence, it allows for the usage of any pixel geometry which can be described by the ROOT geometry package. This specifically includes staggered pixel layouts and configurations featuring gaps or differently sized pixels in certain regions, as depicted in figure 3. Libraries for new pixel layouts describing the relation between pixel indices and centre position of the pixels can be loaded dynamically by EUTELESCOPE at runtime if required. Furthermore, the enhanced geometry framework allows for fast transformation between coordinate systems via the ROOT geometry package. This fast transformation is used in particular in the alignment process.

## 2.4 Further external packages

Interfaces to external packages enhance the functionality offered by EUTELESCOPE, especially in terms of detector alignment, track models, and data analysis and visualisation. For the latter, the ROOT package is used and a description is omitted here. For the former, MillepedeII and the General Broken Lines library are shortly described below.

### 2.4.1 MillepedeII

EUTELESCOPE makes use of the MillepedeII package for the alignment of detector set-ups. MillepedeII is comprised of two standalone parts, namely the MILLE and the PEDE program. MILLE provides an interface to write tracks into a binary file which are then used by PEDE to perform a least-squares fit. MillepedeII is an actively maintained fortran90 package with MILLE interfaced to C.



In most track-based alignment scenarios, two classes of parameters are distinguished: global parameters which are common to the entire set of tracks considered, and local parameters describing individual tracks. Examples of global parameters are the alignment parameters for the telescope sensors, i.e. shifts and rotations towards a defined frame of reference. MillepedeII performs an overall least-squares fit in which both global and local parameters are determined simultaneously. The global parameters are applied appropriately to the detector alignment parameters described in the GEAR file. EUTELESCOPE provides an interface to choose between various alignment modes available in MillepedeII allowing for the combination of shifts and rotations along all axes for both the telescope sensors and the DUTs.

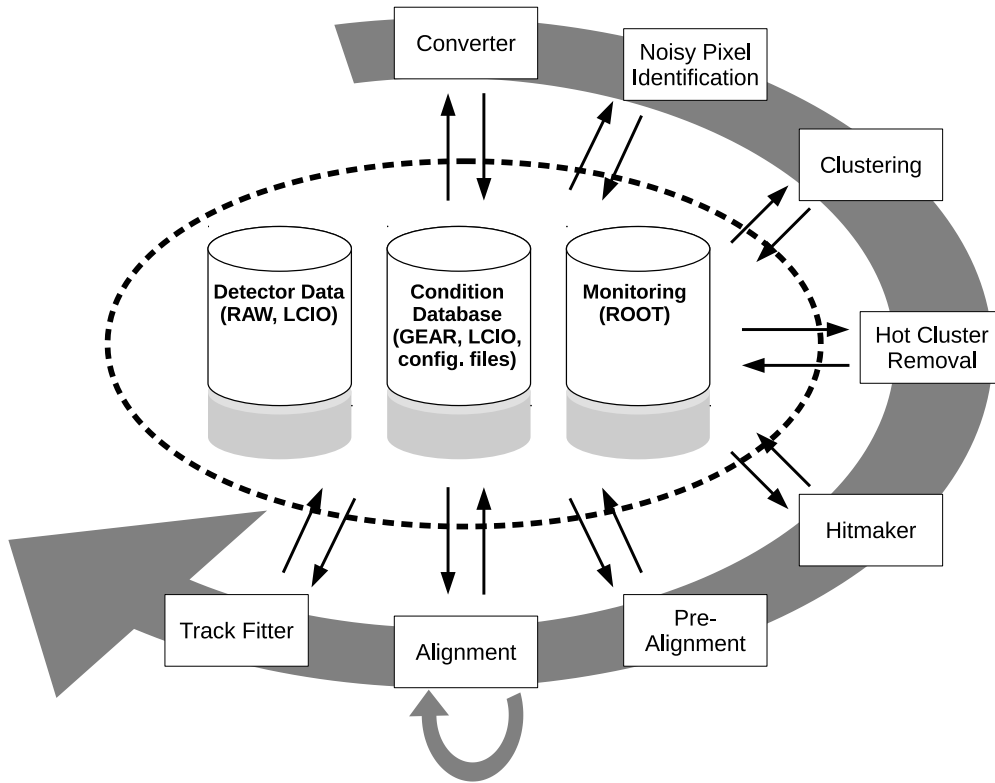
In most practical cases, the position of the detectors along the beam direction can be measured with sufficient precision, i.e. with sub-millimetre resolution. Owing to the low sensitivity of the track model to this coordinate, it constitutes a weak mode and can safely be excluded from the alignment. Rigid mechanics supporting the beam telescope sensors constrain yaw and pitch, i.e. rotations around axes perpendicular to the beam axis.

### 2.4.2 GBL

The General Broken Lines track model is a track re-fit taking into account multiple scattering to accurately describe particle trajectories traversing material. Providing the complete covariance matrix for all track parameters makes the GBL track model suitable for alignment with MillepedeII. The GBL method is mathematically equivalent to a progressive Kálmán filter, but yields a linear equation system with an almost diagonal (bordered band) matrix to be solved by a fast Cholesky decomposition avoiding inversion. The GBL library is actively maintained, uses Eigen3 as the underlying linear algebra library and includes an interface to MillepedeII for track-based alignment. Details of the implementation in EUTELESCOPE are discussed in section 4.

## 3 Reconstruction flow

Using MARLIN, the various steps of the reconstruction flow are executed sequentially, each step configurable by the user. A typical work flow for data reconstruction of pixel devices is illustrated in figure 4. Initially, data are converted from the native format of the DAQ system to the corresponding LCIO format. Subsequently, noisy pixels can be determined, yielding a noisy pixel database which is the foundation of any subsequent noise treatment. The next reconstruction step groups adjacent pixel hits into clusters for which different algorithms capable of dealing with binary and non-binary read-out as well as equidistant and non-equidistant geometries are provided. Using the noisy pixel database, clusters containing at least one noisy pixel can be masked and removed, resulting in a noise-free cluster collection. Hit positions in the local reference frame (the sensor reference frame) are derived based on the cluster shape. These hit positions are used to determine a preliminary alignment by investigating the correlations between the foremost sensor and all downstream sensors in the beam telescope as well as DUT sensors. Starting from the pre-aligned set-up, the next step derives correction constants to the alignment, using the MillepedeII framework. This step might be repeated in order to iteratively converge towards a final set of alignment constants. The last reconstruction step produces the final track fits and possibly exports the tracks for use in an external analysis framework.



**Figure 4.** Different steps within a EUTELESCOPE track reconstruction. The boxes indicate a reconstruction processor, the cylinders denote as-recorded, derived and user-provided data files as well as monitoring output.

### 3.1 Raw data conversion

During data conversion, serving as the initial step of the reconstruction flow, the test beam data are extracted from the custom format of the DAQ system and interpreted and stored in the LCIO format using the interfacing tools discussed in section 2. In principle, the conversion can occur outside or inside EUTELESCOPE. For data acquired with the EUDAQ1 framework, converter plug-ins defined therein convert the recorded data into event-based collections using the `EUTelNativeReader` processor. The event building was already performed during the data acquisition step and therefore the LCIO event contains all of the data associated with the physical event. For data acquired with the EUDAQ2 [28] framework, the data conversion happens outside and prior to the data analysis in EUTELESCOPE. The event building is performed offline based e.g. on time stamps or on trigger IDs and subsequently the data is converted to LCIO or any other data format, preparing the data for usage with EUTELESCOPE.

### 3.2 Noisy pixel treatment

Occupancy maps of the various sensors are produced counting the number of hits which were recorded over a certain number of events for every pixel using the `EUTelNoisyPixelFinder` processor. The processor allows to configure the maximum occupancy, above which pixels are masked as noisy, the number of events to determine whether a pixel is noisy, and which sensor

planes should be considered. These pixels are then stored in a noisy pixel database. Histograms showing the frequency distribution of firing pixels as well as the spatial distribution of the noisy pixels on each sensor are created allowing to monitor the noise behavior of the sensors.

### 3.3 Clustering

In the clustering step `EUTelSparseClustering`, fired pixels in the same LCIO collection are grouped and new cluster collections are created and stored. Exploiting the previously created noisy pixel database, the noisy cluster masking step `EUTelNoisyClusterMasker` tags clusters containing at least one noisy pixel. `EUTelNoisyClusterRemover` creates a collection free from any of the clusters containing pixels tagged as noisy. The modular approach of `EUTELESCOPE` also allows to remove noisy pixels prior to clustering.

A default clustering algorithm for zero-suppressed data is available grouping pixels with adjacent indices. Additionally, the clustering algorithm can be configured to require touching edges of the pixels or to also include pixels with touching corners. As only zero-suppressed data are clustered and no threshold is applied, the algorithm results in an unambiguous set of clusters without the need of seeding.

In addition, a clustering routine which exploits the geometric layout of the pixel implants is implemented, i.e. `EUTelGeometricClustering`. It uses the physical position of the pixel and clusters hit pixels in close spatial proximity. A use-case for this algorithm is a sensor with a staggered pixel matrix. In such a sensor, each pixel implant is surrounded by six other pixels, and not eight (with a common corner) or four (with a common edge) as is the case of a regularly aligned pixel matrix. The geometric clustering routine correctly links and clusters the five adjacent pixels together.

Within the framework, there is no dedicated mechanism to propagate the information that a noisy cluster was removed. At the fitting step there is no information available that at a given position there is a potential missing hit, due to a removed noisy cluster. This information could easily be derived using the noise tagged cluster collections, however as this is not used in the fitting procedures, this mechanism is not implemented by the default *eutel* processors.

### 3.4 Hit position derivation

In the `EUTelHitMaker` step, hit positions in the local frame of reference are derived from the previously obtained clusters. For this, the cluster collections are read in by the processor and the pixels associated with the cluster are retrieved to calculate the hit position. The hit position in the local frame is derived using a charge-weighted centre of gravity:

$$\bar{x} = \frac{1}{Q} \sum_{i=0}^N x_i q_i, \quad (3.1)$$

where  $x_i$  is the index of the  $i$ -th pixel in the cluster,  $q_i$  the charge recorded in that pixel and  $Q = \sum_i q_i$  the total charge. Summation runs over all pixels linked to that cluster for both spatial indices separately. Once the hit position is obtained it is stored in an LCIO collection.

The subsequent processors require hits in the global frame, i.e. user-specified shifts and rotations need to be applied to the hits in the local frame of reference to obtain *global hits*. This is achieved by performing an extrinsic rotation  $M = YXZ$  around the global  $y$ -,  $x$ - and  $z$ -axes, starting with

the latter, using the Tait-Bryan notation of Euler angles, and by additionally shifting the hits by the alignment values provided by the GEAR file. In many cases, the local coordinate system of the first sensor plane defines the coordinates of the global system perpendicular to the beam direction and the beam direction itself defines the  $z$ -axis. Such coordinate transformations are done by `EUTelHitCoordinateTransformer`.

### 3.5 Alignment

The not yet aligned hits in the telescope frame (the global frame of reference) resulting from the last step are used in two following processors separately: the correlation processor `EUTelCorrelator` produces two-dimensional correlation plots for data quality management and the pre-alignment processor `EUTelPreAligner` produces an updated, hence a pre-aligned, GEAR file to correct for  $x$ - and  $y$ -shifts of the sensors. To this end, residuals in both the  $x$  and  $y$ -direction are calculated: The spatial hit position from the first sensor is propagated to all latter ones assuming no beam divergence and initial alignment parameters, if specified, are applied. The difference between the propagated and the measured hit position is filled in one- and two-dimensional histograms. The pre-alignment processor finds the bin with the highest event count to determine pre-alignment constants. In the case of multiple tracks in a single event all possible permutations of propagated and measured hit positions are considered, which pass a simple residual cut configurable in  $x$ - and  $y$ -components.

The `EUTelHitCoordinateTransformer` processor transforms the local hits to the global coordinate system using the alignment constants in the pre-aligned GEAR file. For the alignment with MillepedeII, a binary file is written containing information from preliminary tracks, i.e. local and global derivatives, residuals between a straight-line seed track and the measurement uncertainties. The track finding and track fitting is performed in `EUTelGBL` and is explained in detail in section 4. The results from MillepedeII are then propagated back into the `EUTELESCOPE` framework and a new GEAR file is written. In order to improve the precision of the alignment, this process can be iterated, each time using the updated GEAR file and therefore the updated alignment constants, as is indicated by the circular arrow in figure 4. In a usual telescope set-up, a few ten thousand tracks are sufficient to align the sensors of the typical EUDET-type beam telescope.

### 3.6 Track fit and analysis

In the last step of the reconstruction, again `EUTelGBL` is used to perform the final track fit. Various examples using GBL tracks are included in the framework repository, see section 5. The obtained reconstructed data can be exported e.g. as a ROOT tree to be used in an external analysis framework. If required by such an external framework, the track positions on the sensors can be transformed back to the local coordinate system. In addition, other LCIO collections, e.g. the pixel hit collection, can be exported through either existing or user-provided export routines.

## 4 General Broken Lines processor in EUTELESCOPE

The GBL track model describes the multiple scattering of beam particles in the material traversed by the particle, see also section 2.4.2. The `EUTelGBL` processor in `EUTELESCOPE` makes use of the GBL library [20, 21], using a set of hits resulting from a track finding algorithm.

## 4.1 Track finding

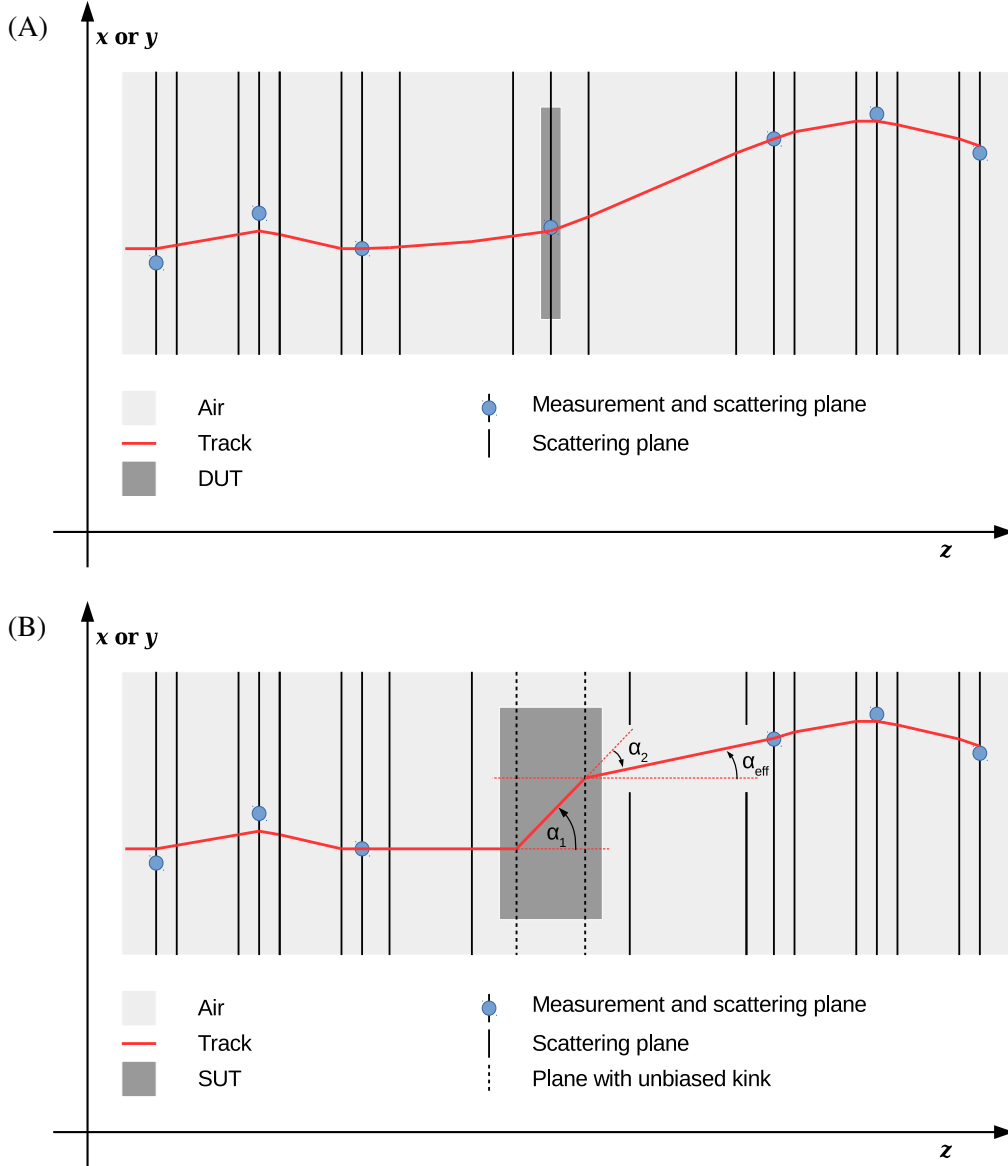
EUTeLGBL expects the beam telescope to consist of six sensors, which are grouped into an upstream and a downstream triplet. The beam direction defines the  $z$ -direction, the  $x$ - and  $y$ -direction are parallel to the pixel columns and rows of the first telescope sensor, with the origin at the centre of this sensor. A pattern recognition algorithm selects hits likely belonging to the same physical track [7, 29]. For each triplet, a straight line between a hit in the first and a hit in the last sensor is calculated, called a doublet. Only doublets with a slope within a user-defined cut are selected in order to decrease the number of false combinations, where the slope is calculated with respect to the nominal beam direction. Then, the distance between the doublet and the hits on the central sensor of the triplet is calculated within the plane of the central sensor. Valid triplets are defined as those triplets, where the distances in  $x$ - and  $y$ -direction are smaller than a user-defined cut value. The downstream and upstream triplets are then extrapolated to the centre of the beam telescope. If the distance between the extrapolations at a configurable  $z$ -position is within a user-defined cut  $d_{\text{match}}$ , the six hits are grouped to form a track. Isolation of upstream (downstream) triplets is ensured by discarding all triplets, whose extrapolations have distances to other upstream (downstream) extrapolations at a user-configurable  $z$ -position smaller than  $2d_{\text{match}}$ . In this case, both triplets are discarded.

Any sensor which is not a telescope sensor is considered a DUT. In order to match DUT hits to a track, the triplets are extrapolated onto the DUT planes and two cuts, one in the  $x$ - and one in the  $y$ -direction, are applied.

## 4.2 GBL track fitting

The GBL implementation in EUTELESCOPE makes use of GBL points. A GBL point can carry a series of attributes, e.g. the propagation matrix from one point to the next, one or more position measurements and their uncertainties, the width of the expected scattering angle distribution at this point in the thin-scatterer approximation, if it is known, as well as local and global derivatives. As the position measurement is optional, a GBL point can either describe a measured hit or any other point of interest along the trajectory, also those without a measured hit. The global derivatives, describing the impact of a change in alignment constants on the residuals, are required for the detector alignment. A GBL trajectory comprises a series of spatially ordered GBL points along the beam direction, connected via the Jacobian matrix providing the propagation from one GBL point to the next. The propagation is either a helix, in case of the presence of a magnetic field, or a straight line between the GBL points.

In practice, one GBL point per measurement plane is created, in addition to GBL points describing the scattering in passive material. Prior to the track fitting, a seed trajectory is formed as a straight line from the first to the last measurement point. Then, residuals in  $x$  and  $y$  of a hit with respect to the seed trajectory and the hit resolution are added to the GBL point of the respective sensor. The amount of multiple scattering assigned to a GBL point is calculated according to the Highland formula and depends on the momentum of the beam, the radiation length of the material and its thickness [30]. In order to describe the path of a particle traversing an extended volume of material between two sensors, two GBL points are introduced in between the two sensors in order to describe the trajectories' offsets and kinks, see black lines representing the scatterer in figure 5 (A)



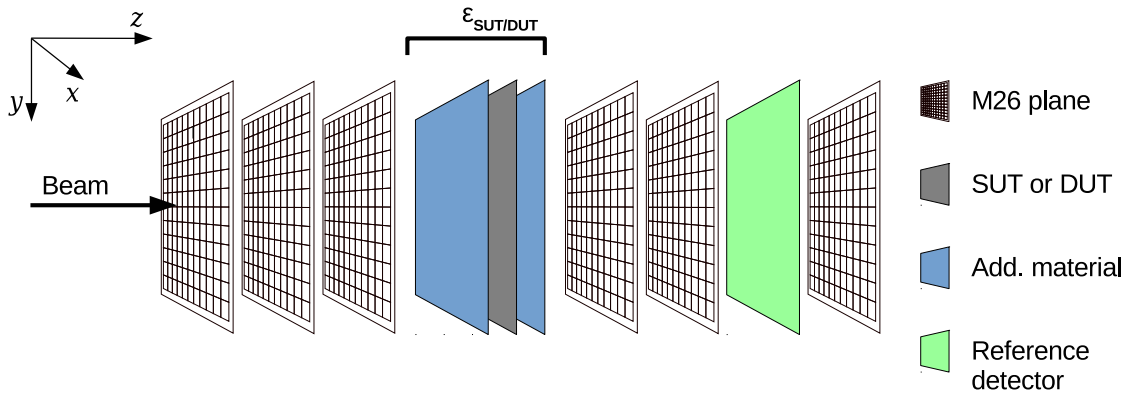
**Figure 5.** The GBL track model with straight lines connecting the scatterers in case of a DUT (A) and a passive sample under test (SUT) of unknown material budget (B) in the centre of the beam telescope.

and (B). At these points the track is allowed to scatter reflecting the multiple scattering expected to happen in the traversed volume. These points do not have any measurement associated with them. Finally, the track fit is performed based on all GBL points. In case a sensor is excluded from the fit, its hit information and global parameters are not added to its GBL point. This might be the case for the DUT, in order not to bias the track fit by the hit information of the DUT.

If the material budget

$$\varepsilon = \frac{x}{X_0} \quad (4.1)$$

of a slab of material or sensor with thickness  $x$  and radiation length  $X_0$  is the quantity under study, no information on the expected multiple scattering is associated to the corresponding GBL point.



**Figure 6.** A sketch of the beam telescope setup used in the examples.

Instead, two local derivatives in both the  $x$ - and the  $y$ -direction are added to the measurements on each sensor downstream of the unknown scatterer, which are multiplied with the distance to the scatterer [31]. These local derivatives are kept as free parameters during the track fit, i.e., these parameters do not enter in the  $\chi^2$  of a track. The local derivatives, in combination with their lever arm, act as free kink angles in the trajectory. However, the two angles in each direction are highly correlated. Therefore, the sum of the two angles in the  $x$ -direction and the sum of the two angles in the  $y$ -direction is calculated, representing a total or effective kink angle at the passive sample under test (SUT), see figure 5 (B), This ensures an unbiased estimate of the kink angle.

## 5 Reconstruction examples

The following sections give practical examples of how the EUTELESCOPE framework can be used to reconstruct data from a variety of beam test setups: empty beam telescopes as well as beam telescopes with either purely passive scatterers or active devices under test with different sensor geometries. For these examples, MIMOSA 26 sensors are used for precise spatial measurements of particle trajectories. [32] Each MIMOSA 26 sensor consists of pixels sized  $18.4 \mu\text{m} \times 18.4 \mu\text{m}$ , which are arranged in 1152 columns and 576 rows. This adds up to a total of about six hundred thousand readout channels per sensor, covering an active area of about  $21.2 \text{mm} \times 10.6 \text{mm}$ . The specifications of the MIMOSA 26 sensors quote a thickness of  $50 \mu\text{m}$  and a resolution of about  $3.3 \mu\text{m}$  was measured. [7] The GBL method is used for track fitting, as described in the previous section, although other alternatives, such as the Deterministic Annealing Filter (DAF) Fitter, are available. The setup of the beam telescope used in all examples is depicted in figure 6. Documented configuration files and original data files are provided as part of the EUTELESCOPE installation [33].

### 5.1 Empty beam telescope analysis

The example with an empty beam telescope, i.e. without any additional device or scatterer, is referred to as GBL\_noDUT here and in the documentation. It is used to illustrate details of the telescope data reconstruction. The data presented here were taken using the DURANTA beam telescope [7], consisting of six MIMOSA 26 sensors, at the DESY II testbeam facility [34] using an electron beam

with a momentum of 4 GeV/c. The  $z$ -positions of the six telescope sensors along the beam axis were 0, 23.5, 47.0, 127.5, 178.5, and 229.5 cm, allowing for a larger lever arm downstream.

### 5.1.1 Raw data conversion and noisy pixel treatment

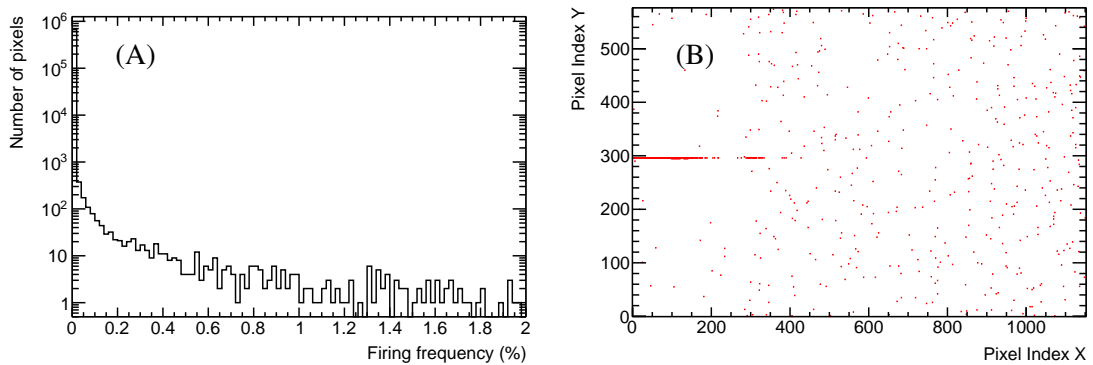
The RAW data are converted to LCIO format as described in section 3.1. For this example, the data were collected using EUDAQ2 and were therefore converted to LCIO format using the external `euCliConverter` in the EUDAQ2 package.

The noisy pixels database is created using the `EUTelNoisyPixelFinder` processor, which computes the occupancy of pixels and applies a cut on this value to mark noisy pixels. The occupancy of noisy pixels for sensor plane 1 of the telescope is shown in figure 7 (A). In the `GBL_noDUT` example any pixel with an occupancy greater than 0.1% is classified as a noisy pixel. A strip of noisy pixels around pixel index  $y$  of 300 can be seen in figure 7 (B), which is a known feature of this sensor in the telescope and therefore the corresponding pixels should be masked. In total, 682 pixels are masked noisy, corresponding to approximately 0.1% of all the pixels on this sensor plane.

Ideally, the noisy pixels are determined from data collected while no beam is present and this collection is used in all subsequent runs. However, while used here only for the sake of an example, on-beam noise suppression typically works rather reliably and can serve as a tool to monitor the sensor performance during operation. If the noise mask is derived from data, caution must be taken not to mask pixels which are in the centre of the beam spot and therefore have a higher occupancy. This is done by varying the noise threshold and validating that not predominantly pixels in the beam spot are masked by inspecting the change in the two dimensional noisy pixel maps. Additionally, control plots during the reconstruction provide data on the number of noisy pixels versus a varied noise threshold.

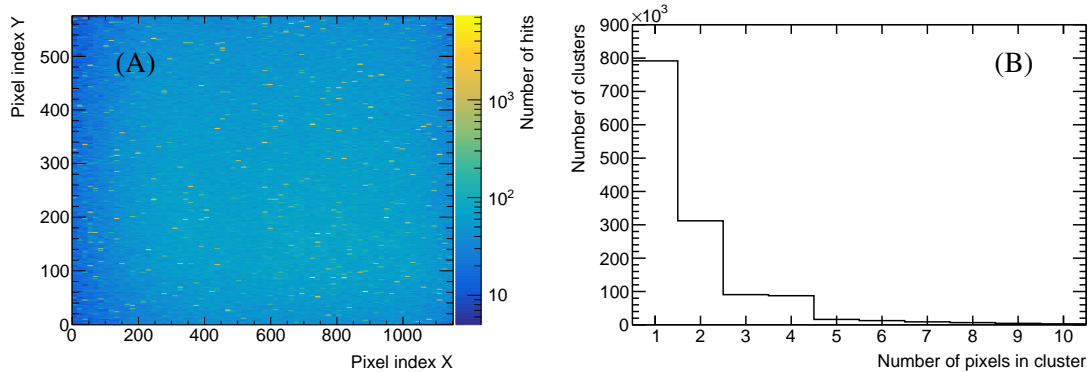
### 5.1.2 Clustering

Adjacent pixel hits are grouped together into clusters using the `EUTelSparseClustering` processor. Clusters that contain pixels listed in the hot pixel database are masked and removed from the output LCIO collection using the `EUTelNoisyClusterMasker` and `EUTelNoisyClusterRemove`



**Figure 7.** The occupancy of all pixels (A) and distribution of noisy pixels in the  $x$ - and  $y$ -direction (B) using the `EUTelNoisyPixelFinder` processor for the `GBL_noDUT` example for MIMOSA 26 sensor plane 1.





**Figure 8.** Cluster hit map (A) and cluster size (B) for the GBL\_noDUT example for one of the MIMOSA 26 sensor.

processors. The hit map for clusters and the associated cluster size distribution are shown in figure 8 (A) and (B), respectively. The cluster hit map is mostly uniform after removing clusters containing at least one noisy pixel; a slightly higher density of clusters in the centre of the sensor reflects the profile of the incident beam. The cluster size distribution, shown in figure 8 (B), peaks at 1.

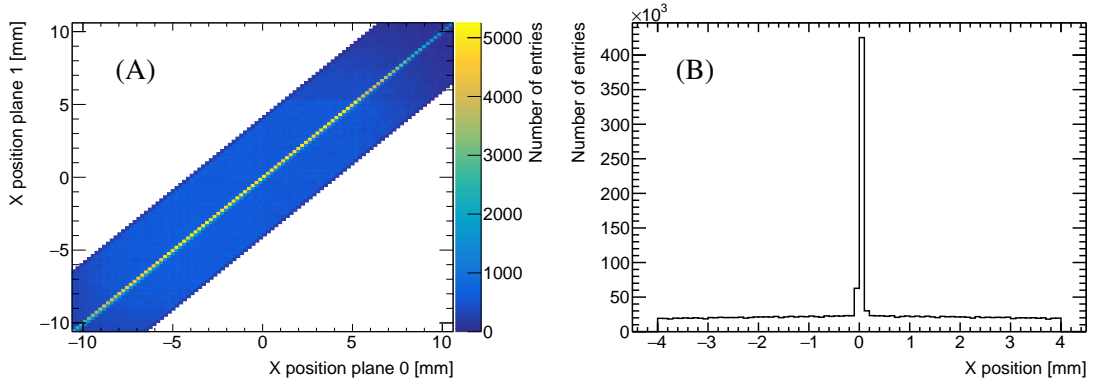
### 5.1.3 Hitmaker and pre-alignment

The hits are calculated based on the clusters and are translated from the measurement frame of reference (the sensor plane) to the global frame of reference using the `EUTelHitMaker` processor and the GEAR geometry description. The `EUTelPreAligner` processor is used to calculate pre-alignment constants in the global coordinate system, keeping the first telescope sensor fixed. The resulting alignment constants are applied to the GEAR geometry file, while in the LCIO file only the hit coordinates in the local frame are stored.

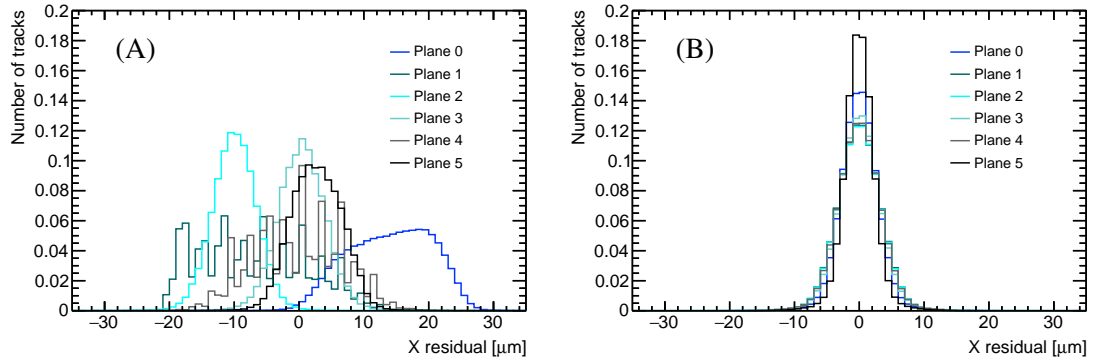
Example outputs of the hit correlations before pre-alignment are shown in figure 9 (A) and (B). Hits with spatial difference of greater than  $\pm 4$  mm are not considered. `EUTelCorrelator` yields figure 9 (A) and shows the hit correlation along the  $x$ -axis between the first and second telescope sensors, i.e. plane 0 and plane 1, for the GBL\_noDUT example. A sharp diagonal through the origin indicates a good alignment between them. Figure 9 (B), produced with `EUTelPreAligner`, shows the hit distance distribution on plane 1 with respect to plane 0 along the  $x$ -axis. Correlations between the hits are clearly visible by the sharp peak, with a shift smaller than 0.1 mm in the  $x$ -direction indicating the accuracy of the alignment during the first alignment step.

### 5.1.4 Alignment

After applying the pre-alignment constants using the `EUTelHitCoordinateTransformer` processor, the alignment of the sensors is performed using the `EUTelGBL` processor in *alignment mode*. `EUTelGBL` employs the hits in the global coordinate system and performs track finding and fitting using GBL as described in section 4. In this GBL\_noDUT example, shifts in the  $x$ - and  $y$ -direction and rotations in the  $z$ -plane are allowed, with the first and last telescope sensor fixed in position and rotation to constrain weak modes, i.e. shifts along the  $z$ -direction as well as rotations and tilts around



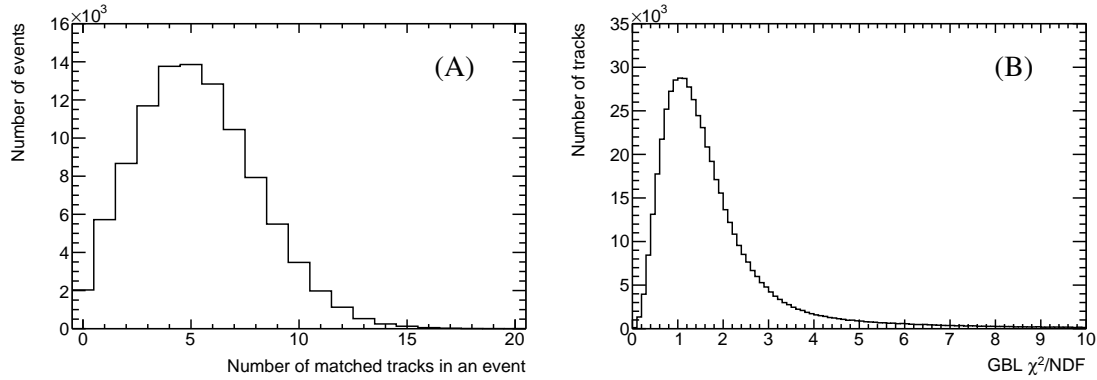
**Figure 9.** Hit correlation along the  $x$ -axis between the first and second telescope sensors, plane 0 and plane 1, (A) and the hit distance distribution on plane 1 with respect to plane 0 along the  $x$ -axis (B).



**Figure 10.** The residual for each telescope sensor during the first (A) and third (B) alignment iteration along the  $x$ -axis.

the  $x$ - and  $y$ -axis. The EUTelPedeGEAR processor provides an interface between MillepedeII and the GEAR geometry description, and is used to write out new alignment constants for each iteration. The alignment is performed iteratively, using the EUTelGBL processor, with each step writing out a new GEAR geometry file with an improved estimate of the alignment. The TripletGBLUtility processor has the functionality to suggest suitable track quality criteria to apply to each successive alignment iteration in order to improve the final result. This is based on approximating the residuals as Gaussian and suggesting cuts of  $\mu \pm 4\sigma$ , where  $\mu$  is the mean and  $\sigma$  is the width of the Gaussian.

The residuals for the GBL tracks after one and three alignment iterations are shown in figure 10 (A) and (B), respectively. After one alignment iteration, the residuals of some of the telescope sensors exhibit sizeable shifts and broad distributions, indicating that further iterations are necessary. After three alignment iterations, the residuals of all telescope sensors are centred at zero and have similar widths, indicating that the sensors are well aligned. The expected residual widths are further discussed in reference [7].



**Figure 11.** Number of matched tracks per event (A) and their  $\chi^2$  per degree of freedom (B) for the GBL fit.

### 5.1.5 Track fit

The final step in the reconstruction is to perform a track fit to the groups of six hits identified during track finding. The final alignment constants from the last GEAR geometry description is loaded using the `EUTelHitCoordinateTransformer` processor, and the fit to the telescope tracks is performed again using the `EUTelGBL` processor with the *alignment mode* turned off. The number of matched tracks per event, i.e. where an upstream triplet could be matched to a downstream triplet, and their goodness-of-fit are shown in figure 11. In this particular run, the number of matched tracks per event peaks at about five tracks per event. The distribution of the  $\chi^2$  per degree of freedom of the tracks indicates a good quality of the fit in the majority of tracks, peaking at approximately 1.0 with a mean of about 1.6. The tracks are written to file in the format of a ROOT n-tuple using the `EUTelGBLOutput` processor, allowing to further process the track data based on the particular user analysis needs.

## 5.2 Passive scatterer analysis

To investigate the material budget of a scatterer, a sample under test (SUT) can be placed in the centre of the beam telescope. Electrons undergo multiple scattering when traversing the inserted material and its effect on the trajectory is measurable for a large range of material budgets due to the momentum range available at the DESY II testbeam facility (up to  $6 \text{ GeV}/c$ ). Measuring each particle track with the beam telescope then allows for the calculation of the kink angle at the SUT position. The width of the kink angle distribution is in turn a function of the material budget of the SUT.

The SUT example, referred to as `GBL_SUT`, uses a data set taken at the testbeam line 21 with the `DATURA` telescope and a beam momentum of  $4 \text{ GeV}/c$ . The tested object is a copper block with four different areas with different thicknesses, which are chosen to reflect material budgets of  $\varepsilon = 0\%$ ,  $25\%$ ,  $50\%$  and  $100\%$ . In figure 12 (A), a picture of the copper block inserted between the telescope arms is shown.

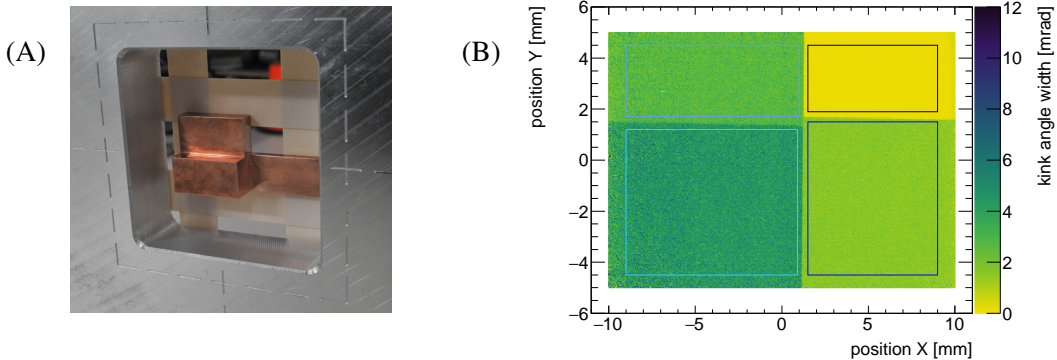
Data for this example were taken with `EUDAQ2` and converted into the `LCIO` format. For the reconstruction with `EUTELESCOPE`, the SUT is added as a plane in the geometry description and an estimate of the radiation length can be specified. The steps noisy pixel masking, clustering, hit making, and pre-alignment are executed in the same way as described in the `GBL_noDUT` example,

see section 5.1. For the alignment, in principle two different approaches are possible. Preferably, a dedicated alignment run with an empty telescope is used to align the MIMOSA 26 sensors and the SUT is inserted afterwards. This allows to simply use the aligned GEAR file when analysing runs that include the SUT. If dedicated alignment runs are not possible, the alignment is done with the SUT in place and with a SUT plane introduced in the GEAR file with an initial guess of the material budget. During execution of the EUTe1GBL processor the SUT plane is excluded from the list of sensors to be aligned. The EUTe1GBL processor has a parameter to flag, in the final track fit, the unbiased calculation of the kink angle at the SUT plane, see 4. The output contains the effective kink angle for each reconstructed track, allowing to produce a position-resolved map of kink angle widths. Again, with the ROOT file written by the EUTe1GBLOutput processor an in-depth analysis of the kink angles can be performed outside of EUTELESCOPE.

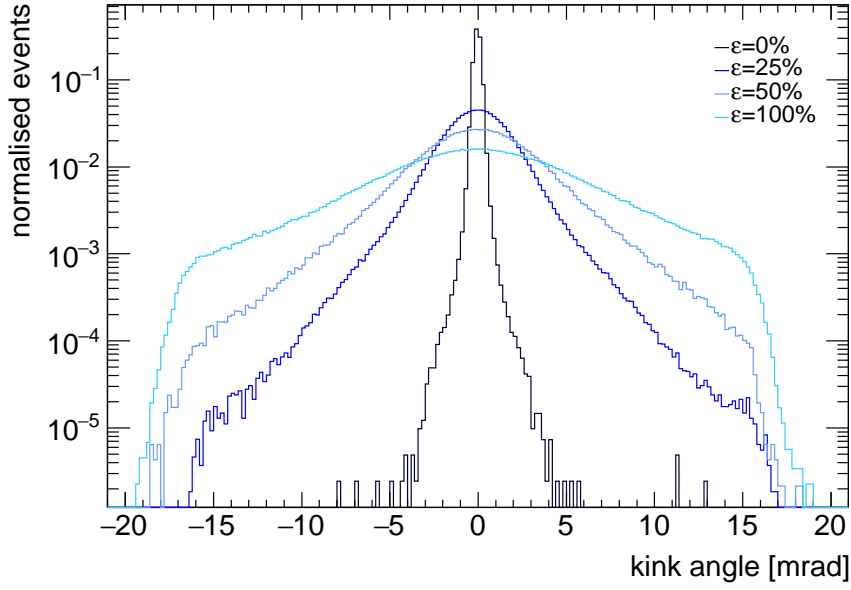
The reconstructed kink angle map is displayed in figure 12 (B), where the four regions of different material budgets are marked. For these regions, the kink angle distributions are plotted in figure 13. Here, the effect of increasing multiple scattering with increasing material budget is clearly visible. With the kink angle distribution and application of an appropriate scattering model, e.g. the Highland formula [30], it is possible to retrieve the (unknown) material budget  $\varepsilon$  of the SUT.

### 5.3 Analysis including the ALiBaVa system

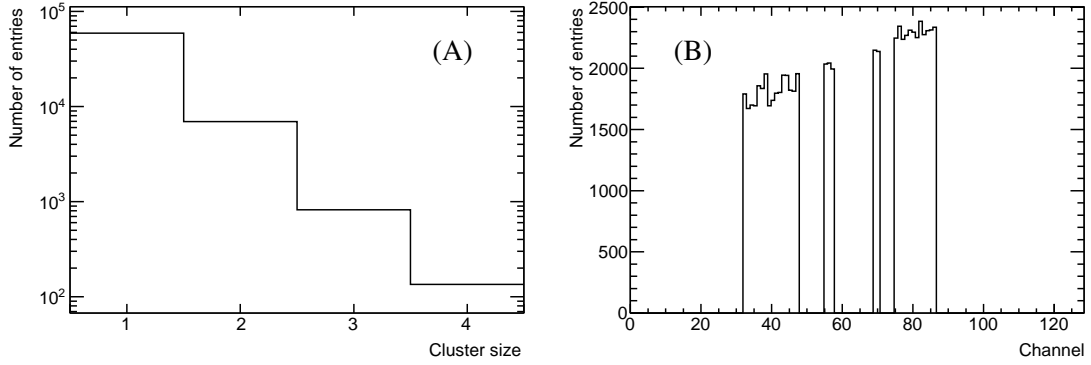
The example comprising the beam telescope together with a strip sensor as the DUT is referred to as ALiBaVa here and in the documentation. The strip sensor is read-out by the ALiBaVa system and is placed in the centre of the telescope. The ALiBaVa system is a compact read-out system for strip sensors and is based on the Beetle chip [35], which was developed for the LHCb experiment. It is commonly used to investigate properties of irradiated strip sensors or for performance studies of different sensor designs. The system has two main parts: the motherboard, which digitises signals, processes triggers and handles the communication with the read-out computer, and the daughter board, where two read-out chips and the sensor connection are located. Analogue front-end signals are sampled with the Beetle chip clock frequency of 40 MHz. The parameters of the pulse shaper and the preamplifier can be changed, to accommodate different load capacitances for different



**Figure 12.** (A) The studied copper block with four different areas featuring  $\varepsilon = 0\%$ ,  $25\%$ ,  $50\%$ ,  $100\%$ . (B) A map of the kink angle width of the SUT, marked are also the four different regions.



**Figure 13.** The kink angle distributions for the different regions of material budget  $\varepsilon$  indicated in figure 12.



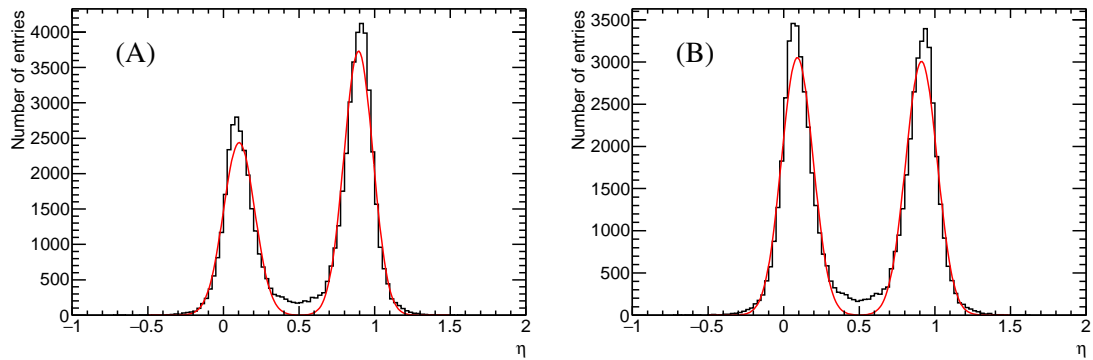
**Figure 14.** (A) Cluster size distribution in logarithmic scale, showing a mean cluster size of 1.1 with an RMS value of 0.4. (B) Distribution of found seeds during the clustering process over the channel domain.

sensors. A typical usage scenario has three different run types: first a calibration run, to obtain the correct conversion factor from ADCs to electrons; then an off-beam pedestal run, to obtain the base signal levels; and finally the actual data run. All run types are implemented in the EUTELESCOPE framework and in the following some of the important steps are highlighted.

The noise analysis of individual channels is done via a multitude of processors taking care of pedestal noise and common noise<sup>6</sup>. Similar to the clustering for the MIMOSA 26 sensors, the corrected ALiBaVa data is clustered using AlibavaClustering. Details are described in reference [36].

The data analysed in this example was recorded with the DATURA beam telescope at the DESY

<sup>6</sup>AlibavaPedestalNoiseProcessor, AlibavaPedestalSubtraction, AlibavaConstantCommonModeProcessor, AlibavaCommonModeSubtraction



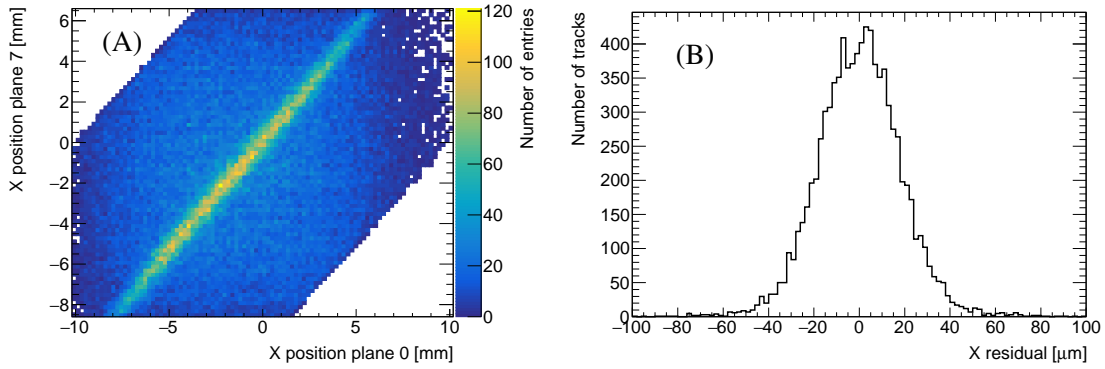
**Figure 15.** (A) An asymmetric  $\eta$ -distribution as observed after clustering, indicative of cross-talk. The two Gaussian fits are added to guide the eye. (B) The  $\eta$ -distribution after cross-talk correction [36].

testbeam line 21 with a beam momentum of 5 GeV/c. The data was taken with a non-irradiated epitaxial sensor with p-stop isolation at -300V bias voltage. Figure 14 (A) shows the cluster size distribution in a run with 500000 events. The distribution decreases exponentially, with a mean cluster size of about 1.1 with an RMS value of about 0.4, which is typical for a non-irradiated sensor of used pitch and active thickness. Figure 14 (B) shows the distribution of seed coordinates. In each event, the processor searches for a channel with a corrected signal larger than five times the noise value of the respective channel. The channel must not be masked and must not be adjacent to a masked channel on either side. Channels fulfilling these requirements are considered seeds of a cluster. Non-masked channels, neighbouring a seed, are added to the cluster if they have a signal higher than 2.5 times their noise. Due to the shape of the beam profile in this particular run, more seeds are found on channels with a higher channel number. The dead channels on the DUT sensor are because of faulty wire-bonds. As they and their neighbouring channels are masked in the analysis, this results in the missing channels seen in figure 14 (B).

After clustering, the  $\eta$ -distribution is calculated for each run to correct any cross-talk in the ALiBaVa data as shown in figure 15. With a second-order finite impulse response filter, moving from the rightmost to the leftmost available channel, the reconstructed DUT data is filtered in two iterations. After each iteration, the data is reclustered to calculate the charge-to-seed ratios. Two iterations usually suffice to correct for any cross-talk noise. One possible source of the cross-talk could be the serial readout mode of the Beetle chip used in the ALiBaVa system: in each event, channels are read out serially, starting with channel zero. A signal distortion in the readout cable could lead to charge from earlier channels being added to later channels. A bias toward higher channel numbers could be the cause of the asymmetric cross-talk observed in the  $\eta$ -distributions. The ALiBaVa cluster data stream is then merged with the clustered telescope data using the AlibavaMerger processor. Subsequently, the analysis chain as exemplified in section 3 is followed and detailed in references [36, 37].

#### 5.4 Analysis including a strip detector and a FE-I4 reference detector

The integration time of the pixel sensors used in a beam telescope might not match the integration time of a DUT read-out. For this reason, a reference sensor with an integration time similar to the



**Figure 16.** (A) Correlation plot of the  $x$ -position of hits in the FE-I4 detector with the  $x$ -position of hits in the first MIMOSA 26 sensor. (B) Biased residuals between fit position and hit position in the FE-I4 detector.

DUT is required to be able to measure the hit detection efficiency of the corresponding DUT.

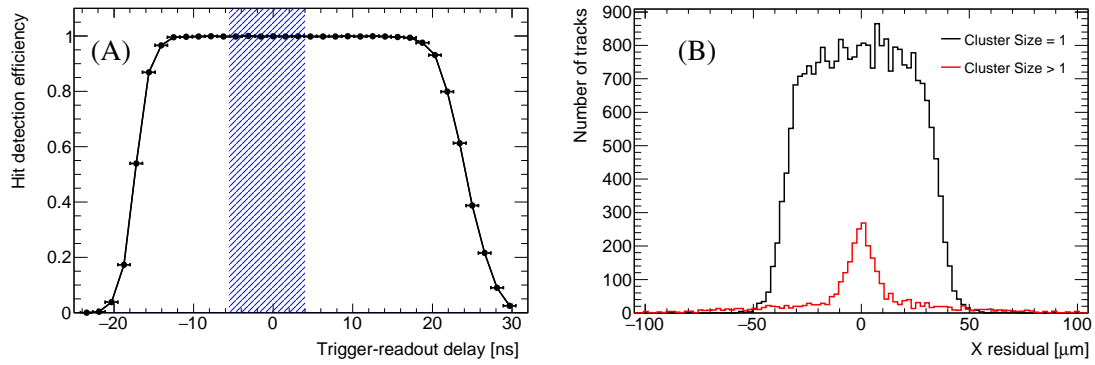
The example called GBL\_DUT uses data taken in test beam 22 at DESY with a beam momentum of  $4\text{ GeV}/c$ . In this example, the tracks extrapolated from the DURANTA telescope were used to characterize an ATLAS ITk Strip module [38] with a pitch of  $75.5\ \mu\text{m}$ , which was placed in the centre of the beam telescope. An additional FE-I4 pixel detector was placed before the last telescope sensor and used as a reference sensor [39]. This is an example of a standard measurement used widely in detector R&D and quality control during production, where either a sensor, the DAQ electronics and DAQ system, or a combination of those, is under test.

The data were taken with EUDAQ2 and the conversion to LCIO was performed externally. All reconstruction steps use the reconstruction scheme described in sections 5.1. The same clustering algorithm is used, as is the GBL algorithm for alignment and the final track fit. The noisy pixels are not masked for the DUT, thus leaving the possibility for the user to perform this task in the post-reconstruction processing.

In figure 16 (A) the correlation plot of the FE-I4 with respect to the first MIMOSA 26 sensor is shown. A slightly blurry diagonal protrudes indicating a synchronised data taking and a parallel orientation of the  $x$ -direction of the sensor. In the alignment step, no cut is used on the distance between the insensitive  $y$ -position of the DUT and the track extrapolation on the DUT. Three alignment iterations have been performed.

In the final track fit, the hit information of the DUT is not included by the track fitting algorithm, in order not to bias the track position with the hits from the tested device. Only the tracks with a matched hit on the FE-I4 are used in the analysis. These are the only tracks that are in-time with the read-out of the DUT, and thus the ones usable for DUT-related studies. Figure 16 (B) shows the biased residual distribution in the  $x$ -direction of the FE-I4 ( $50\ \mu\text{m}$  pitch). A Gaussian width of about  $(16 \pm 1)\ \mu\text{m}$  is observed, which is compatible with the expectation for the given set-up.

Figure 17 (A) shows the hit detection efficiency of the DUT as a function of the time delay between the arrival of the trigger and the readout of the channels. The efficiency reaches almost 100% for a plateau of about 30 ns and the shaded region was used for event selection. Figure 17 (B) shows the residual distribution for the strip detector separately for hits with a single strip firing and with more than one strip firing. Hits with a single firing strip exhibit an approximately



**Figure 17.** (A) Hit detection efficiency of the DUT as a function of the time delay between trigger and readout. The blue area denotes the time window used in the data analysis. (B) Unbiased residuals of the DUT along the sensitive coordinate of the strips. The residuals for hits with a single strip and for hits with more than one strip are shown separately.

rectangular distribution with a width slightly below the strip pitch, as expected. As a consequence of charge sharing, hits with more than one firing strip exhibit a much narrower distribution, but occur considerably less frequently.

## 6 Conclusion

This document presents the EUTELESCOPE software framework for the reconstruction of particle trajectories recorded with beam telescopes. Its modular and comprehensive design accommodates many use cases independent of sensor choice, geometry and beam conditions. The wide range of applications was demonstrated through examples ranging from an empty beam telescope set-up, to the inclusion of passive SUTs as well as active DUTs and an additional timing reference detector. A precise description of particle trajectories is rendered possible using the GBL track model. Track residual widths as expected from the sensor geometry as well as sub-micrometer alignment precision are achievable by making use of various external packages for a proper geometrical description of the set-up and its alignment. The configurable reconstruction flow and the individual processors with some of their key features were highlighted. Additionally, the examples shown are reproducible for the user with data samples available from the framework repository. The user-driven development of EUTELESCOPE allows for and invites the addition of new processors targeting future applications.

## Acknowledgments

The measurements leading to these results have been performed at the Test Beam Facility at DESY Hamburg (Germany), a member of the Helmholtz Association (HGF). This work was supported by the Commission of the European Communities under the 6th Framework Programme ‘Structuring the European Research Area’, contract number RII3-026126.



## References

- [1] The EU Telescope Developers. EU Telescope. <http://github.com/eutelescope/eutelescope>. Accessed: 20.03.2020.
- [2] G. McGoldrick, M. Cerv, and A. Gorisek. Synchronized analysis of testbeam data with the Judith software. *Nucl. Instr. Meth. Phys. Res. A*, 765:140 – 145, 2014. HSTD-9 2013 - Proceedings of the 9th International Hiroshima Symposium on Development and Application of Semiconductor Tracking Detectors.
- [3] G. McGoldrick et al. Judith - Pixel particle detector testbeam analysis. <https://github.com/gmcgoldr/judith>. Accessed: 20.03.2020.
- [4] M. Kiehn et al. Proteus beam telescope reconstruction. <https://gitlab.cern.ch/unige-fei4tel/proteus>, <https://doi.org/10.5281/zenodo.2586736>. Accessed: 20.03.2020.
- [5] B. Schwenker. Test Beam Software Framework. <https://bitbucket.org/BenjaminSchwenker/tbsw.git>. Accessed: 20.03.2020.
- [6] D. Hynds, S. Spannagel, et al. Corryvreckan - The Maelstrom for Your Test Beam Data. <https://gitlab.cern.ch/corryvreckan/corryvreckan>. Accessed: 20.03.2020.
- [7] H. Jansen et al. Performance of the EUDET-type beam telescopes. *EPJ Techn. Instrum.*, 3(1):7, 2016. DESY-16-055, arXiv:1603.09669.
- [8] S. Spannagel. *CMS Pixel Detector Upgrade and Top Quark Pole Mass Determination*. Springer International Publishing, 2017. ISBN: 978-3-319-58879-7, <https://doi.org/10.1007/978-3-319-58880-3>.
- [9] A. Bulgheroni et al. EU Telescope: Tracking Software. <http://www.eudet.org/e26/e28/e182/e425/eudet-memo-2007-20.pdf>, 2007. Accessed: 20.03.2020.
- [10] A. Bulgheroni et al. EU Telescope, the JRA1 tracking and reconstruction software: a status report (Milestone). <https://www.eudet.org/e26/e28/e615/e835/eudet-memo-2008-48.pdf>. Accessed: 20.03.2020.
- [11] I. Rubinskiy. EU Telescope. Offline track reconstruction and DUT analysis software. <https://www.eudet.org/e26/e28/e86887/e107460/EUDET-Memo-2010-12.pdf>. Accessed: 20.03.2020.
- [12] T. Bisanz et al. EU Telescope 1.0: Reconstruction Software for the AIDA Testbeam Telescope. <https://cds.cern.ch/record/2000969>. Accessed: 20.03.2020.
- [13] ilcSoft community. Marlin - Modular Analysis & Reconstruction for the LINear Collider. [http://ilcsoft.desy.de/portal/software\\_packages/marlin/](http://ilcsoft.desy.de/portal/software_packages/marlin/). Accessed: 20.03.2020.
- [14] F. Gaede, T. Behnke, N. Graf, and T. Johnson. LCIO - a persistency framework for linear collider simulation studies. arXiv:physics/0306114, 2003.
- [15] ilcSoft community. LCIO - Linear Collider Input/Output. <http://lcio.desy.de/>, <http://ilcsoft.desy.de/portal/>. Accessed: 20.03.2020.
- [16] ilcSoft community. GEAR - GEometry API for Reconstruction. [http://ilcsoft.desy.de/portal/software\\_packages/gear/](http://ilcsoft.desy.de/portal/software_packages/gear/). Accessed: 20.03.2020.
- [17] Eigen3 authors. Eigen3. <http://eigen.tuxfamily.org>. Accessed: 20.03.2020.

- [18] V. Blobel. Software alignment for tracking detectors. *Nucl. Instr. Meth. A*, 566(1):5 – 13, 2006.
- [19] C. Kleinwort. Millepede II wiki. [https://www.terascale.de/wiki/millepede\\_ii/](https://www.terascale.de/wiki/millepede_ii/). Accessed: 20.03.2020.
- [20] C. Kleinwort. General broken lines as advanced track fitting method. *Nucl. Instr. Meth. Phys. Res. A*, 673:107–110, May 2012.
- [21] V. Blobel, C. Kleinwort, and F. Meier. Fast alignment of a complex tracking detector using advanced track models. *Computer Physics Communications*, 182(9):1760 – 1763, 2011.
- [22] C. Kleinwort. General broken lines wiki. <https://www.terascale.de/wiki/generalbrokenlines/>. Accessed: 20.03.2020.
- [23] R. Brun and F. Rademakers. ROOT - an object oriented data analysis framework. *Nucl. Instr. Meth. Phys. Res. A*, 389(1-2):81–86, 1997.
- [24] ilcSoft community. iLCinstall. [http://ilcsoft.desy.de/portal/software\\_packages/ilcinstall/](http://ilcsoft.desy.de/portal/software_packages/ilcinstall/), <https://github.com/iLCSoft/iLCInstall>. Accessed: 20.03.2020.
- [25] ilcSoft community and The EU Telescope Developers. iLCInstall for EU Telescope installation, a fork from iLCinstall. <https://github.com/eutelescope/iLCInstall>. Accessed: 20.03.2020.
- [26] The iLCSoft developers and Arling, J.-H. and Bisanz, T. EU Telescope iLCInstall: Version 2.1, April 2020.
- [27] P. Ahlburg et al. EUDAQ—a data acquisition software framework for common beam telescopes. *Journal of Instrumentation*, 15(01):P01038–P01038, jan 2020.
- [28] Y. Liu et al. EUDAQ2 – A flexible data acquisition software framework for common test beams. *JINST*, 14:P10033, 2019.
- [29] M. Dragicevic et al. Test beam performance measurements for the Phase I upgrade of the CMS pixel detector. *J. Instrum.*, 12(05):P05022, 2017.
- [30] V. Highland. Some practical remarks on multiple scattering. *Nucl. Instr. Meth. Phys. Res. A*, 129(2):497–499, 1975.
- [31] H. Jansen, J. Dreyling-Eschweiler, P. Schütze, and S. Spannagel. Scattering studies with the datura beam telescope. In Zhen-An Liu, editor, *Proceedings of International Conference on Technology and Instrumentation in Particle Physics 2017*, pages 243–250, Singapore, 2018. Springer Singapore.
- [32] C. Hu-Guo et al. First reticule size MAPS with digital output and integrated zero suppression for the EUDET-JRA1 beam telescope. *Nucl. Instr. Meth. Phys. Res. A*, 623(1):480 – 482, 2010. 1st International Conference on Technology and Instrumentation in Particle Physics.
- [33] Bisanz, T. and Arling, J.-H. and Behr, J. and Dreyling-Eschweiler, J. and Eichhorn, T. and Hamnett, P. and Jansen, H. and Morton, A. and Perrey, H. and Rossi, E. and Rubinsky, I. and Spannagel, S. and Yildirim, E. EU Telescope: Version 2.2, April 2020.
- [34] R. Diener, J. Dreyling-Eschweiler, H. Ehrlichmann, I.M. Gregor, U. Kötz, U. Krämer, N. Meyners, N. Potylitsina-Kube, A. Schütz, P. Schütze, and M. Stanitzki. The DESY II test beam facility. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 922:265 – 286, 2019.
- [35] R. Marco-Hernández. A portable readout system for microstrip silicon sensors (ALIBAVA). In *Nuclear Science Symposium Conference Record 2008*, pages 3201–3208. IEEE, 2008.

- [36] T. Eichhorn. *Development of Silicon Detectors for the High Luminosity LHC*. PhD thesis, Universität Hamburg, Hamburg, 2015. DESY-THESIS-2015-024.
- [37] M. Centis Vignali and T. Eichhorn. Characterisation of Irradiated Thin Silicon Sensors for the CMS Phase II Pixel Upgrade. *EPJC*, 77(8):567, 2017.
- [38] ATLAS Collaboration. Technical Design Report for the ATLAS Inner Tracker Strip Detector. Technical Report CERN-LHCC-2017-005. ATLAS-TDR-025, CERN, Geneva, Apr 2017.
- [39] M. Garcia-Sciveres et al. The fe-i4 pixel readout integrated circuit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 636(1, Supplement):S155 – S159, 2011. 7th International Hiroshima Symposium on the Development and Application of Semiconductor Tracking Detectors.