EUDET-Memo-2007-20

EUTelescope: tracking software

A. Bulgheroni[*],

T. Klimkovich, P. Roloff[†],

A.F. Żarnecki[‡]

December 3, 2007

Abstract

The main goal of the JRA1 within the EUDET project is the construction of new and the improvement of exisiting test beam infrastructures. The JRA1 collaboration developed a high resolution, low material pixel telescope to characterize Device Under Test. Along with this hardware instrument, a software tool performing all the off line procedures needed to extract from the data acquired by the DAQ the precise spatial information has been developed as well and made available to the community.
This paper is about the current status of the development of the tracking software tool named EUTelescope and the foreseen future improvements.

[*]INFN - Roma3, Roma, Italy
[†]DESY, Hamburg, Germany
[‡]Institute of Experimental Physics, University of Warsaw, Warsaw, Poland

1 Introduction

The main goal of the JRA1 within the EUDET project is the construction and the improvement of test beam infrastructures. One of the main activity currently on going is the development of a high resolution, low material pixel telescope along with a dedicated software framework called EUTelescope. The main objectives of EUTelescope are the following:

1. Reduce the input data produced by the DAQ system to a set of high level objects as particle track parametrization or space points to be used by the end users to characterize their instruments.

2. Describe the telescope in terms of figures of merit both at the sensor plane level (SNR, pedestal and noise level, Eta function correction, ecc...) and at the system level (alignment constant, detection efficiency, spatial resolution).

3. Collaborate in the development of a common software framework for experiments at the future International Linear Collider.

Keeping in mind the last item, EUTelescope has been coded as a Marlin package and uses LCIO[1] as a persistent data model.

2 The overall analysis strategy

EUTelescope has been designed to exploit as much as possible the modularity offered by Marlin, each single analysis step has been coded into a separate Marlin processor. This approach, a part of making the software debug much easier, makes possible to the final users to chain their own existing analysis codes at any position in the EUTelescope analysis stream.
Figure 1 is a flow diagram of the main analysis strategy. Similar processors are grouped into the blue boxes, while intermediate files are represented by the yellow ones. Hexagons represent input information that have to be provided either by the user or from a condition database. In the following subsections, the main steps of the analysis procedure are described.

2.1 Format conversion

The very first step in the analysis procedure is the conversion from the native format used by the DAQ software to the LCIO one. This step has been introduced only for debug purpose and it is going to be removed soon, forcing the DAQ software to write the readout data directly in LCIO format. To avoid any loss of information, during the conversion step only very few operations are performed, as the CDS calculation (Sec. 2.1.1), and, in any case, the integrity of the native raw data as to be preserved.

[1]All the ILC Soft packages are described here: `http://ilcsoft.desy.de/portal`

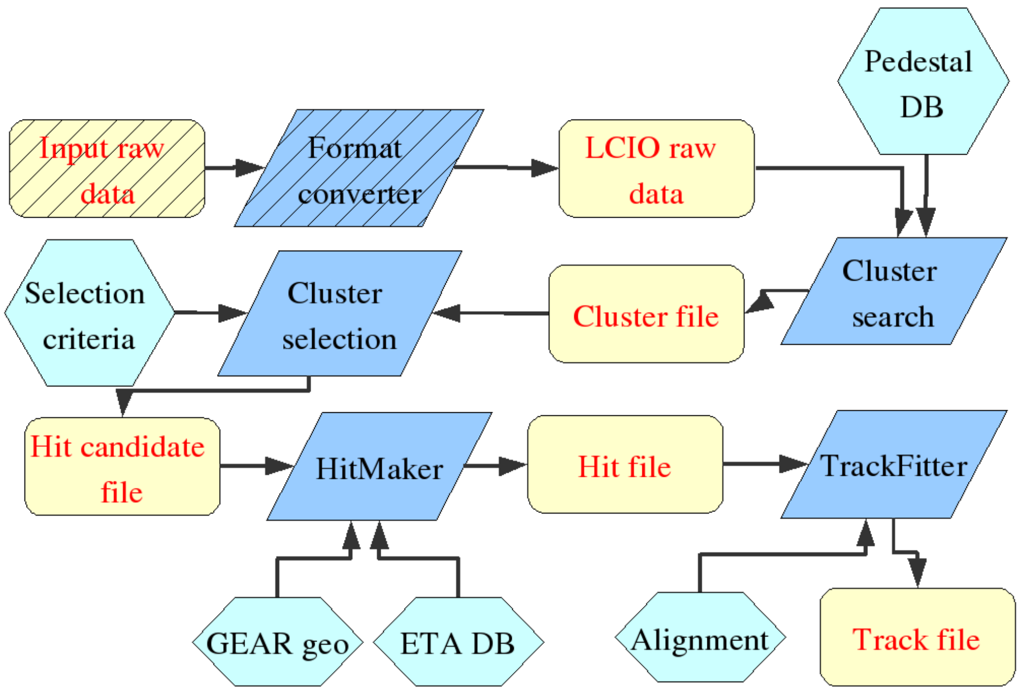

Figure 1: *A schematic representation of the overall analysis strategy*

A special processor called **EUTelMimoTelReader** inheriting from DataSourceProcessor has been written for this purpose. The use of this kind of processor requires not to have any LCIO input files in the global section of the steering file, and to provide one input native raw. The input native format is structured into events and characterized by a Begin Of Run Event (BORE) and an End Of Run Event (EORE) at the beginning and at the end of the run respectively. In all other events, different detector data streams can be stored together; the EUTelMimoTelReader is taking care to extract all the data streams corresponding to MimoTel detectors and copying them according to the adopted data model (see Sec. 3) into a LCIO event structure.

2.1.1 CDS calculation in RAW3 mode

CDS is a very powerful technique to reduce the noise in particle detectors. Monolithic Active Pixel Sensors uses CDS very extensively and it is based on the difference between two following sampling of the same pixel signal. When working in RAW3 mode, the DAQ producer is streaming out for each triggered event, three following full frames, being the trigger accepted by the system during the second one. Each frame contains all the readout pixels chronologically sorted, starting from the first one. The three readout frames are described in Fig. 2. The trigger is arriving sometime during the readout of the second frame. The arrow in the Fig. 2 is showing which pixel is readout when the trigger is accepted by the DAQ and not the hit pixel. The position of the arrow is dividing the second frame in two parts:

1. the preceding one (**region 1**) made by pixels that are already readout at the trigger

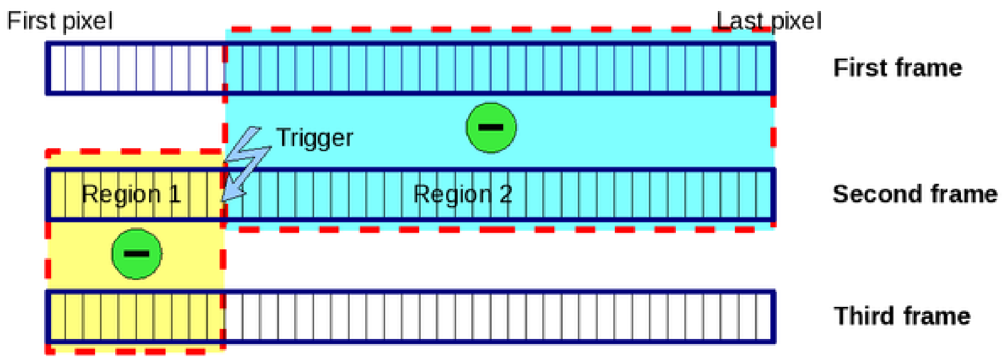

Figure 2: *Timing convention used for the CDS calculation when the DAQ system is working in RAW3 mode.*

time,

2. the following part (**region 2**) made by pixels that are readout after the trigger.

A priori, it is impossible to know where the particle is passing through the detector and consequently both regions have to be analyzed separately. If the particle is on region 1, the produced signal will pop up only at the next reading, so during the first part of the third frame (yellow part). Instead, if it is on region 2, the particle signal is going to be sampled and then the second part of the first frame has to be used as reference signal. The off-line software has the duty to calculate the CDS and needs to know from the DAQ software the **pivot pixel**, i.e. the pixel being readout when the trigger arrived and the three frames. To properly calculate the CDS, one has to make the difference between the two yellow regions and the two blue ones.

2.2 Cluster search

The next step in the analysis procedure is the cluster search. A cluster is a group of nearby pixels having a signal over noise ratio high enough to be distinguishable from noise fluctuations. EUTelClusteringProcessor is the processor responsible for this operation: it is taking as an input a TrackerData object containing the data resulting from pedestal and common mode subtraction and outputting a collection of TrackerPulse with the found clusters and another TrackerData collection containing the pixel signal corresponding of the cluster. The importance of having two output collections is outlined in Sec. 3.2.
The EUTelClusteringProcessor can accept as input both full frame and zero suppressed data and apply to the two types of data different clustering algorithms. So far, some cluster search algorithms have been implemented and they are briefly described in the following sections.

2.2.1 Fixed frame clustering with RAW data

This is the easiest clustering procedure one can imagine. It is based on a two level SNR selection, the first level for the SNR of the pixel with the highest SNR (seed pixel) in the cluster, and the second for the overall cluster SNR. The user needs to provide the following information:

Seed pixel minimum SNR. This is a floating point number representing the minimum value of SNR that is required for a pixel to be considered as a seed pixel.

Cluster minimum SNR. This is a floating point number representing the minimum value of SNR that is required for a group of pixels to be accepted as a cluster.

Cluster size along X and Y. These are two odd integer number, representing the size of the cluster along the two directions in pixel numbers. Only odd numbers are accepted because in this cluster implementation the seed pixel has to be the central one.

The clustering procedure can be summarized as follow:

1. The input data matrix is scanned looking for seed pixel candidates. This means that a list of all pixels having a SNR in excess the minimum seed pixel SNR is prepared.
2. At the end of the pixel matrix scan, if the seed candidate list is not empty then this is sorted according to descending signal.
3. For each entry in the seed candidate list, a rectangular cluster candidate is built around it. The rectangular sizes are the one provided by the users.
4. The cluster candidate SNR is verified against the minimum cluster SNR. If it is accepted, then all pixels belonging to the cluster are flagged in order to avoid including them into another cluster.
5. The procedure is repeated for each element in the seed candidate list.

The output of the cluster search will be stored into a EUTelFFClusterImpl and this will be attached to a TrackerPulse object.

2.2.2 Fixed frame clustering with ZS data

The output of this clustering algorithm is again a EUTelFFClusterImpl in which some pixels can be missing. The procedure is very similar to the one used for RAW data a part from the very initial steps.

1. A vector of floats is created with as many components as the number of pixels in the matrix. All of them are initialized to zero. The 2D position of a pixel in the matrix is in one to one relation with the position in the vector.

2. The input zero suppressed pixel list is scanned. The signal of these pixels is copied in the right position of the vector. At the end of this process, the vector will contain zeros only in the positions corresponding to pixels not exceeding the signal threshold.

3. In the same loop on input zero suppressed data, a list of seed pixel candidates is filled comparing the pixel SNR to the user selected minimum seed SNR.

4. At this point, the same fixed frame clustering procedure for RAW data can be applied.

The output of this cluster search is a EUTelFFClusterImpl attached to a TrackerPulse object.

2.2.3 Sparse clustering

On the contrary of the previous algorithms, this procedure is not based on a fixed cluster size and shape. It is instead based on a proximity and minimum signal requirements. The user has to provide the two following information:

Minimum SNR. This value represents the minimum signal to noise ratio for a pixel to being added to a cluster.

Maximum distance. When building the cluster only pixels being close enough are merged together. This user defined value (in pixel unit) corresponds to the maximum distance two pixels can be displaced into a cluster.

The working principle is based on two loop cycles.

1. The **main loop**, performed on all the pixels in the zero suppressed list, is aimed to find pixels above the user–defined SNR threshold.

2. The **vicinity loop**, started every time a pixel over threshold in the main loop is found, is looking for pixels above the SNR threshold and close enough to the initial one.

The output of the procedure is, as usual, a pair of collections: one TrackerData (redefined as a EUTelSparseClusterImpl) and a TrackerPulse.

2.3 Cluster selection

After the cluster search is performed, specific cluster selection criteria can be applied using the EUTelClusterFilter processor. Clusters can be selected according to all their characteristics as: signal, signal to noise ratio and noise. Moreover, for signal and SNR, the calculation can be limited to a subset of pixels belonging to the cluster. This processor is very useful especially when the performance of the system has to be compared against different selection criteria.

2.4 Cluster center and hit maker

Once a cluster is found, then its center has to be evaluated. There are several different ways discussed in literature to calculate the cluster center. Two procedures have been implemented so far:

Charge center of gravity. The cluster center is calculated as the signal weighted average position.

ETA function correction. The cluster center is initially calculated using the charge center of gravity and then corrected with a non linear weighting function called η function. The main goal of this correction is to flatten the cluster center 2D distribution within a pixel.

Once the cluster center is found, this can be translated from pixel unit in the sensor local frame of reference to a space point in the global telescope frame of reference. The required conversion factor, such as plane positions and pixel pitches, are taken from the GEAR description of the telescope system. The output of the hit maker processor is a standard TrackerHit collection with a reference to the original cluster information.

2.5 Analytical track fitting

Scattering of beam particles in telescope sensor planes, DUT or other material layers, has to be taken into account for low energy running. With the possible distances between telescope planes of the order of 10–100 mm and the scattering angles up to about 0.3 mrad (expected for $680\mu m$ Si layer with 3 GeV electron beam), the expected track displacement due to scattering can be of the order of micrometers. It can be larger than the expected position resolution of the telescope sensor layers. Therefor it is crucial to take multiple scattering into account in the track fitting algorithm. Dedicated track fitting method was developed for the EUDET beam telescope, following the analytical approach described in [2]. The approach is based on few simplifying assumptions:

- all telescope planes are parallel to each other
- the incoming beam is perpendicular to the telescope planes and has a small angular spread
- particle scattering angles in subsequent telescope layers are also small
- thicknesses of all material layers are very small compared to the distances between planes
- particle energy losses in telescope layers can be neglected

With these assumptions the problem of finding track position in each telescope plane by searching minimum of χ^2 function can be reduced to solving of a matrix equation. Moreover, track fitting can be performed separately in XZ and YZ planes (where Z is defined along the beam axis direction).

For use in the EUDET Telescope data analysis fitting method described in [2] had been implemented in the dedicated Marlin processor called **EUTelTestFitter**. The track fitting procedure consists of the following steps:

1. measured space points calculated in hit maker processor are read from input *TrackerHit* collection and copied to local tables.
2. lists of hits are created for each active sensor plane. If required, cuts on hit positions can be applied.
3. fitting procedure is finished if not enough planes are fired.
4. list of fit hypotheses is defined, based on the number of hits reconstructed in subsequent telescope planes. Each hypothesis corresponds to the unique set of hits (one hit per plane). Hypotheses with missing hits (no hit in one or more planes) are also considered.
5. the list of fit hypotheses is scanned to find the one with best χ^2 value.[2] For hypotheses with missing hits corresponding "penalties" are added to χ^2 values.
6. the best hypothesis is accept if the χ^2 value is below threshold, otherwise the fitting procedure is finished.
7. fitted track is written to the output *Track* collection. Measured particle positions corrected for alignment and fitted positions can also be written out as *TrackerHit* collections (options available via steering parameters).
8. hits used in the accepted hypothesis are removed from the hit list and the fitting procedure is repeated from step 3.

Fit procedure uses the geometry information taken from GEAR. However, additional changes to the geometry description (alignment corrections, removing layers from the fit) can be applied with dedicated steering parameters.
Shown in Figure 3 is the χ^2 distribution obtained from the the described track fitting procedure for the telescope test data. Telescope with 5 sensor layers was exposed to 3 GeV electron beam at DESY in August 2007. For comparison, χ^2 distribution obtained from the straight-line fit to the same data is also shown. When multiple scattering is taken into account, χ^2 distribution is very close to the one expected for 8 degrees of freedom.[3]. For straight-line fit the χ^2 distribution is much wider and has a long tail

[2]Calculating χ^2 value for all fit hypotheses is very time consuming. However, if hypotheses are properly ordered, the procedure can be significantly optimized. This is based on the observation that hypothesis can be rejected without χ^2 calculation, if any subset of hits resulted in χ^2 value above the threshold.

[3]We fit 10 parameters (X and Y position in each telescope layer) to 10 corresponding measurements. However, there are in addition 8 constraints on the particle scattering angles in X and Y in all but last layer. Constraint on the scattering angle in the first layer can be imposed, if we take into account small angular spread of the incoming beam.

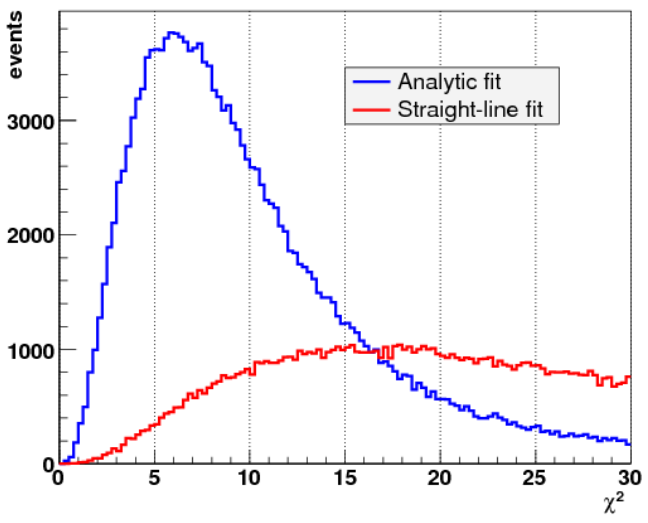

Figure 3: χ^2 *distributions obtained from the telescope test data. Results based on the analytical track fitting (blue line) are compared with the straight-line fit (red line). Telescope with 5 sensor layers was exposed to 3 GeV electron beam at DESY.*

towards large values. With the same track quality requirement, $\chi^2 < 30$, about twice as much tracks are reconstructed with the analytical approach than with the straight-line fit.

The effect of the multiple scattering is also clearly visible when we consider precision of the particle position determination. It was studied by comparing the position measured in one of the telescope layers (which was treated as DUT) with the position expected from the fit to the remaining four layers. From the observed difference between the measured and fitted position we can estimate the fit quality. This is shown in Figure 4, for telescope consisting of 5 sensor layers, exposed to 3 GeV electron beam, with middle plane used as DUT. When analytical fit method is used, the observed width of the distribution, as obtained from the Gaussian fit within $\pm 2\sigma$, is about 5.2 μm. This is in very good agreement with the fit error of 4.2 μm expected from analytical calculations, taking into account the estimated position resolution in single sensor of 3 μm. Also shown in Figure 4 is the distribution obtained for the straight-line fit, with weaker χ^2 cut, $\chi^2 < 100$, to get similar number of tracks. The width of the distribution increases to about 7.8 μm. We can estimate that the straight-line fit accuracy in the considered setup is about 7.2 μm. With tight χ^2 cut, $\chi^2 < 30$, precision of about 6.1 μm can be obtained, but the number of fitted tracks is reduced by over 30%.

Analytical track fitting method implemented in the EUTelTestFitter processor of Marlin turned out to be very useful for analysis of low energy data. Also the implemented track searching algorithm turned out to be very efficient. With high intensity hadron beam at CERN up to 100 tracks were found per event.

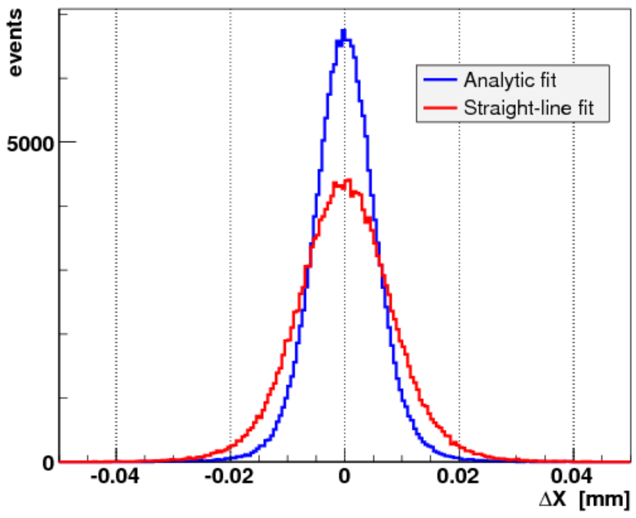

Figure 4: *Difference between the position measured in the middle telescope layer and the position fitted to the remaining four layers. Results based on the analytical track fitting (blue line) are compared with the straight-line fit (red line). Telescope with 5 sensor layers was exposed to 3 GeV electron beam at DESY.*

To preserve code modularity only very basic diagnostic histograms are included in EUTelTestFitter (χ^2 distribution, number of fitted tracks, etc.). More advanced analysis of the track fitting results can be done with two accompanying processors called **EUTelFitHistograms** and **EUTelDUTHistograms**, dedicated for study of telescope sensor performance and DUT performance, respectively. In addition to signal distributions or efficiency histograms, dedicated histograms for alignment verification, allowing for alignment corrections calculation are also included.

2.6 Grid usage

For large datasets taken during test beam measurements it is important to parallelise the processing. A possible solution is to run the analysis software on the Grid. For this purpose a PERL script has been developed to automatise the job submission. It uses the commands provided by the gLite middleware [3]. All steps described above can be executed on the Grid.

2.7 Alignment

Before tracks can be reconstructed it is important to align the sensors of the telescope. A simple alignment procedure has been implemented for this purpose. In the future also a more advanced alignment routine based on the Millepede II package will be available. Both approaches are described in the following:

2.7.1 Simple alignment

A simple alignment processor called **EUTelAlign** has been developed. The method was inspired by the alignment procedures used for the ZEUS strip telescope [4].

Here the reference system is defined by the center of the first telescope sensor. Furthermore a parallel mounting perpendicular to the beam direction z is assumed.

In general, the misalignment of each sensor can be decribed by three shifts alont the x, y amd z axis off_x, off_y and off_z and by three rotations θ_x, θ_y and θ_z aroud these axis. The parameter off_z has to be determined from mechanical measurements while the other parameters are obtained by the alignment processor.

The positions of the hits (except for the first plane) are given by:

$$x = (\cos\theta_y \cos\theta_z) \cdot x_{meas} + (-\sin\theta_x \sin\theta_y \cos\theta_z + \cos\theta_x \sin\theta_z) \cdot y_{meas} + \text{off}_x \quad \text{and} \quad (1)$$

$$y = (-\cos\theta_y \sin\theta_z) \cdot x_{meas} + (\sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta_z) \cdot y_{meas} + \text{off}_y. \quad (2)$$

Here x_{meas} and y_{meas} are the measured positions in a given sensor.

To derive the alignment constants the distance between the predicted position of a particle from the first plane x_{pred} and y_{pred} to the position in one of the following planes is minimised. The predicted positions are taken from a linear extrapolation parallel to the beam from the first plane. The following function is minimised:

$$\chi^2 = \frac{(x - x_{pred})^2 + (y - y_{pred})^2}{\sigma^2}, \quad (3)$$

where σ is the error of the reconstructed position. The fit is done using the implementation of the MINUIT program [5] included in the ROOT framework [6].

All sensors are aligned separately. The minimisation is done in three steps:

- First, the rotations are neglected and only the two shifts are treated as free parameters. The result is used as starting point for the next step.
- In a second step also the angles are included in the fit. Again, the result defines the starting point for the following step.
- Hits with a very large contribution to χ^2 are finally excluded from the fit. This cut removes outliers and hence improves the quality of the result.

It is possible to align only the telescope sensors before the DUT and then fit straight line tracks using these sensors with the **EUTelAlign** processor. The tracks from the aligned sensors can then be used to align the rest of the telescope. This was motivated by the setup used for the test beam measurements which were performed at CERN in September 2007. Here the sensors in the first box were aligned as discribed above. Tracks from the aligned sensors in the first box were afterwards used to align the sensors in the second box.

2.7.2 Alignment using Millepede II

An alignment procedure of the telescope based on the Millepede II program [7] is developed at the moment. This approach is based on fits of full tracks. Parameters are grouped into local and global parameters. Local parameters are only present in subsets of the data. Here the global parameters are the alignment constants. The linear least sqares problem is solved by a simultaneous fit of all parameters. This is not depended on the number of local parameters. Thus a large number of tracks can be considered for the alignment.

3 Data model

3.1 Full (raw) matrix description

The best way to store the full pixel matrix information is to use either a TrackerRawDataImpl or a TrackerDataImpl depending if the pixel signal is an integer value of a floating point. All the pixel signals sorted from left to right from top to bottom are added to the short (float) STL vector of the TrackerRawDataImpl (TrackerDataImpl). The detector number and the matrix boundaries are instead embedded into the cellID through the use of a MATRIXDEFAULTENCODING. To obtain the 2D position of a pixel in the matrix, a specific helper class (EUTelMatrixDecorder) has been implemented.

3.2 Cluster description

A cluster is a group of pixels passing a list of selection criteria based on their signal, signal to noise ratio and / or proximity. In the LCIO data model there are two “natural” ways to describe a cluster:

TrackerPulse. This class can be used to describe general properties of a cluster, like the total charge, the time stamp and the cluster quality. Every TrackerPulse object can be linked to a TrackerData.

TrackerData. This class is mainly composed by a vector of float that can contain all the pixels belonging to the cluster.

An abstract base class (**EUTelVirtualCluster**) implementing the decorator pattern around the TrackerDataImpl class has been defined. All possible implementations describing different kind of clusters (for example **EUTelFFClusterImpl** or **EUTelSparseClusterImpl**) are inheriting from EUTelVirtualCluster. This has the advantage that whatever cluster implementation the user is using, an EUTelVirtualCluster derived object can be attached to the TrackerPulse and used by any other processors.

4 Conclusion

EUTelescope is an analysis and reconstruction software coded by the JRA1 members to complement the test beam infrastructure being developed. EUTelescope has been successfully used for the reconstruction of all the tracks acquired during the current year data taking periods. Further developments are foreseen in view of the JRA1-11 milestone (month 36) concerning, in particular, a better handshaking with the DAQ system and an improved integration of device under test data streams.

Acknowledgement

This work is supported by the Commission of the European Communities under the 6th Framework Programme "Structuring the European Research Area", contract number RII3-026126. A.F.Ż. acknowledges partial support by Polish Ministry of Science and Higher Education, project no. EUDET/217/2006

References

[1] R.D. Heuer, D. Miller, F. Richard, P. Zerwas (eds.): *TESLA Technical Design Report*, Part IV", DESY 2001-011, ECFA 2001-209, 2001.

[2] A.F. Zarnecki and P. Niezurawski, "EUDET telescope geometry and resolution studies," EUDET-Report-2007-01, arXiv:physics/0703058.

[3] http://glite.web.cern.ch/glite/

[4] M. Milite: *The Internal Structure of Charm Jets in Photoproduction at HERA and Tests of the ZEUS Microvertex Silicon Sensors*, PhD thesis, Universität Hamburg (2001).
D. Contarato: *Silicon Detectors for Particle Tracking at Future High-Energy Physics Experiments*, PhD thesis, Universität Hamburg (2005).

[5] F. James: *MINUIT Function Minimization and Error Analys is Reference Manual Version 94.1*, CERN Program Library Long Writeup D506 (1998).

[6] R. Brun and F. Rademakers: *ROOT - An Object Oriented Data Analysis Framework*. In *Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996*, Nucl. Instr. Meth. A389, 81 (1997).

[7] http://www.desy.de/ blobel/mptalks.html