



## **ATLAS Test Beam Online Monitor development**

Michał Wysokiński

AGH University of Science and Technology

Kraków, Poland

Work done in the context of 2012 DESY Summer Student Programme

Under the supervision of

Igor Rubinsky

October 4, 2012

### **Abstract**

DESY participates in many hardware development projects like the ATLAS and CMS upgrades or the ILC construction. The facility offers a few test beam lines for the purpose of testing new cutting edge detector systems. In order to acquire desired data from beam tests its quality needs to be monitored in real time. This report describes the work on a tool which meets this task, called “Online Monitor”. The performance of the tool was analysed and new optimisation possibilities were investigated. The new version of the application was tested in both online and offline mode and with different beam conditions.

# Acknowledgements

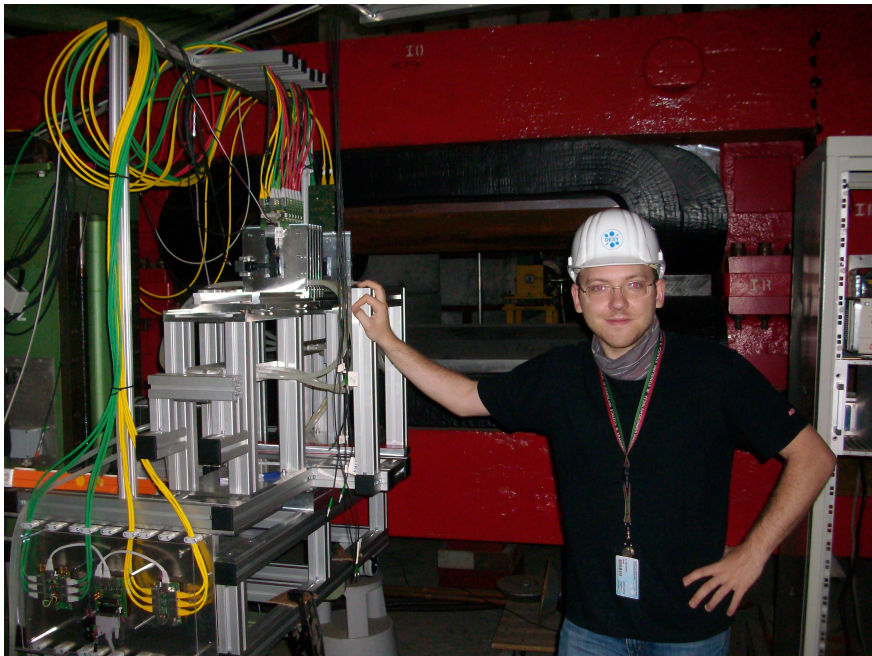
*I would like to sincerely thank my supervisor Igor Rubinsky for his time and effort he made to introduce me to this project and for the test beam experience where “one does not simply take data”.*

*I am also very grateful to the Organizers of the 2012 DESY Summer Student Programme for allowing me to work at DESY and participate in many great activities.*

*Finally, I would like to express my gratitude to all people who helped me during my work and without whom it would be much more difficult, especially:*

- *Marcel Stanitzki for his valuable comments about my presentation and key points for investigation,*
- *Ingrid Gregor for her comments and encouragement,*
- *Sabine Krohn for dealing with paperwork,*
- *DESY ATLAS group for their support.*

*Thank you very much.*  
*Michal*



# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>EUDAQ framework</b>	<b>4</b>
2.1	Run Control . . . . .	5
2.2	Producer . . . . .	5
2.3	Data Collector . . . . .	5
2.4	Logger . . . . .	5
2.5	Monitor . . . . .	5
<b>3</b>	<b>DATURA telescope</b>	<b>5</b>
3.1	General Overview . . . . .	5
3.2	MIMOSA26 sensors . . . . .	6
<b>4</b>	<b>Online Monitor</b>	<b>6</b>
4.1	Performance analysis . . . . .	6
4.2	Threshold dependency . . . . .	7
4.3	Functions . . . . .	9
4.3.1	Clusterisation . . . . .	9
4.3.2	Correlation . . . . .	10
4.3.3	Algorithms Optimisation Conclusion . . . . .	11
4.4	New Running Modes . . . . .	13
4.5	New Data Quality Monitoring plots . . . . .	14
<b>5</b>	<b>Conclusions</b>	<b>14</b>

# 1 Introduction

Test beam facilities from all around the world serve the purpose of testing new cutting edge detector systems. There is no better way to determine whether the detector is working or not than just putting it into an actual beam. In Europe there are two main test beam facilities: DESY and CERN. The DESY test beam can consist of either electrons or positrons with energy up to 6 GeV. The CERN's facility is capable of creating pion and muon beams with energy up to 180 GeV. Test beam setups are designed to be simple and easy to use by users who just want to take data and not to bother with the setup. In order to achieve this many frameworks have been developed and one of them called EUDAQ will be described in the next chapter.

## 2 EUDAQ framework

EUDAQ is a data acquisition framework that was developed as part of the EUDET project. It is written in C++, multiplatform and generic in a sense that it is independent from the hardware layer. The general architecture of the framework is shown on the picture below and followed by its description.

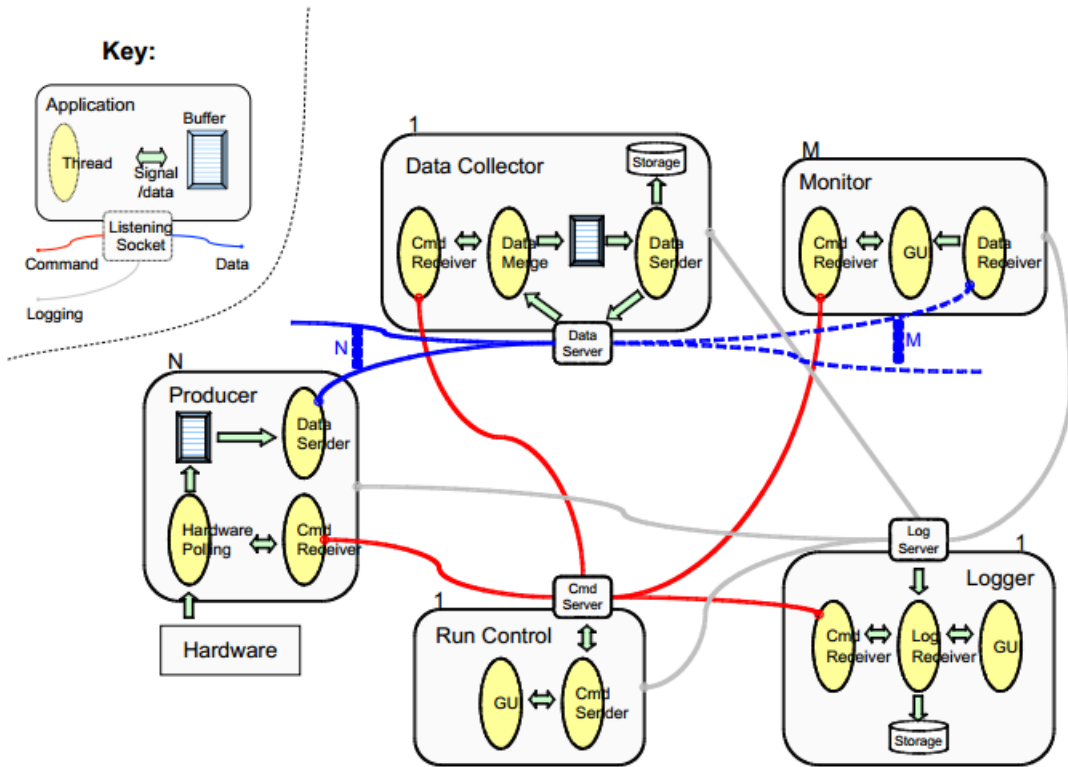


Figure 1: EUDAQ architecture

All elements described below are independent software processes with different tasks to accomplish.

## **2.1 Run Control**

Run Control is a process that controls every other process. It takes commands from a user and passes them to the selected unit.

## **2.2 Producer**

Producers are directly connected to each piece of hardware that produces data like a telescope or a device under test (DUT). It waits for the data, reads it, builds it and sends it to the Data Collector.

## **2.3 Data Collector**

Data Collectors receive all data sent by the Producers, combine it and create raw files for storage. Output format is configurable and compatible with some other data acquisition frameworks.

## **2.4 Logger**

Logger unit keeps track of all log messages sent by other processes and stores them. It allows to retrace all steps made during data taking.

## **2.5 Monitor**

This process is a tool for monitoring hardware status. It is supposed to be working in real-time and also show data quality information. In order to achieve this it needs to analyse data which is being taken and display its flaws as soon as possible. Because of that, it is an essential part of the entire processing chain.

# **3 DATURA telescope**

## **3.1 General Overview**

DATURA telescope is a telescope operated at DESY. It is being used mainly by the ATLAS and CMS groups and is compatible with the EUDAQ framework. The telescope consists of six MIMOSA26 sensor planes which will be described in the next section. The main task of the telescope is to register tracks of ionizing particles passing through the sensors and a DUT. Combined data from the telescope and the DUT allows to study DUT properties like the resolution, sensitivity and noise.

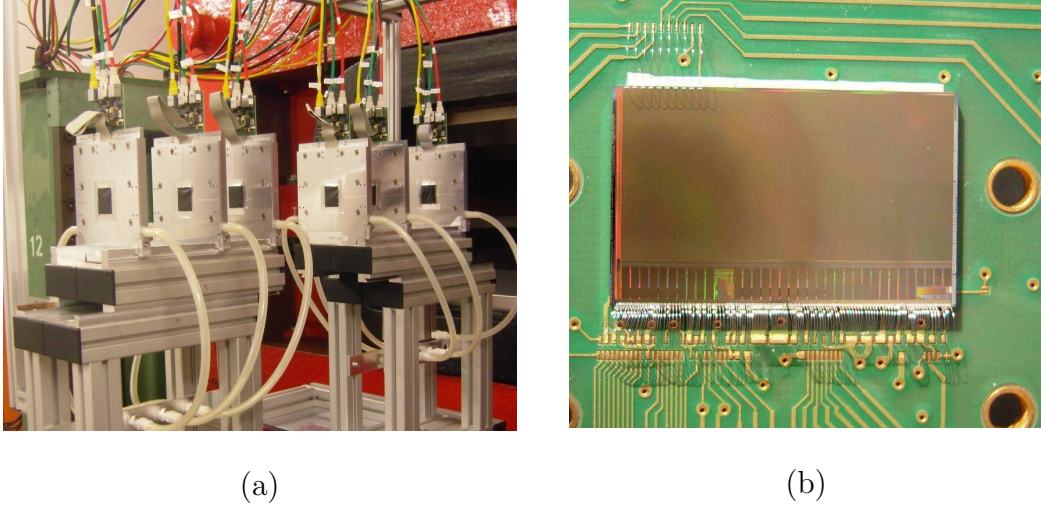


Figure 2: (a) DATURA telescope at DESY, (b) MIMOSA26 sensor

### 3.2 MIMOSA26 sensors

The sensors are placed perpendicular to the beam and placed on the same height. Any deviation in their positions is later suppressed by the procedure called the alignment. Each sensor is composed of a matrix of pixels which registers particles leaving charge in them. The MIMOSA26 sensor has 1152x576 pixels and reads all of them within 100  $\mu\text{s}$ .

## 4 Online Monitor

Online Monitor is an application in the EUDAQ framework that serves the purpose of online data quality monitoring. It analyses data taken by the telescope and a DUT and creates plots and histograms showing data properties like a raw hitmap, cluster distribution, hot pixel map and many more. In order to allow beam test users to react in case of any data flaws the tool should operate in real-time, which was not the case at the beginning of this project.

### 4.1 Performance analysis

The very first thing that was done in order to get the information about the application performance was a profiling analysis. The profiling analysis shows how much time is spent in different functions and how many times they are being called. It pointed a direction in which the development should go and functions which had to be improved. It also suggested an interesting dependency which was not foreseen.

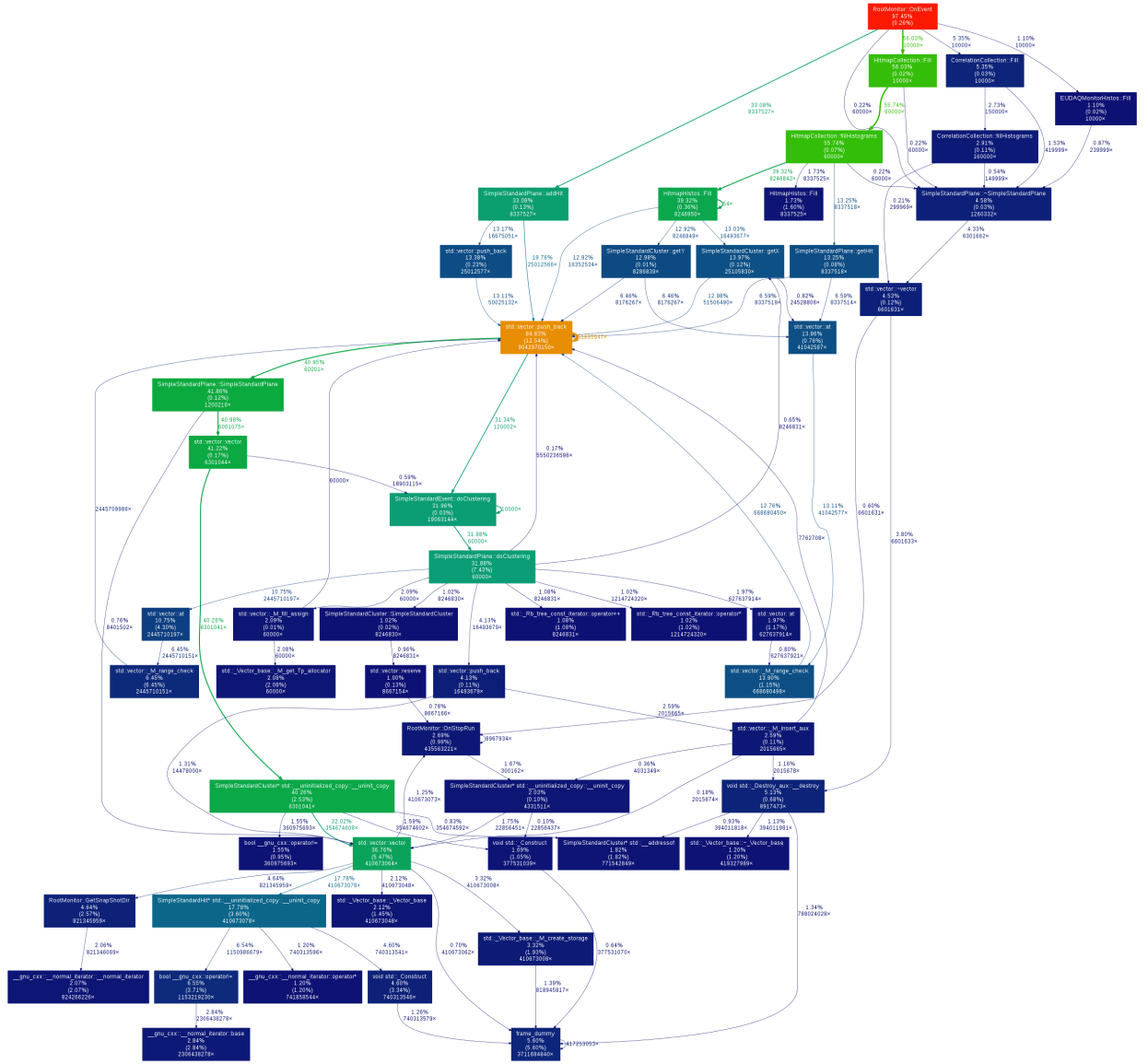


Figure 3: Callgraph created for the performance analysis

## 4.2 Threshold dependency

The results suggested that the performance of the application depends strongly on a set of data that is being supplied. This dependency is caused by different threshold (signal to noise ratio) settings during data acquisition and is a consequence of a total number of points that needs to be analysed. Lower thresholds consist of more noisy pixels and in consequence more data to be analysed, meanwhile higher thresholds suppress noise and decrease the number of points to analyse. This relation is show in the Figure 4. The red line shows the region where the processing time should in order to deliver a real time experience.

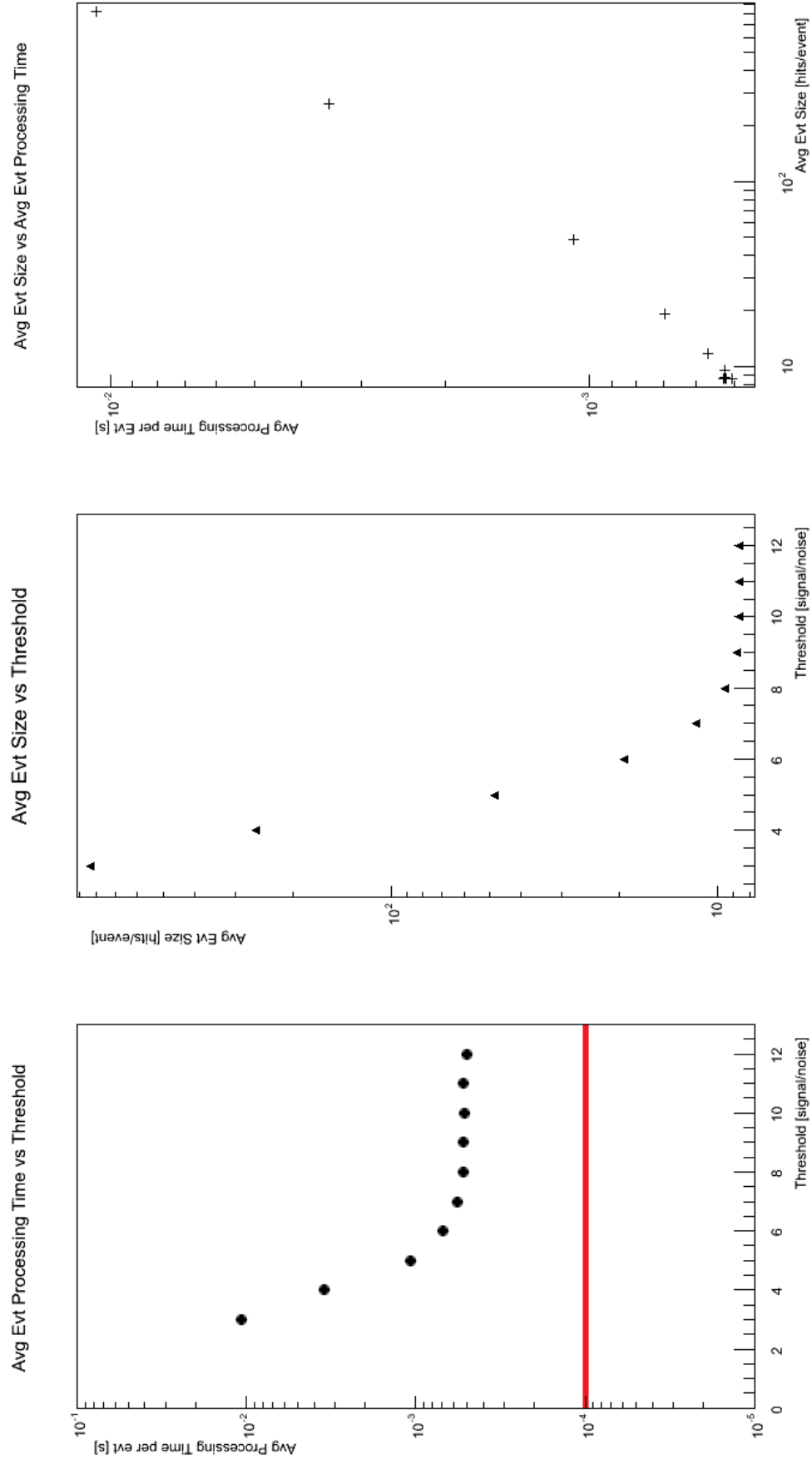


Figure 4: Results from the threshold dependency analysis

Because of these results all used beam conditions were taken into consideration and significant differences were found. For the lower thresholds raw hitmap filling and clusterisation were crucial for the performance, however for the higher thresholds correlation was the most time consuming operation. It was also discovered that a lot of time is spent on the STL Vector operations. However it was not possible to change this behaviour without rewriting the entire application, which was not the case.

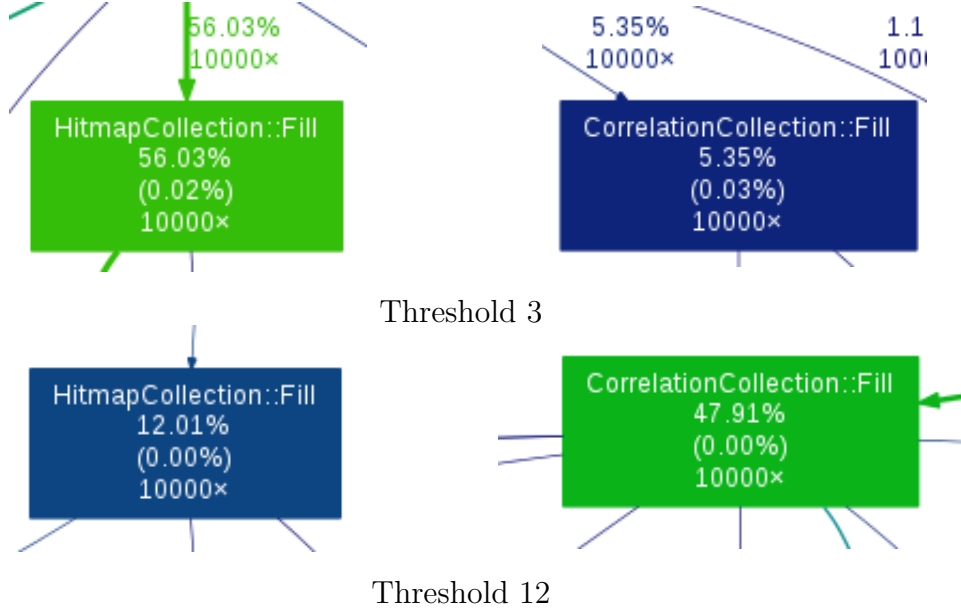


Figure 5: Selected results from the profiling analysis

## 4.3 Functions

The preliminary analysis shown in the previous chapter suggested that the two algorithms needed to be investigated in details were the clusterisation and the correlation algorithms. The approach to optimising them along with the short description on how they work will be described in this chapter.

### 4.3.1 Clusterisation

When a particle is passing through a sensor it leaves a certain amount of charge, subsequently this charge is distributed amongst pixels and this group of pixels is called a cluster. The clusterisation algorithm finds pixels belonging to the same cluster and labels them.

The first version of the algorithm had the complexity of  $\theta(n^2)$ . It is because all hits from the plane had to be compared with each other. The new version of that algorithm is a little bit more clever. It first sorts all the hits by  $X$  and  $Y$  coordinates and then looks for cluster candidates in the closest proximity. The complexity is estimated to  $\theta(n \log(n) + n)$ .

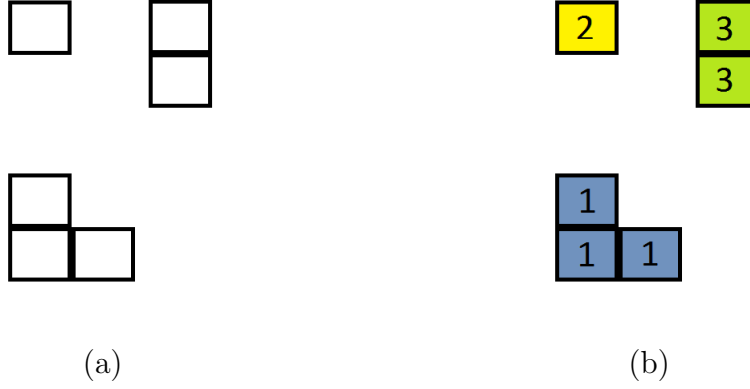


Figure 6: (a) Raw hits, (b) Labeled clusters

After improving this algorithm another profiling analysis was made and it showed that for the usual thresholds (6-8) the gain in performance is not very noticable (below 10%).

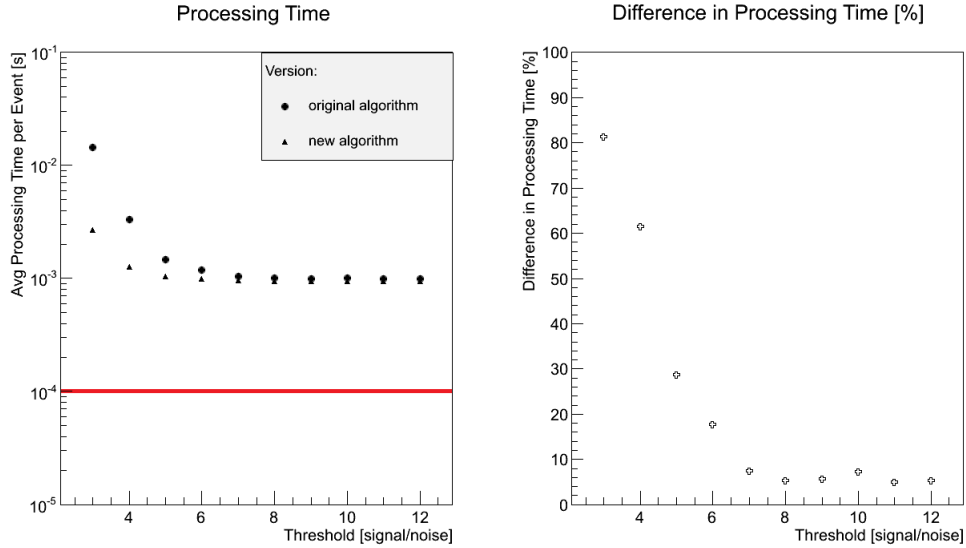


Figure 7: Profiling analysis for the new clusterisation algorithm

#### 4.3.2 Correlation

The correlation algorithm creates correlations between clusters from different sensor planes. The main task of the correlations is showing if beam is present inside the telescope and if all planes are well positioned.

The first version of the algorithm had the complexity of  $\theta(n^2)$ . It was because all clusters from every two planes had to be compared with each other and for all of them correlation

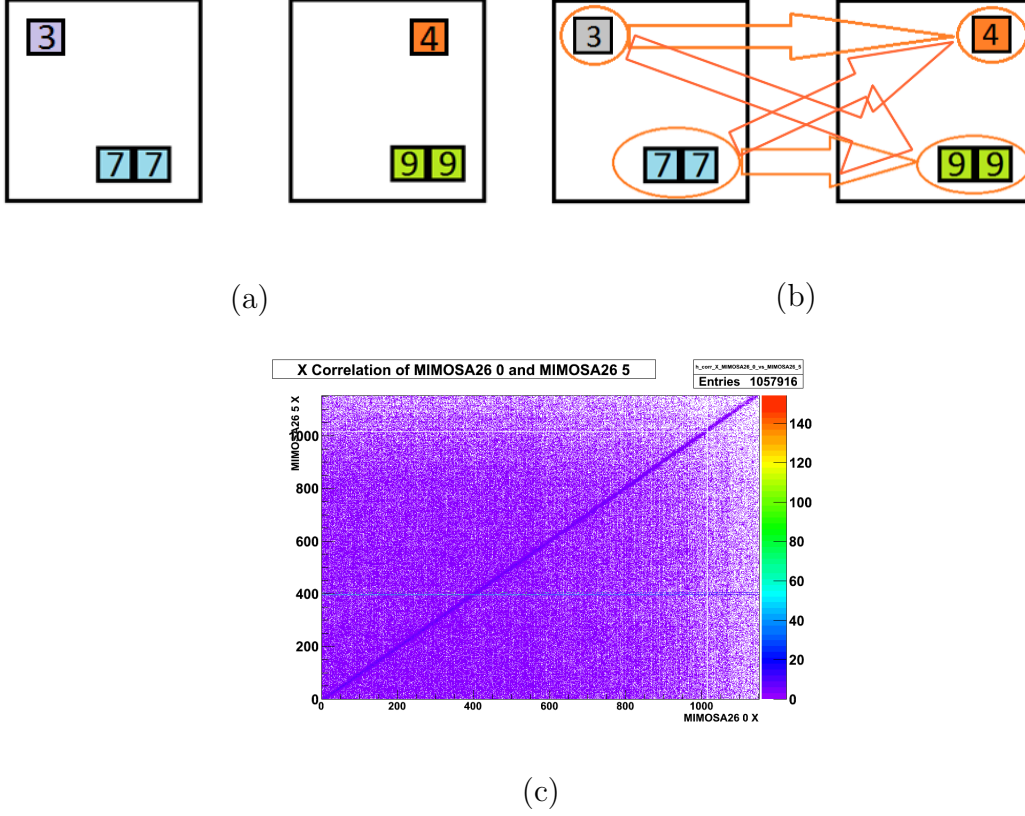


Figure 8: (a) Raw clusters, (b) Correlated clusters, (c) Old correlation histogram

was created. It also resulted in an unreadable correlation histogram, which can be seen above.

The new version of the algorithm uses track reconstruction to correlate all clusters. It first reconstructs all tracks (track is defined as a coincidence of hits in a defined window registered with at least three sensor planes) and then uses this information to create the correlations. The algorithm is not only faster than the previous one but also extracts information about a number of tracks going through the sensors.

#### 4.3.3 Algorithms Optimisation Conclusion

Even though two new algorithms were developed the gain in the performance was not as significant as it was needed. Because of that a “brute force” solution to skip the events was chosen.

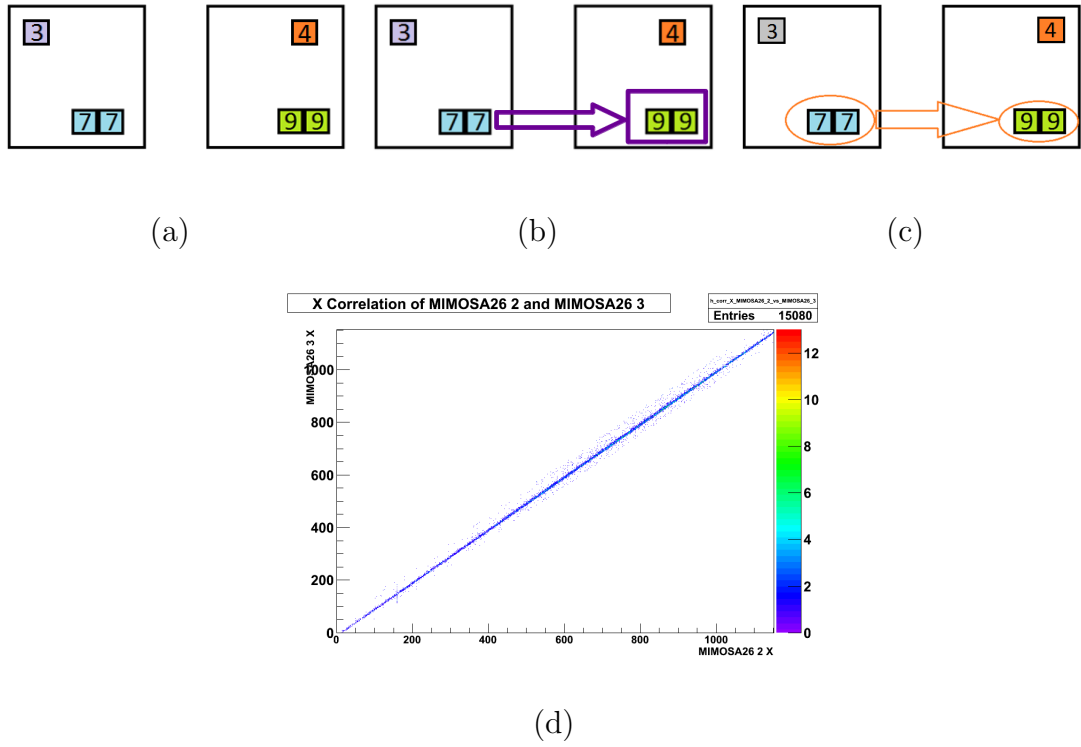


Figure 9: (a) Raw clusters, (b) Looking for clusters in the window, (c) Correlated clusters, (d) New Correlation Histograms

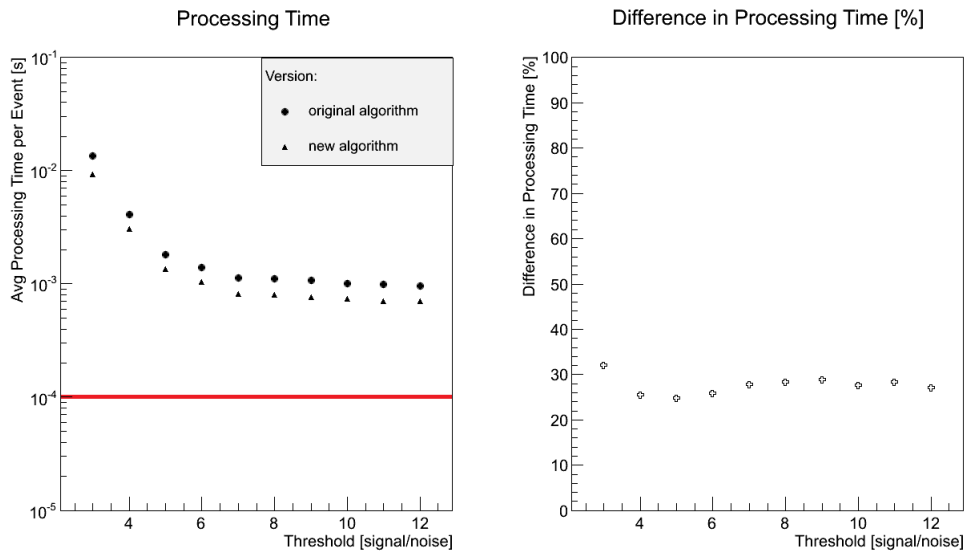


Figure 10: Profiling analysis for the new correlation algorithm

## 4.4 New Running Modes

Two new running modes were introduced in order to deliver a real time experience. The first one allows to skip a certain percentage of events taken during the analysis. The second one allows to skip a certain number of events per every event taken. With those new modes the region of computing frequency reached  $kHz$  regime what means that the Online Monitor is now able to keep up with beam conditions reaching a few  $kHz$ .

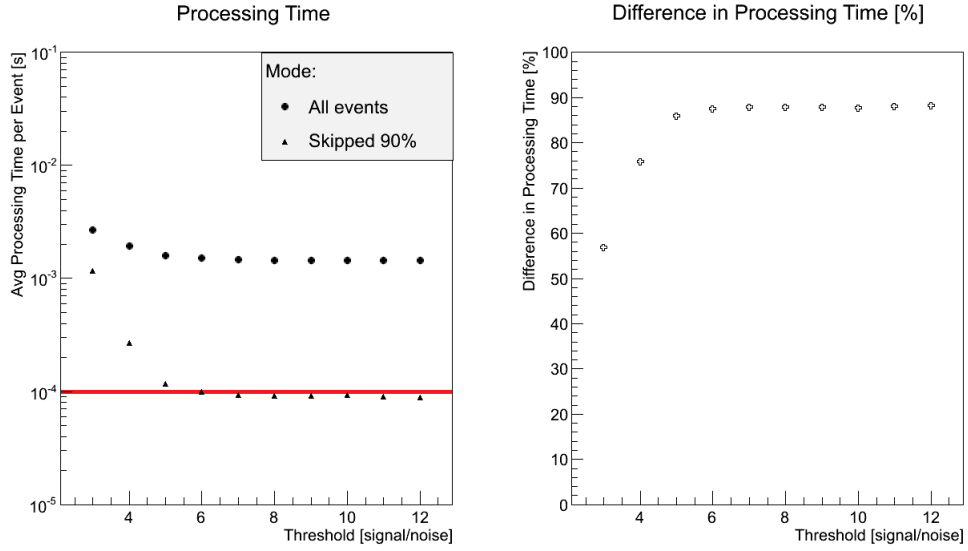


Figure 11: Profiling analysis for one of the skipping modes

## 4.5 New Data Quality Monitoring plots

Because of the new correlation algorithm it is also possible now to store the information about a number of tracks per event.

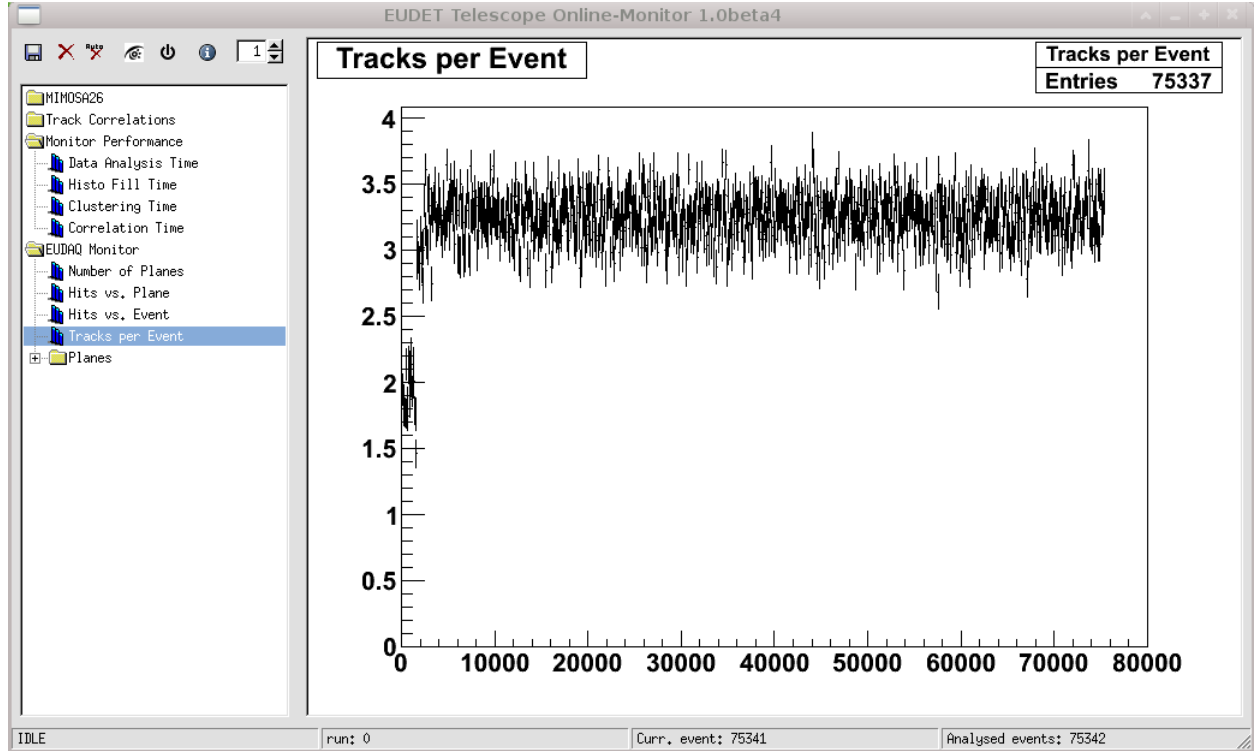


Figure 12: New histogram with information about tracks per event

## 5 Conclusions

New algorithms and functionalities for the Online Monitor were developed. It finally operates in real-time and hopefully will be a better tool. However the work on the tool is not finished. It needs even more work to be a better Data Quality Monitoring application. Hopefully the modifications made during this project will prove useful in time.