Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Bachelor's Project

Interactive visualization system for hybrid active pixel detectors within the ATLAS experiment at CERN

Petr Mánek

Supervisor: Ing. Stanislav Pospíšil, DrSc.

Study Programme: Open Informatics

Field of Study: Computer and Information Science

March 21, 2016

Aknowledgements

Zde můžete napsat své poděkování, pokud chcete a máte komu děkovat.

Declaration

Abstract

Translation of Czech abstract into English.

Abstrakt

Abstrakt práce by měl velmi stručně vystihovat její obsah. Tedy čím se práce zabývá a co je jejím výsledkem/přínosem.
Očekávají se cca 1 – 2 odstavce, maximálně půl stránky.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 About the Timepix Detectors

1.2 The Timepix Network at ATLAS

1.3 The Problem of Efficient Data Manipulation

1.4 Structure of This Document

Chapter 2

Data Structure and Storage

In this chapter, we introduce the reader to the process of data acquisition from Timepix detectors and describe the structure of the measured results. We then give details on the automated process which evaluates measured frames to produce additional statistics. We also briefly describe some of the storage formats used to archive such processed data, their advantages and disadvantages. Considering all these properties, we then propose a data scheme capable of archiving such data for longer time periods, while striving to offer almost instantaneous access based on the time of measurement.

2.1 Output Produced by Timepix

Similarly to photodetectors found in common digital cameras, Timepix detectors generate measurements in the form of individual frames. A single captured frame consists of values recorded by all pixels over a given time period, length of which is referred to as *the acquisition time*. Returning to our camera analogy, this figure resembles the time of exposition. Upon prolonging it, we can expect more particles to interact with our detector's pixels, making the resulting frames ultimately more saturated.

The technical principle behind the measurements is analogous to that of a Medipix sensor. Every pixel is equipped with an integer register called *the counter*. When acquisition starts, this counter is set to zero. Throughout the set time period, the counter is possibly incremented multiple times, and its value at the end is read out as measurement's result for the individual pixel. This process is synchronized across all detector's pixels, producing an integer matrix which constitutes the captured frame.

Since the pixels may not be identical due to material irregularities and manufacturing errors, every pixel has a *threshold* parameter, which is subject to calibration. If, during the measurement, the analog input measured from the pixel's semiconductor exceeds this threshold, the pixel is considered to be interacting with a particle.

2.1.1 Raw Output

Provided that every Timepix detector has 2 layers of 256×256 pixel matrices, every captured frame consists of 131,072 integer values. The interpretation of these values depends on another parameter, *the operation mode.* While it is technically possible to configure every pixel in a different mode, we have so far preferred to configure all pixels identically, making this essentially not a parameter of a pixel, but that of a frame.

The following operation modes are available:

Hit Detection Mode (One-Hit Mode) In this mode, the counter is set to one, when the theshold is exceeded. Upon multiple interactions, the counter is not further incremented. The result is a Boolean value, indicating whether the pixel has interacted with a particle.

Hit Counting Mode (Medipix Mode) In this mode, the counter is incremented upon every transition from state below the threshold to state above the threshold. The result is an integer value representing the number of particles which have interacted with the pixel.

Time over Threshold Mode In this mode, the counter is incremented by every clock cycle spent above the threshold. The result is an integer value corresponding to the energy of the interacting particle. Further calibration to convert counter value to energy is required, though.

Time of Arrival Mode In this mode, the counter is incremented by every clock cycle after the threshold is first exceeded. The result is an integer value corresponding to the time interval before the end of the measurement.

If a captured frame contains data from pixels configured in multiple different modes, the frame is said to be measured in the **Mixed Mode** and should contain further details on the exact pixel configuration.

2.1.2 Cluster Analysis

In our measurements, we strive to configure our detectors to capture frames containing disconnected components corresponding with the interacting particles. These components, referred to as *the clusters*, are discovered and evaluated in an automated process called *the cluster analysis.* This procedure involves a connectivity-checking algorithm, such as *flood-fill*, operating on the pixel matrices to distinguish individual clusters. In later stages, clusters are processed, measured and classified in various categories with regards to their shape. In addition, if the frame has been captured in TOT mode and calibration data is available, the automated processing script uses it to convert raw measured counter values to energy approximations.

The output of cluster analysis is a list of clusters. Every cluster may contain multiple non-zero pixels denoted by their planar coordinates. The original pixel matrix is not preserved. Instead, every pixel record is coupled with its corresponding counter value and pixels unreferenced by any cluster are assumed to be of zero value.

As we hinted at the beginning of this section, many other values are calculated for every cluster during the automated processing. Most notable examples are:

Shape Classification By measuring geometric properties of a cluster (such as radius or size), we are able to estimate whether the cluster resembles more a line segment or a circular blob. Similarly, we can also estimate whether the cluster looks thin or thick. From that information, we can infer the type of interacting particle and direction of its movement relative to the plane of incidence. To formally define the categories of cluster, we will use terminology consistent with the Medipix research.

Size, Volume The size of a cluster is equal to the number of connected pixels which constitute it. The volume is a sum of counter values of those pixels.

Centroid, Volumetric Centroid The centroid is defined as an unweighted average of pixel coordinates in the cluster. In analogous way, the volumetric centroid is the very same average weighted by corresponding counter values.

Minimum and Maximum Cluster Height These two figures refer to the lowest and the greatest counter values of pixels in the cluster.

Energy-based Properties *(available only in TOT mode)* If the energy approximations are available, many of the above-mentioned values can be also calculated with the energy substituted for counter values.

It is important to note at this point, that while not maintaining the original data structure, the process of cluster analysis does not lose any of the measured data. It merely reorganizes it into form which is more practical to work with. If at any point in time should we decide to alter some of the calculated statistics, it would be possible to reconstruct the measured data.

2.2 Common Storage Formats

2.2.1 The Single-Frame and Multi-Frame Formats

2.2.2 The ROOT Format

Another storage option is the ROOT Data Analysis Framework. Originally concieved at CERN in 1995, the framework provides a set of powerful tools with various applications in data mining, manipulation and visualization. Unlike other similar toolkits, ROOT comes with its own machine-independent binary file format (identified by the `.root` extension). This format is designed to store enormous amounts of data within various types of data structures efficiently, while maintaining good overall performance by employing low-level memory optimization techniques and multi-tier content caching.

Used by many physicists at CERN for several years now, ROOT seems like a good choice of a data archivation format as many researchers have already learned its caveats and know well how to operate it despite often lacking deeper background in Computer Science. For the purposes of programmatic access, ROOT also does well with documented APIs in Python, R and C++.

Should the processed data be stored in ROOT, a basic relational database concept comes to mind. With standard tables generalized in the form of *trees* and their columns in the form of *leaves.* One tree would suffice for the information about captured frames (such as acquisition time, operation mode, etc.) and other for the list of clusters for every frame. Such trees would efficiently abstract the entire storage structure, allowing for multiple frames to be stored in a single file, grouped for instance by a common time interval.

In spite of being over 20 years in development, ROOT is not perfect. Using memory monitoring tools such as *valgrind*, we have confirmed that the C++ implementation of the ROOT framework is riddled with various memory leaks, making it unsuitable for time-extensive operations. Some might also argue, that a full tree data structure might be overly-complicated and too general for a simple output described in previous sections. Lastly, ROOT framework has quite a complex object structure, making it hard to learn for first-time users.

2.3 Expected Volume of Acquired Data

2.4 Performance Optimizations

Chapter 3

Communication Protocol

3.1 Requirements

3.2 Underlying Standards

3.3 Web Methods

Chapter 4

Data Server

4.1 Role of the Application

4.2 Decomposition

4.3 Dependencies

4.4 Object-Oriented Design

4.5 A Note on Parallelism

4.6 Performance Optimizations

Chapter 5

Web Visualization

5.1 Naive Decomposition

5.2 Final Decomposition

5.3 Underlying Standards

5.4 Dependencies

5.5 Website Structure

Chapter 6

Conclusion

6.1 System Deployment

6.2 Data Import

6.3 Automating Data Acquisition

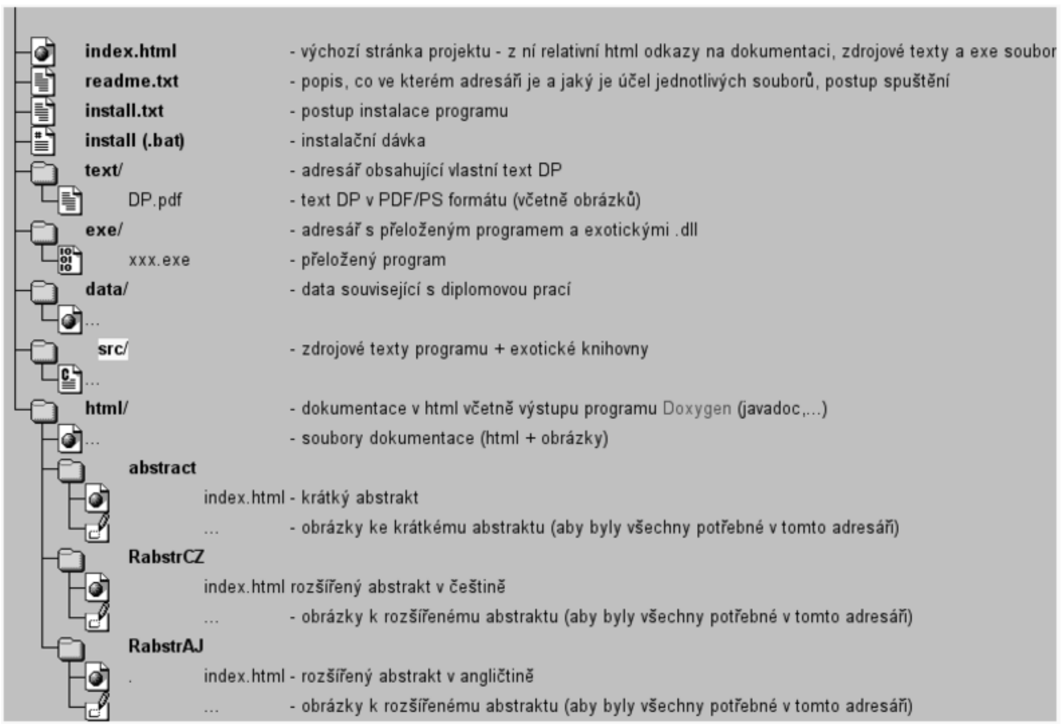
6.4 Future of the Application

Appendix A

Obsah přiloženého CD

Tato příloha je povinná pro každou práci. Každá práce musí totiž obsahovat přiložené CD. Viz dále.

Může vypadat například takto. Váš seznam samozřejmě bude odpovídat typu vaší práce. (viz [**?**]):



```
index.html              - výchozí stránka projektu - z ní relativní html odkazy na dokumentaci, zdrojové texty a exe soubor
readme.txt              - popis, co ve kterém adresáři je a jaký je účel jednotlivých souborů, postup spuštění
install.txt             - postup instalace programu
install (.bat)          - instalační dávka
text/                   - adresář obsahující vlastní text DP
    DP.pdf              - text DP v PDF/PS formátu (včetně obrázků)
exe/                    - adresář s přeloženým programem a exotickými .dll
    xxx.exe             - přeložený program
data/                   - data související s diplomovou prací
    ...
  src/                  - zdrojové texty programu + exotické knihovny
    ...
html/                   - dokumentace v html včetně výstupu programu Doxygen (javadoc,...)
    ...                 - soubory dokumentace (html + obrázky)
    abstract
            index.html  - krátký abstrakt
            ...         - obrázky ke krátkému abstraktu (aby byly všechny potřebné v tomto adresáři)
    RabstrCZ
            index.html rozšířený abstrakt v češtině
            ...         - obrázky k rozšířenému abstraktu (aby byly všechny potřebné v tomto adresáři)
    RabstrAJ
    .       index.html  - rozšířený abstrakt v angličtině
            ...         - obrázky k rozšířenému abstraktu (aby byly všechny potřebné v tomto adresáři)
```

Figure A.1: Seznam přiloženého CD — příklad