Studies on the track resolution difference between HLT1 and HLT2 at LHCb's RICH1 subdetector

Experimental Physics (EP) Department, LHCb Experiment,
European Organization for Nuclear Research (CERN)

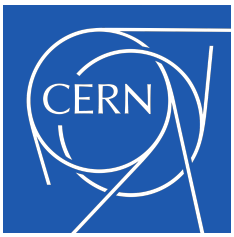
Summer Student Programme Report

Isaac Palma
National University of Costa Rica (UNA)

Supervisors:

- Ph. D. Saverio Mariani
- Ph. D. Sergio Arguedas

August 2024



1 Introduction

The Large Hadron Collider beauty (LHCb) is an experiment that is part of the Large Hadron Collider (LHC) at CERN and its primary objective is to study the decays of hadrons containing b or c quarks [1]. As shown in Fig. 1, it is composed of several subsystems [2], forming

- a tracking system, made up of a silicon pixel detector surrounding the interaction region (Vertex Locator, VELO), and tracking stations located upstream (Upstream Tracker, UT) and downstream (SciFi, Scintillating Fibers) of a dipole spectrometer;
- a particle identification system, made up of two Ring-Imaging Cherenkov (RICH) detector, an electromagnetic and hadronic calorimeter and muon stations.

As LHC delivers a collision rate of 40 MHz, a trigger system is designed to select the events of interest with the possible highest efficiency, as well as reducing the volume of processed information from 4 TB/s to 10 GB/s [3]. This trigger system, known as the High-Level Trigger (HLT), is composed of two stages: HLT1 and HLT2 [3]. It is worth to notice that in the LHC Run 2 period of data-taking, spanning between 2015 and 2018, another level of the trigger system (L0), based on hardware boards, was used upstream of HLT. In the current system, as its efficiency did not scale with the increased LHC luminosity [4], it was discarded and a completely software-based system for Run 3 [4] is adopted.

HLT1 must be fast enough to handle the collision rate of 40 MHz and only performs therefore on graphics processing units (GPU) a partial reconstruction of the collisions, including tracking and simple particle identification information, but not for example a full track fit including multiple scattering parametrization or the RICH reconstruction [5]. Increasing the amount of information available in HLT1, which would reflect on an improved selection efficiency, is hence of paramount importance.

In this document, studies related to include the RICH into HLT1 are documented. However, porting algorithms from HLT2, running instead on CPUs, to HLT1 is not straightforward, making it essential to understand the key differences between these stages in terms of both physics and computing architectures. In this case, the differences in the resolution of the reconstructed positions and directions for particles entering the RICH1 detector are compared, in order to understand how they are impacted by the HLT1 partial reconstruction.

Among the software tools used in the LHCb, Allen implements the HLT1 trigger system, and it was extensively utilized during this summer school. Allen is an application designed to run on GPUs, where it performs partial detector reconstruction and selection to achieve an output rate of about 1 MHz [6]. To use Allen, it must be compiled in one of the environments specified in the documentation [7], with the two main options being standalone or as part of the LHCb software stack. Compilation on CPUs is also supported. For this study, the standalone and the complete stack were both used.

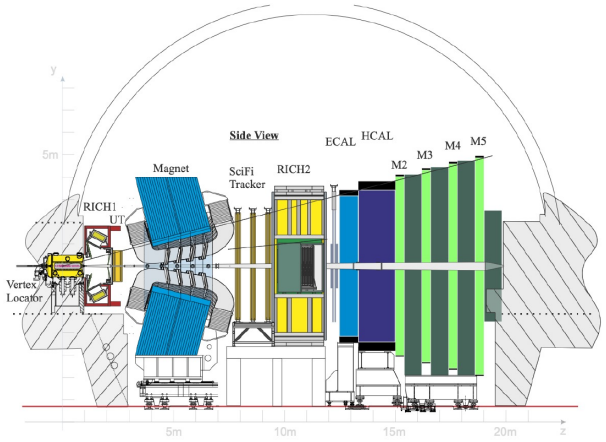


Figure 1: Sketch of the LHCb detector, with the main components indicated [2].

In Allen, the algorithms that process the data are organized into Python files called sequences. Typically, a sequence calls multiple device algorithms, which are written in C++ under NVIDIA's CUDA framework. It is helpful to access and visualize the data processed by these algorithms, so they are often accompanied by monitoring algorithms known as `output_monitor`. This document will present the process of enabling and adding monitoring algorithms, as well as exploring other parts of the stack, to analyze and access data generated by and related to these tools.

2 Monitoring algorithms

2.1 Creation

The development of a monitoring algorithm consists of the implementation of a function named `output_monitor`. This function will receive the fundamental arguments that form an Allen kernel, allowing it to access the processed data. These arguments are [8]:

- `arguments`, the input or output objects or variables of the algorithm;
- `runtime_options`, a singleton used to store configurable execution options and to provide interaction with ROOT for storing desired data;
- `context`, an abstraction representing the execution environment of the computations.

Once a function with this structure is declared, the monitoring service must be invoked. "Branches" are created for each variable that requires monitoring, to a structure known as a "monitor tree", the main component of the ROOT

files that results from calling a monitor algorithm. Values are then assigned to these branches, for example, by iterating over a list of elements and saving the attribute values for each element in the list. In this way, a file named `monitoring.root` will be created when a sequence calling this algorithm is executed, containing the stored data of interest.

2.2 Examples

To explore the different types of tracks in the LHCb, it is essential to understand some fundamental differences between them. This discussion focuses on a comparison between VELO and long tracks. The primary distinction lies in the detectors where the particles leave hits: for VELO tracks, this occurs only in the VELO detector, whereas for long tracks, it happens across all detectors, as illustrated in Fig. 2.

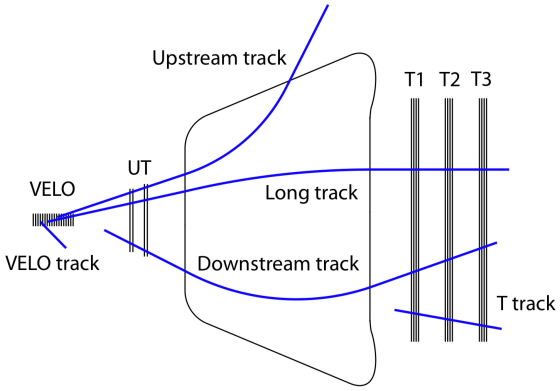


Figure 2: Track types at LHCb [9].

For VELO tracks, the algorithm taking into account the multiple scattering in the detector material includes a monitoring operating on VELO tracks. The monitored data include the x, y, and z coordinates at the point of closest approach (POCA) to the beam, as well as the slopes in $t_x = dx/dz$ and $t_y = dy/dz$, and the pseudorapidity $\eta = -\log(\tan(\theta/2))$, being θ the particle angle with respect to the beam axis.

Similarly, I developed a new monitoring algorithm for long tracks, as needed to understand their behavior to later study the resolution in the RICH1 subdetector. This algorithm monitors the same data as for the VELO tracks, at the beam POCA. The basic structure of this algorithm is as follows:

Algorithm 1 output_monitor for long tracks.

Variables: beamPOCA_x, beamPOCA_y, beamPOCA_z, beamPOCA_tx, beamPOCA_ty, eta, monitoring_tree, event_list
for each event, index **in** event_list **do**
    Let state be the state at index in event_list
    Let tx = state.tx()
    Let ty = state.ty()
    Let slope2 = tx × tx + ty × ty
    Let eta = eta_from_rho(rho)
    Store eta as eta in monitoring_tree
    Store tx as beamPOCA_tx in monitoring_tree
    Store ty as beamPOCA_ty in monitoring_tree
    Store state.x() as beamPOCA_x in monitoring_tree
    Store state.y() as beamPOCA_y in monitoring_tree
    Store state.z() as beamPOCA_z in monitoring_tree
end for

Algorithm 2 eta_from_rho. [10]

Parameters: rho
Let z = 1
if rho > 0 **then**
    Let big_z_scaled = 53.817371
    Let z_scaled = z / rho
    if |z_scaled| < big_z_scaled **then**
        return $\log(\text{z\_scaled} + \sqrt{\text{z\_scaled} * \text{z\_scaled} + 1})$
    else
        return $\log(2 * \text{z\_scaled} + 0.5/\text{z\_scaled})$
    end if
end if
return z + 22756

Fig.s 3 to 5 present some results of the VELO (in red) to long (in blue) track comparison. For the η, both positive and negative values are found for VELO tracks, which is not the case for the long ones.

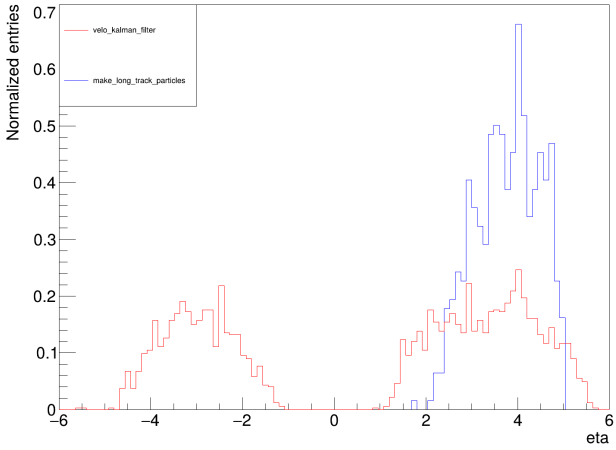


Figure 3: Normalized η distribution for VELO (in red) and long tracks (in blue)

As shown in Figure 3, the observed η distribution for VELO tracks includes both positive and negative values, whereas the long tracks exhibit only positive η values. This is motivated by the fact that the VELO is centered approximately at the origin of the reference system, $(x, y, z) = (0, 0, 0)$, with parts extending along both the positive and negative axes. This positioning allows for the reconstruction of negative η values and, as a consequence, of the proton-proton collision vertices. In contrast, long tracks utilize hits from the VELO, UT, and SciFi detectors, only covering positive η values. The x and y coordinates of the POCA to the beam are noted instead to follow very similar distributions.

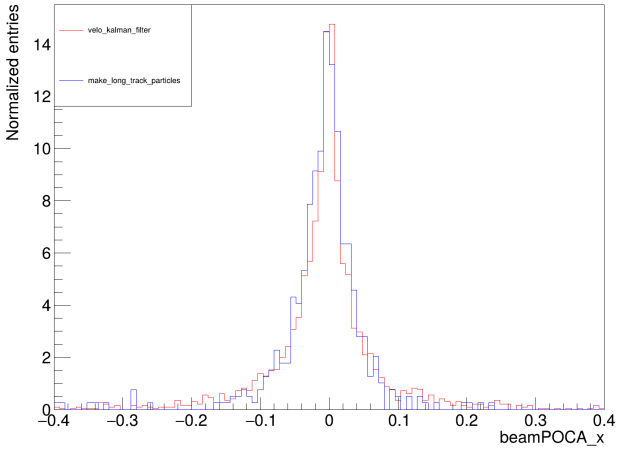


Figure 4: Normalized *beamPOCA_x* distribution for VELO (in red) and long tracks (in blue)

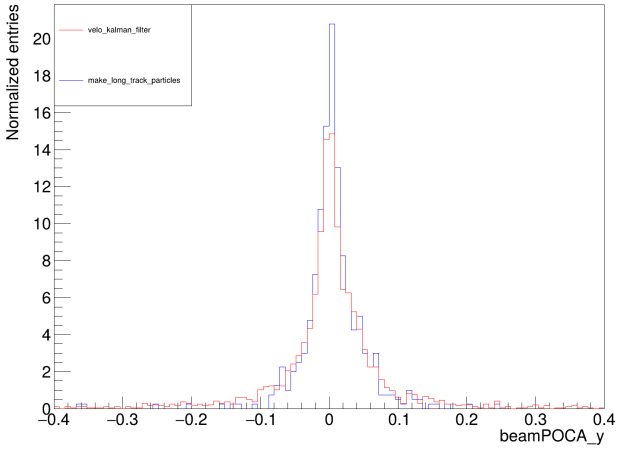


Figure 5: Normalized *beamPOCA_y* distribution for VELO (in red) and long tracks (in blue)

3 Results

In this section, the results of the resolution analysis for the reconstruction between the HLT1 and HLT2 triggers are presented. The analysis was carried out using the Rec tool, which is part of the LHCb stack, similar to Allen. Rec

consists of C++ algorithms that are executed through Python scripts, managed by another component of the stack called Moore.

For this analysis, the input data consisted of expected 2024 simulations. The process involved running the Rec algorithm, `TrackResChecker`, to determine the resolution for a specific track. This algorithm takes the selected track and simulated particle, constructs an extrapolated state, a state being a 5-dimension vector with x, y, t_x, t_y and $\frac{q}{p}$ at a given z, and subtracts it to the particle's "true" state to determine the reconstruction resolution. The calculations were performed at the entrance and exit points of the RICH1 subdetector.

To achieve this, the `TrackResChecker` algorithm was used twice: once for the HLT1 reconstruction algorithms and once for the HLT2 reconstruction algorithms. The input options for both cases were specified using the following files:

- For HLT1 and HLT2: `Moore/Hlt/Moore/tests/options/default_input_and_conds_hlt1.py`

The Python scripts invoking the Rec algorithms were:

- For HLT1: `Moore/Hlt/RecoConf/options/hlt1_reco_allen_trackresolution.py`
- For HLT2: `Moore/Hlt/RecoConf/options/hlt2_default_reco_track_master_fitter_with_mcchecking.py`

These scripts were executed with the command line tool `gaudirun.py`, steering the events loop, and the entire process can be run with the following command structure: `Moore/run gaudirun.py input_options_file reco_script`

For example:

```
Moore/run gaudirun.py Moore/Hlt/Moore/tests/options/default_
input_and_conds_hlt1.py Moore/Hlt/RecoConf/options/hlt1_reco_
allen_trackresolution.py
```

The results of this analysis are presented as a comparison of the resolution of the reconstructions for the x and y coordinates, at both the entrance and exit of the RICH1 subdetector. These graphs were generated using ROOT in conjunction with Python.

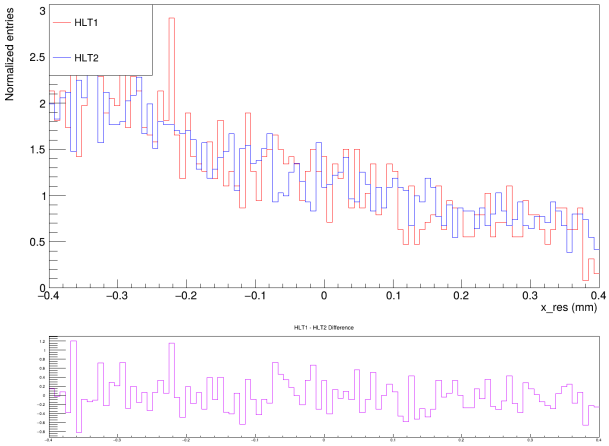


Figure 6: Normalized comparison of *x_res* for HLT1 (in red) and HLT2 (in blue) at RICH1 entrance / mm.

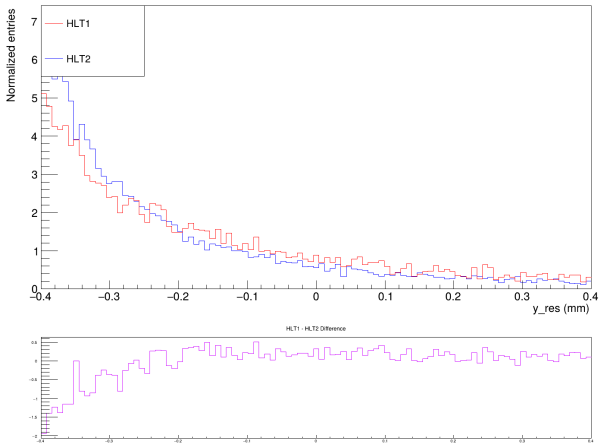


Figure 7: Normalized comparison of *y_res* for HLT1 (in red) and HLT2 (in blue) at RICH1 entrance / mm.

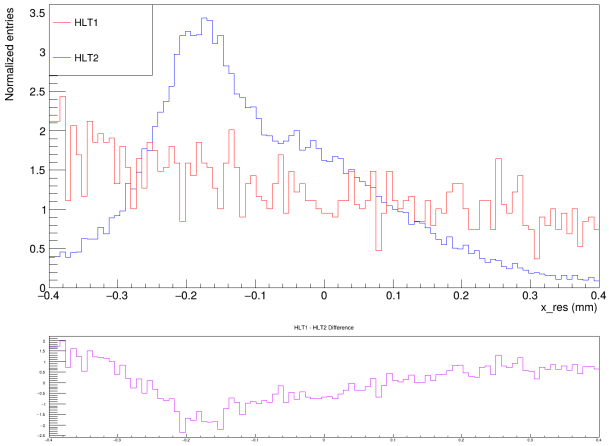


Figure 8: Normalized comparison of *x_res* for HLT1 (in red) and HLT2 (in blue) at RICH1 exit / mm.

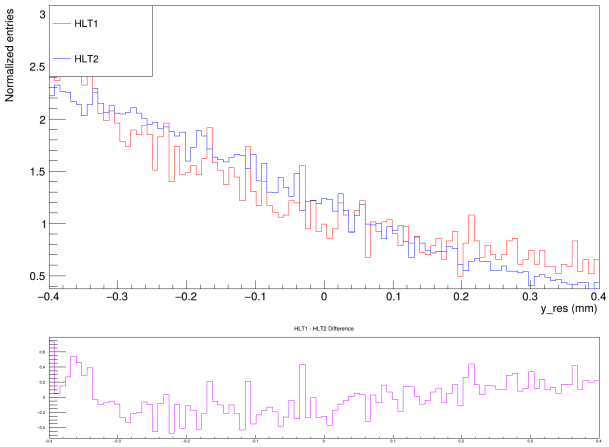


Figure 9: Normalized comparison of *y_res* for HLT1 (in red) and HLT2 (in blue) at RICH1 exit / mm.

In Fig.s 6, 7 and 9, it can be observed that the resolution measurements for the x coordinate at the entrance and the y coordinate at both the entrance and exit are very similar, with the differences between the two resolutions tending to be close to zero. This suggests that the reconstruction for these coordinates aligns closely with that of HLT2, indicating that the inclusion of RICH1 in

HLT1 could be possible, enhancing the particle selection efficiency in HLT1.

For the x coordinate at the exit (figure 8), it is observed that the distributions are not similar, showing a peak difference that is proportionally greater than the rest. This is noteworthy, as the function used to extrapolate and subsequently calculate the resolution of that extrapolation relative to the actual state of a particle is the same in both cases.

4 Conclusions

This report outlines the process of enabling and creating monitoring algorithms in Allen, a key component of the high-level trigger system of the LHCb experiment. Monitoring algorithms have been utilized and developed to analyze the behavior of various types of trajectories in the RICH1 subdetector, as well as to compare the resolutions obtained for different coordinates in the HLT1 and HLT2 reconstructions.

The results indicate that, for most of the studied coordinates, the resolutions of HLT1 and HLT2 are comparable, suggesting that the inclusion of RICH1 in HLT1 could be feasible. However, the observed discrepancy in the resolution of the extrapolation of the x coordinate at the RICH1 subdetector output highlights the need for a more detailed analysis of the extrapolation algorithms used to understand better the differences between HLT1 and HLT2 in this regard.

4.1 About the summer school

This summer school was an immensely enriching experience. I am deeply grateful to the institutions that made this opportunity possible, including my university, UNA, the National Council of Rectors (CONARE, the partner with CERN in Costa Rica), and CERN itself. I would also like to express my gratitude to my supervisors, Sergio and Saverio, for their attentive guidance throughout the program.

Furthermore, this school has sparked my interest in exploring areas where computer science can be applied beyond traditional corporate settings, contributing to the advancement of science. I look forward to the possibility of continuing to collaborate on projects like this in the future, particularly at CERN.

References

[1] Thomas Boettcher. *Allen: A High Level Trigger on GPUs for LHCb*. June 26, 2020. URL: https://indico.cern.ch/event/831165/papers/3717135/files/9918-boettcher_allen_CTD2020_proceedings_v2.pdf (visited on 08/07/2024).

[2] Roel Aaij et al. "The LHCb upgrade I". In: *JINST* 19.05 (2024). All figures and tables, along with any supplementary material and additional information, are available at http://lhcbproject.web.cern.ch/lhcbproject/Publications/LHCbProjectPublic/LHCb-DP-2022-002.html (LHCb public pages), P05065. DOI: 10.1088/1748-0221/19/05/P05065. arXiv: 2305.10515. URL: https://cds.cern.ch/record/2859353.

[3] Large Hadron Collider beauty experiment. *Trigger system*. Large Hadron Collider beauty experiment. URL: https://lhcb-outreach.web.cern.ch/data-collection/trigger-system/.

[4] LHCb collaboration. "The LHCb Upgrade I". In: (2023). URL: https://arxiv.org/pdf/2305.10515.

[5] Brij Kishor. "Standalone track reconstruction and matching algorithms for GPU-based High level trigger at LHCb". In: (Sept. 2, 2022), p. 8. URL: https://cds.cern.ch/record/2826068/files/LHCb-PROC-2022-013.pdf.

[6] LHCb Starterkit. *LHCb data flow in Run 3*. The LHCb Starterkit lessons. 2020. URL: https://lhcb.github.io/starterkit-lessons/first-analysis-steps/dataflow-run3.html.

[7] LHCb Collaboration. *Build Allen*. Allen. URL: https://allen-doc.docs.cern.ch/setup/build.html.

[8] LHCb Collaboration. *Adding a new device algorithm*. Allen. URL: https://allen-doc.docs.cern.ch/index.html.

[9] Salvatore Aiola et al. "HybridSeeding: A standalone track reconstruction algorithm for scintillating fibre tracker at LHCb". In: *Computer Physics Communications* 260 (Mar. 2021), p. 107713. ISSN: 00104655. DOI: 10.1016/j.cpc.2020.107713. arXiv: 2007.02591 [hep-ex, physics:physics]. URL: http://arxiv.org/abs/2007.02591 (visited on 08/15/2024).

[10] LHCb Collaboration. *KinUtils.cuh*. URL: https://gitlab.cern.ch/lhcb/Allen/-/blob/master/device/event_model/common/include/KinUtils.cuh?ref_type=heads.