EPJ Web of Conferences **214**, 01042 (2019) https://doi.org/10.1051/epjconf/201921401042
CHEP 2018

New approaches for track reconstruction in LHCb's Vertex Locator

Johannes Albrecht[1], *Dylan* Bourgeois[2], *Victor* Coco[3], *Ben* Couturier[3], *Christoph* Hasse[1,3,*] *Niklas* Nolte[1,3], and *Sebastien* Ponce[3] on behalf of the LHCb collaboration

[1]Technical University Dortmund, Germany
[2]École polytechnique fédérale de Lausanne, Switzerland
[3]CERN, Switzerland

Abstract. Starting with Upgrade 1 in 2021, LHCb will move to a purely software-based trigger system. Therefore, the new trigger strategy is to process events at the full rate of 30MHz. Given that the increase of CPU performance has slowed down in recent years, the predicted performance of the software trigger currently falls short of the necessary 30MHz throughput. To cope with this shortfall, the Upgrade High Level Trigger currently uses impact parameter cuts on VELO tracks to reduce the amount of tracks which require further processing. The presented work introduces an alternative approach to determine the impact parameter while achieving a more reliable uncertainty prediction than the current method, which could potentially improve the physics quality of these cuts at a later time.

1 Introduction

The LHCb detector is a single arm forward spectrometer with the goal of studying b and c hadrons. A significant upgrade of the experiment is planned for the Long Shutdown 2 (2018-2020) [1]. When resuming operations in 2021 the instantaneous luminosity will be five times higher than before, which introduces new challenges for LHCb's event reconstruction. Furthermore, LHCb will use a fully software based trigger system which will have to process data at the full collision rate of 30 MHz. The proposed High Level Trigger (HLT) system is split into two stages. The first stage (HLT1) will provide a partial event reconstruction which will be used to perform first selections. This will reduce the rate of events needing to pass through the full event reconstruction in the second stage (HLT2). Figure 1 shows the different track types that are reconstructed in HLT1 and HLT2. The tracks in the Vertex Locator (VELO) are the first ones to be reconstructed in HLT1 and are used to determine the location of the primary vertices. Afterwards, these tracks are extrapolated into the Upstream Tracker detector (UT) to build UT tracks. In a last reconstruction step of HLT1 the UT tracks are further extended into the tracking stations (T1-T3) at which point it is possible to reconstruct long tracks, which are used for HLT1 selections.

The latest scenarios aiming to achieve the necessary HLT1 processing rate for such a reconstruction chain are presented in [2]. The most promising ones currently all require a

*e-mail: christoph.hasse@cern.ch

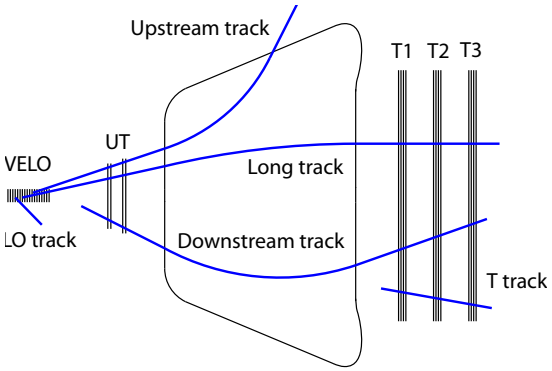

Figure 1. Visualization of the common track types in the LHCb's track reconstruction. A subset of VELO, Upstream, and Long tracks are reconstructed in HLT1, while Downstream and T-tracks are only available in HLT2.

selection to reduce the rate of VELO tracks passed to the following parts of the reconstruction. First, an overview of the VELO reconstruction and the shortcomings of the current selection is given, followed by the introduction of a new prototype method which aims to improve the selection quality by using a machine learning based approach.

2 Reconstruction of VELO Tracks

A simplified sketch of the VELO including a typical signal decay is shown in Figure 2. Since the VELO is located outside of the magnetic field, the tracks inside it are almost perfectly straight lines. This makes the reconstruction procedure rather simple, as a standard track forwarding algorithm is sufficient to find the correct set of hits produced by a single particle. To determine the track parameters the current reconstruction algorithm first performs a linear

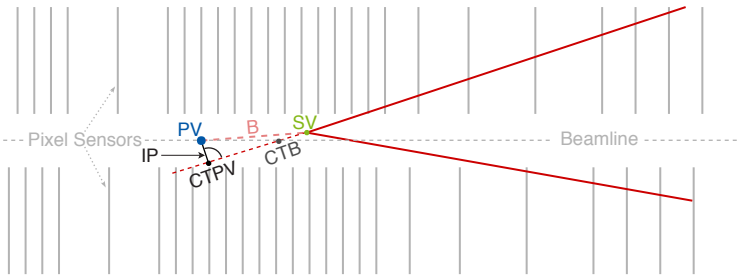

Figure 2. Sketch of B meson coming from the primary vertex (PV) and decaying inside the LHCb Vertex Locator into two daughter particles at the secondary vertex (SV). Additionally, the closest to beam position (CTB), closest to PV position (CTPV), and the impact parameter (IP) are shown.

regression and uses its result to seed a Kalman Filter which yields the final track parameters. The second fit via a Kalman Filter is needed to account for each hit's slight deviation from the straight line due to multiple scattering in material. Unfortunately, due to the lack of a magnetic field, there is no momentum estimate for the track at this stage, forcing the Kalman Filter to rely on a fixed momentum estimate to quantify the scattering magnitude.

After the complete set of VELO tracks is reconstructed they are used to determine the location of the p - p collisions, also called primary vertices (PV). Based on these PVs it is possible to perform the aforementioned selection of VELO tracks which aims so select tracks from secondary vertices by requiring a minimum Impact Parameter (IP). Figure 2 visualizes the IP as the distance between the PV and the point on a track at which it is closest to the PV (CTPV). Given that this point is unknown until after the PV reconstruction, the Kalman Filter in the VELO reconstruction extrapolates the track to a closest to beam position (CTB) from which the CTPV point can later be determined by means of a straight line extrapolation. This is sufficient as the extrapolation can ignore multiple scattering in this case, given that both points are usually within the vacuum and not far apart.

This IP based selection is similar to what was done in LHCb's trigger during Run I (2009-2013). However, it is usually beneficial to cut on a quantity known as χ^2_{IP} which includes a reconstruction quality of the CTPV and PV position. This is so far not possible at the VELO track stage as it requires a correct uncertainty estimate of the CTPV position, which is difficult without a momentum estimate. It is for this reason, that in Run II (2013-2018) these selections are delayed until after the reconstruction of long tracks, which provide a good momentum estimate.

3 VELO Track Fit

The goal of the proposed new method is to achieve a precise estimate on the CTB state (Equations (1) to (5)) and its uncertainty by including a momentum estimate in the procedure. The general idea is explained in more detailed in the following subsection while Section 3.1 focuses on the neural network architecture and its design choices. Finaly, Section 3.2 will detail the dataset and training procedure.

As mentioned above, the VELO reconstruction performs a straight line least squares fit of each track, yielding an approximation of the track's parameters which can be used to determine a first approximation of the CTB state:

$$x = x_0 + m_x \cdot z \tag{1}$$

$$y = y_0 + m_y \cdot z \tag{2}$$

$$z = -\frac{x_0 \cdot m_x + y_0 \cdot m_y}{m_x^2 + m_y^2} \tag{3}$$

$$\frac{\delta x}{\delta z} = m_x \tag{4}$$

$$\frac{\delta y}{\delta z} = m_y \tag{5}$$

with the track's x- and y- slope m_x, m_y and intercept x_0, y_0. This approximation is usually relatively close to what the current Kalman Filter based method achieves as depicted in Figure 3. Additionally, Figure 3 shows that the difference in resolution is more significant in the case of the position while the comparison of the slope's resolution only shows minor differences. Taking into account that the slopes are only used for the extrapolation from the CTB to the CTPV state, which are usually small, the shown differences in the slope will not significantly impact the estimation of the IP. Therefore, the goal of the following method will be to improve the position while ignoring the slight discrepancies in the slopes. This choice is made in an attempt to keep the network as simple as possible, while this decision can easily be reverted if a resolution of the position is achieved at which the slopes' resolution becomes the limiting factor.

Focusing again on the left plot in Figure 3, the shortcoming of the linear regression is mainly due to the fact that it has no information of the process of multiple scattering and thus treats the random deviations from a straight trajectory incorrectly.

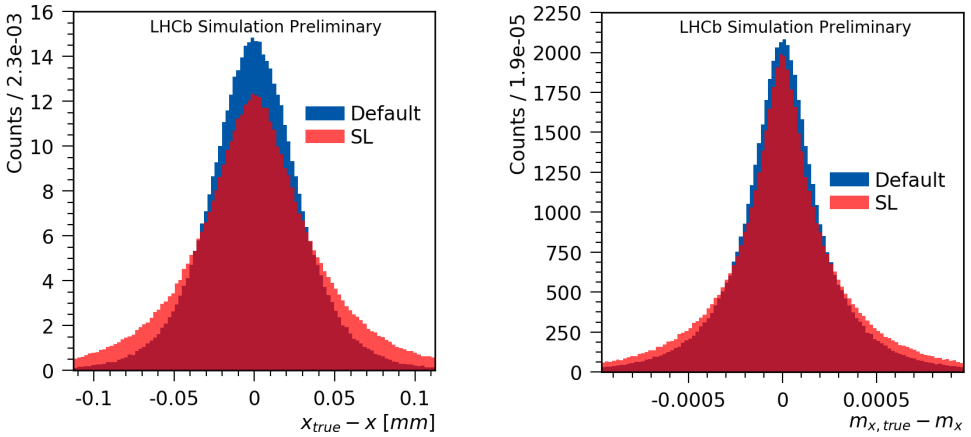

Figure 3. Comparison of the resolution of the x-position (left) and x-slope (right) between the straight line (SL) approximation and the default simplified Kalman Filter.

The goal is to make up for this by defining a machine learning model which is able to use the approximated CTB state and improve upon it based on the hit-residuals $\vec{r}_i$

$$\vec{r}_i = \begin{pmatrix} x_{hit} - (x_0 + (z_{hit} - z) \cdot m_x) \\ y_{hit} - (y_0 + (z_{hit} - z) \cdot m_y) \\ z_{hit} \end{pmatrix}. \tag{6}$$

Assuming these residuals are mainly due to multiple scattering, they will be directly correlated with the scattering angle θ_{plane} shown in Figure 4. The distribution of the multiple

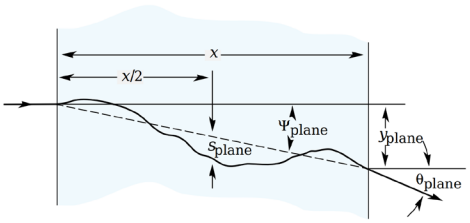

Figure 4. Multiple scattering of a particle traversing material including visualization of the scattering angle θ_{plane}.

scattering angle can be approximated by a Gaussian with an RMS width [3]:

$$\theta^{\text{rms}}_{plane} = \frac{13.6\,\text{MeV}}{\beta cp} z \sqrt{\frac{x}{X_0}} \left[1 + 0.038 \ln\left(\frac{xz^2}{X_0 \beta^2} \right) \right]. \tag{7}$$

The momentum, velocity, incident particle charge, and radiation length are represented by p, βc, z, and x/X_0. As shown in Equation (7) the multiple scattering angle is directly

correlated with a particles momentum. A Neural Network should therefore be able to deduce an approximated momentum from the provided residuals and use it to describe the multiple scattering magnitude. Given that the above assumptions hold, the Neural Network should be able to furthermore use the gained knowledge to improve the estimation of the CTB state's position and predict its uncertainty.

3.1 Model Design

The method explained above defines the inputs of the model to be the approximated CTB state as well as the hit-residuals $\vec{r}_i$. The output has to be the refined position (x, y, z) as well as an uncertainty estimate for these values, which determines the model's output dimension to be fixed in size. Unfortunately, the input dimension is more complicated as it can vary for every track since the number of hits on a track varies from as low as three up to more than 15. In addition to these two constraints, any proposed model design should ensure that its architecture or size will not limit its later use in a trigger environment due to too high inference time.

Given these constraints, the most promising model is build from two commonly used Neural Network architectures. The first being a Long-Short-Term Memory (LSTM) [4] as well as a standard dense Neural Network layer. LSTMs have shown great success capturing correlations in sequential data and considering that a track is essentially a sequential dataset of hits, the LSTM is a excellent choice for our studies, especially, due to its capability to process input data of arbitrary length. A sketch of the model's architecture is shown in Figure 5. In

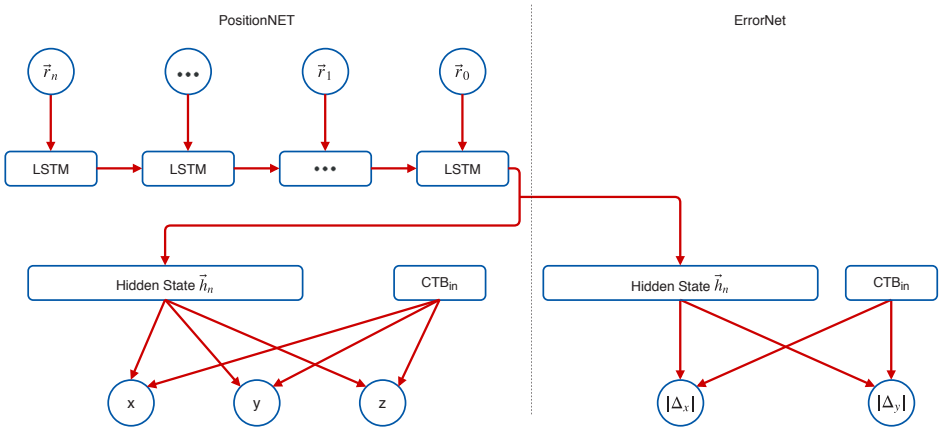

Figure 5. Architectures of the Neural Networks designed to predict the x,y, and z position and its uncertainty of the CTB state.

the first stage of the model the hit-residuals $\vec{r}_i$ are processed by an LSTM in reversed order. This is done to ensure that the hits which are more sensitive to the CTB state's position have a stronger correlation to the LSTM's output. The second stage is based on two independent single layer dense networks, which each take as input a concatenation of the LSTM's output and the approximated CTB state $(x, y, z, \frac{\delta x}{\delta z}, \frac{\delta y}{\delta z})$. One of them will be trained to estimate the position of the CTB state while the other one will predict the uncertainty of that prediction.

3.2 Network Training Procedure

The data used during training and validation is extracted from reconstructed LHCb Monte Carlo events. To avoid biasing the network towards specific signals, it is important to define a training dataset which sufficiently covers the phase space of possible decay channels. Simulated minimum bias data samples which aim to mimic the data produced in p-p collisions before any selections would be good candidates. However, these samples normally contain only a few signal events. For this reason, a sample of simulated signal events which contains a wide range of possible b-decays is chosen. These should sufficiently sample the phase space of possible VELO tracks while also yielding a large enough dataset to achieve a good generalization during training.

The training of the described model is performed in two distinct stages. At first the training focuses on the part of the model predicting the position (left half of the sketch in Figure 5), further referred to as the PositionNet. The true CTB state's position, the networks learning target, is determined based on the Monte Carlo truth information and used to evaluate the performance of a prediction. This is done by translating the predicted value to the correct z-position and evaluating the differences in x and y:

$$CTB_p = (x_p, y_p, z_p) \tag{8}$$

$$CTB_t = (x_t, y_t, z_t) \tag{9}$$

$$\Delta_x = x_t - \Big[x_p + (z_t - z_p) \cdot m_x\Big] \tag{10}$$

$$\Delta_y = y_t - \Big[y_p + (z_t - z_p) \cdot m_y\Big] \tag{11}$$

The subscript p and t indicate predicted and Monte Carlo truth information respectively. The slopes m_x and m_y are taken from the approximated CTB state. For the training of the network we use the following linear loss function,

$$\mathbb{L} = 100 \cdot (|\Delta_x| + |\Delta_y|) + |z_t - z_p|, \tag{12}$$

using Equations (10) and (11) and the difference of the predicted and true $z-$position. This loss allows the network to focus on predicting a CTB position which is on the right trajectory but not necessarily at the perfect z-position. It was observed that this loss leads to slower convergence than the often used Mean-Squared-Error (MSE) loss function, but it leads to better results. Additionally, it was observed that the training was significantly improved by using a cyclic learning rate, similar to that which has been introduced in [5].

For the second stage of the training, the weights of the PositionNet are fixed. Furthermore, the training and validation dataset are fed into the network to calculate an observed absolute error $|\Delta_{x,y}|$ which will be used as the training target for the dense network depicted on the right half of Figure 5 (ErrorNet). A standard MSE loss is used to train the ErrorNet, which will result in the network being optimized to, on average, correctly estimate the absolute error. Assuming the prediction will approximately represent the mean absolute deviation, it is possible to relate this to the standard deviation

$$\sigma_{x,y} = \sqrt{\pi/2}\langle|\Delta_{x,y}|\rangle. \tag{13}$$

4 Results

The plots in Figure 6 show the resolution of the predicted x- and y-position for the current default method in comparison to the new LSTM based method. As shown, the new method

is able to achieve a resolution that is slightly better than the current default, which ensures that any further reconstruction or selection based on the CTB position will be as precise as when using the Kalman Filter based estimate. The data used for this comparison is based on

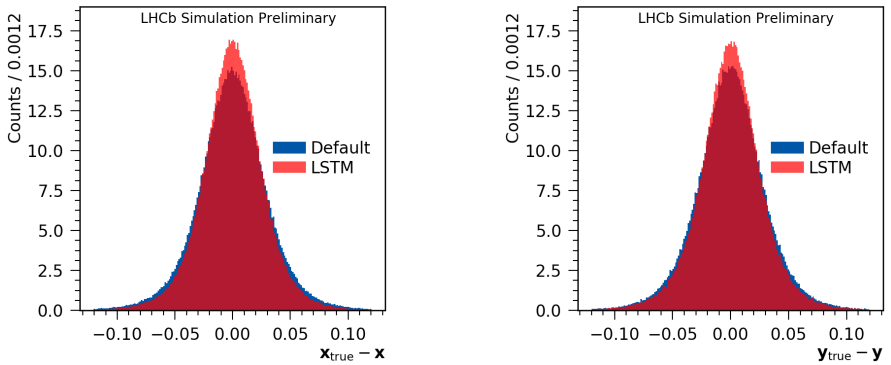

Figure 6. Comparison of the x- and y-resolution between the ML based method and the default simplified Kalman Filter.

simulated $B_d \rightarrow K\pi$ events, to crosscheck the prior assumption that a training using inclusive b-decay events will not introduce any bias. Given that the results are, modulo statistical uncertainty, identical to the results observed on the validation set during training, this assumption seems to hold. The predicted uncertainty on the x- and y-position of the model is compared to the current method in Figure 7, by comparing the distribution of the deviation from the true value $(\Delta x, \Delta y)$ divided by its predicted uncertainty (σ_x, σ_y). In the scenario of either method being able to correctly estimate the uncertainty of its prediction, one should observe a Gaussian distribution with $\mu = 0$ and $\sigma = 1$. Both methods yield a distribution with a mean of zero, indicating no significant bias in our prediction. The deviations from a standard deviation of one show that both methods overestimate the uncertainty of its prediction. Nevertheless, the new machine learning based method does yield a standard deviation much closer to one compared to the default method.

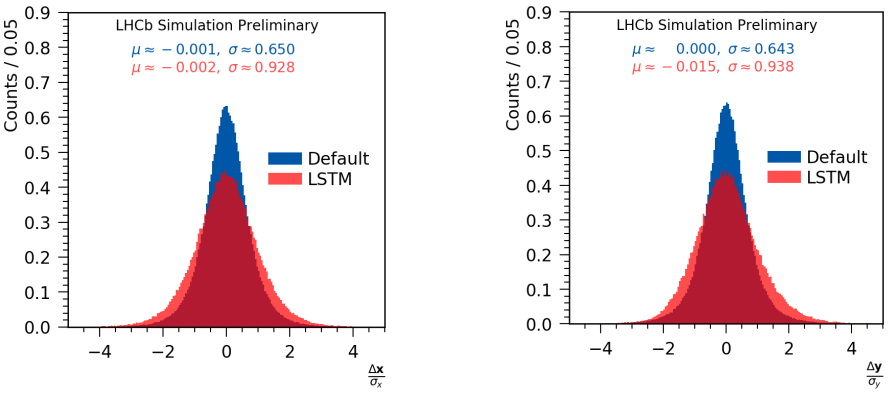

Figure 7. Comparison of the x- and y-pull distribution between the ML based method and the default simplified Kalman Filter.

5 Conclusion and Outlook

A new machine learning based method to improve the prediction of the closest to beam state and its uncertainty has been introduced. It was further shown, that the method is able to successfully predict the x- and y-position to an accuracy at which it is competitive with the currently employed Kalman Filter based method. The uncertainty estimate of these predictions was proved to be more realistic in the case of the machine learning based method. However, for a further adoption of this method it is necessary to validate the above results on other decay signals to better ensure no hidden biases are introduced. Three-body, four-body, and charm decays will probably be good candidates for further validation. Furthermore, the presented method will need to be benchmarked to ensure its inference time is within the time budget of the trigger. In summary, the above prototype is producing promising results which indicate that a method of this kind could be used to improve the quality of an IP or χ^2_{IP} based selection of VELO tracks.

6 Acknowledgements

The authors want to thank the open source community for providing and maintaining many helpful machine learning tools. The presented work was developed using the PyTorch [6] framework.
This work has been sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research (grant no. 05E15CHA)
J. Albrecht gratefully acknowledges support of the Deutsche Forschungsgemeinschaft (DFG, Emmy Noether programme: AL 1639/1-1) and of the European Research Council (ERC Starting Grant: PRECISION 714536).

References

[1] LHCb Collaboration, Tech. Rep. CERN-LHCC-2012-007 (2012)
[2] LHCb Collaboration, Tech. Rep. CERN-LHCC-2018-007 (2018)
[3] M. Tanabashi et al. (Particle Data Group), Phys. Rev. **D98**, 030001 (2018)
[4] S. Hochreiter, J. Schmidhuber, Neural Comput. **9**, 1735 (1997)
[5] L.N. Smith, CoRR **abs/1506.01186** (2015), 1506.01186
[6] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, *Automatic differentiation in PyTorch*, in *NIPS-W* (2017)