YEAR 3 PARTICLE PHYSICS COMPUTING LABS

UNIVERSITY OF BIRMINGHAM

SCHOOL OF PHYSICS & ASTRONOMY

Modelling the Vertex Locator at LHCb

Author:
Solomon Sanderson

Course Lead:
Alan Watson

ID No:
2123945

Date:
December 23, 2022

Abstract

An investigation into the properties of the upgraded Vertex Locator (VELO) detector at CERN's LHCb experiment using a Python model. Aiming to create an accurate model of the detector accounting for properties such as resolution and hit efficiency, use this model to calculate track reconstruction efficiency and use straight line fits to determine the transverse momentum resolution of our detector. The VELO at LHCb determines the collision point of particles; LHCb aims to use this information to uncover the reasons for CP violation to understand the differences between matter and anti-matter. We are undertaking this study as the current VELO is relatively new, first used in 2022. and we want to increase our understanding of the detector, including its quirks and future improvements. We obtained a plot of reconstruction efficiency as a function of pseudorapidity and a transverse momentum value of $p_t \approx 1.75$ GeV, with a resolution between 10^{-6} GeV and 10^{-7} GeV.

Keywords: Vertex Locator, LHCb, CERN, Python Model, Particle Physics

1 Introduction

LHCb aims to investigate the differences between matter and antimatter by studying the beauty quark (b-quark). LHCb takes a different approach to the ATLAS and ALICE detectors; instead of surrounding the collision point with layered detectors, they use a series of sub-detectors [1].

For this project, we study one of LHCb's sub-detectors called the vertex locator (VELO). This detector is close to the interaction region [2]. Using these measurements, scientists can reconstruct the path of the particles to find the location at which the particles collided - the vertex.

We are modelling the upgraded VELO detector used in run three of the LHC, which started in June 2022 [3]. As the luminosity and beam energy of the LHC has increased following the upgrades in long shutdown 3, the sensors will face increased rates and occupation and deliver a readout of 40 MHz [4]. To achieve this, the sensors used previously had to be replaced with pixel detectors [5].

This project aims to:

- Create a model of the vertex locator at LHCb, taking care to account for the per-hit efficiency and sensor resolution.
- Use this model to calculate the track reconstruction efficiency for charged particles produced at the origin.
- Perform a straight-line fit of the particle hits to determine the resolution of our sensors' transverse momentum.

This project is interesting as creating a model of the VELO detector allows us to test the sensor giving us an understanding of its limitations, quirks and how we can improve it in the future.

Additionally, there appears to be limited information on modelling for the new VELO detector online, this means that our model could be useful to other scientists wishing to learn more about the detector.

For our investigation, we used objected oriented Python programming to model the individual sensors before combining them into a sensor array - our detector. Next, we created generated particles with random and uniform distributions of pseudorapidity and azimuth angle to investigate their interactions with the sensor. Using a fit, we can deduce the particle's transverse momentum, momentum resolution, impact parameter and impact parameter resolution.

2 Background & Theory

2.1 LHCb

Through studying the differences between matter and anti-matter, LHCb aims to uncover the reason for charge conjugation parity (CP) violation. CP Violation showed that there is a difference in the decay of matter and anti-matter, thought to be behind the universe's matter-antimatter asymmetry [6].

The LHCb experiment consists of multiple sub-detectors in series along the beam pipe, the detector's layout is seen in Figure 1. This geometry is chosen as particles with b-quarks originating from the beam collision are emitted at small angles to the beam line [6].

The vertex locator is the sub-first detector in the series and it measures the position at which the particles collide.

We can say that a particle has been reconstructed if it hits 3 or more of the VELO sensors [5]. Using the hit positions the trajectory of the particle can be estimated through a simple linear fit - the path of the particles is approximately straight (there is some scattering) since there is no magnetic field. The point at which the paths meet gives the location of the vertex.

The detector consists of a total of 52 sensors located within the LHC beam pipe, 26 of them are on the left, and 26 are on the right. The detector is split into 2 halves so the sensors can be retracted during LHC beam injection. Injection is when the beam from numerous smaller accelerators is transferred to the LHC. During this process the diameter of the beam is increased as it has not been focused, in addition, there are stray particles which lay outside the main beam. When the protons are injected from the SPS into the LHC they have an energy of 450 GeV [7], if a beam of this energy hit a component it would destroy it, further damage can occur if a particle shower was created by this interaction. Particles in the shower would spread over a larger volume than the original beam and potentially destroy a large section of the detector. After injection, the sensors can be moved back into position, as during collisions the beam is focused and stable. When the sensors are fully

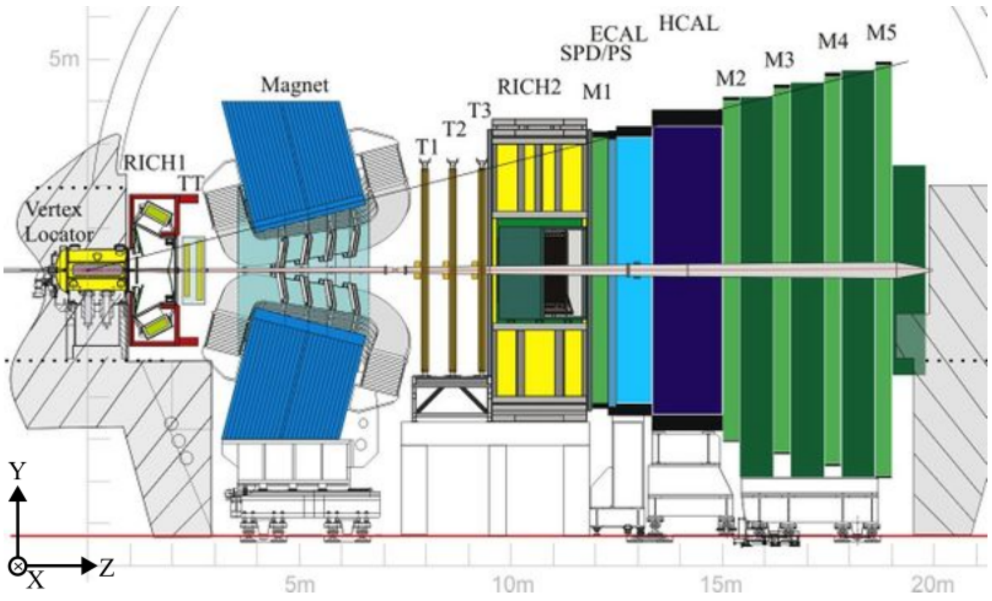


Figure 1: Cross-section view of the LHCb and its main sub-detector elements. The vertex locator is labelled on the left-hand side of the diagram. Collisions typically take place at around (x, y) = (0, 0), Modified from [6].

extended they overlap slightly, to ensure that there is full coverage [4]. The layout of the sensors and their geometry can be seen in Figure 2.

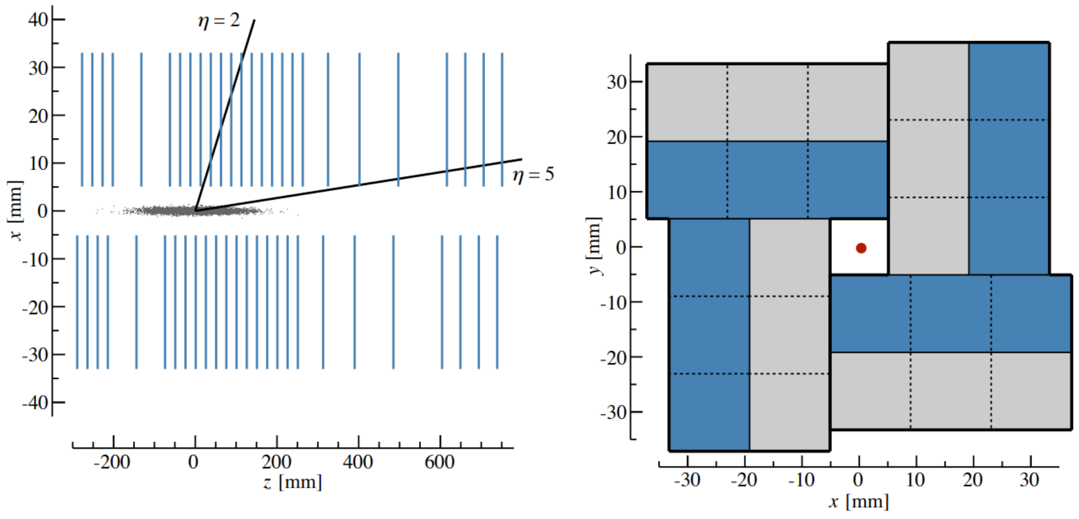


Figure 2: Diagram showing the positions of the left and right sensors along the beam axis and their geometry [8]. The black elliptical region centred at $x = y = 0$ shows the region in which the beams overlap and the particles can collide.

2.2 Coordinate Systems

We use 2 coordinate systems in this project. One of these systems is the Cartesian coordinate system which uses x, y and z, as illustrated in Figure 1.

Alternatively, we can use a spherical polar coordinate system which is defined by pseudorapidity η, azimuth angle ϕ and distance r. Illustrated in Figure 3 below.

We can calculate x and y values given z, η and ϕ with the following:

$$y = z\tan(\theta), \quad x = \frac{z\tan(\theta)}{\tan(\phi)} \text{ where } \theta = 2\tan^{-1}(e^{-\eta}) \tag{1}$$

Where z is an array of sensor locations and θ is the angle to the beamline.

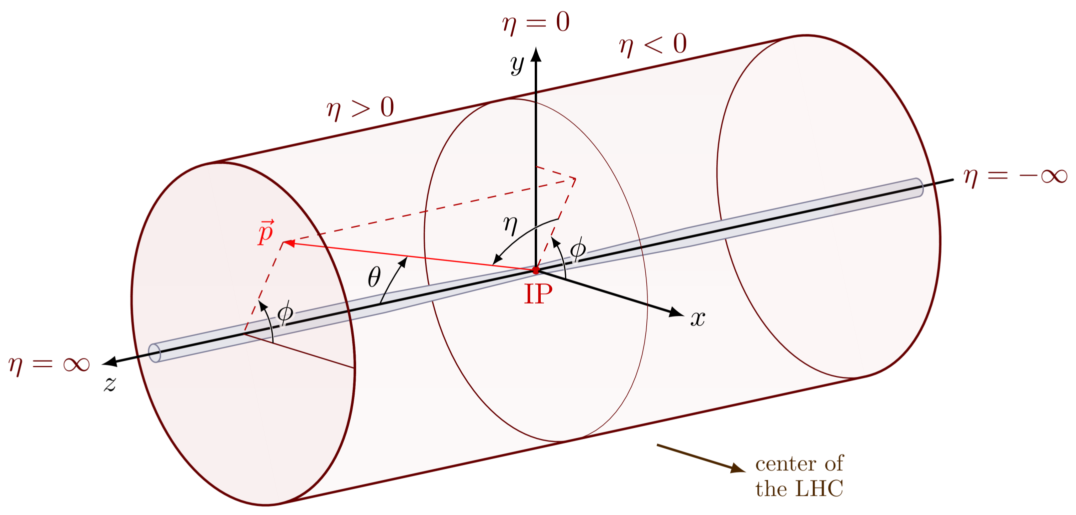


Figure 3: Diagram showing the spherical polar coordinate system with ϕ and η labelled. Modified from [9].

2.3 Sensor Properties

When modelling the sensors we must account for as many of their properties as possible to maximise the accuracy of our model. Two that we aim to include here are hit efficiency and hit resolution, these are detailed below:

Hit efficiency is the ratio of the number of hits a particle gives and the expected number of hits [10] and we give it a default value of 0.98 as specified by the lab manual [5].

Hit resolution is the minimum distance at which the detector can distinguish 2 particles from one another, for our sensors this is 0.012 mm. To account for this we should use a convolution of the Landau and Gaussian functions [11], however, for simplicity we use only a Gaussian. Hence, we smear the path of the particles by adding a random number with a Gaussian distribution to the x and y position of our particles. It would be easy to think that due to the sensor consisting of discrete pixels a uniform random distribution would be appropriate, however, this does not take into account processes such as the sharing of charge between sensors or similar.

The reconstruction efficiency is the percentage of total particles that are reconstructed, this is given by the following equation:

$$\text{Reconstruction Efficiency} = \frac{\text{Reconstructed Particles}}{\text{Total Particles}} \tag{2}$$

3 Method

3.1 Code Structure

To increase the clarity and re-usability of the code I created each class in its own file and called each of them from main.py. The 4 Python files are detailed below. The dependency graph in Figure 4 visualises the code structure:

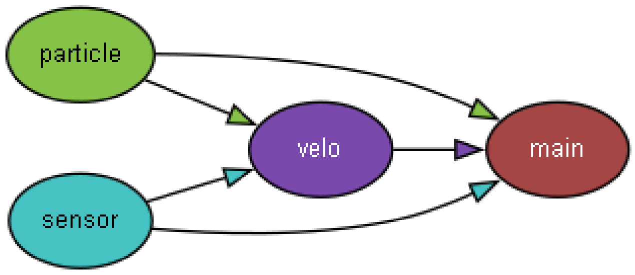


Figure 4: Dependency graph for my code, made using pydeps.

3.1.1 particle.py

In particle.py we define a class to describe the particles in our model. We create a particle with an initial starting position and momentum. We use these values to determine the trajectory of the particle with the xy_pos method. This particle also has a method for setting its momentum magnitude and several others for calculating transverse momentum, impact parameter and their errors.

Defining the particle as a class is particularly helpful as it allows us to easily keep track of the position and momenta of a large number of particles.

3.1.2 sensor.py

Here we primarily define the left and right sensor classes. These represent the left and right-hand sensors that make up the complete VELO detector. We define the left and right sensors separately as they have different geometries and it allows us to position the left and right sensors asymmetrically as seen in Figure 2.

I also define 2 functions for visualising our sensors. plot_sensor plots a left (blue) and right (red) sensor in 2 dimensions. We do the same but in three dimensions with plot_sensor_3d, this is done using 4 cuboids created with cube_plot.py. Each left and right sensor consists of 2 cuboids. We can then plot any number of particles on this axis, although we must note that plotting is very resource intensive and extremely large numbers (larger than 50) are a struggle.

3.1.3 velo.py

In this file we define the velo class, which is a model of the entire VELO detector. When it is initialised, left and right sensor objects are created at the given z values. It has a hits method which takes a particle object, a hit efficiency and a hit resolution.

In this file, I have also defined a function which calculates the reconstruction efficiency. We also define the function fit_3d. This function makes two straight line fits, one for the x, z plane and another for the y, z plane.

3.1.4 main.py

This is where the code should be run from. This is the core script of the program, it is from here that we use particle.py, sensor.py and velo.py to calculate reconstruction efficiency, transverse momentum, impact parameter and the respective resolutions.

3.2 Implementation

Using our code we calculate the reconstruction efficiency, transverse momentum and its resolution, the impact parameter and its resolution.

First, we create a particle object, before calling the xy_pos method, passing it an array of all of the z values. In this method, we calculate an array of xy positions at given z values from the momentum using the fact that ratios of momenta and position are equal. We can use the following to calculate the x, y position at a given z.

$$x = \frac{p_x}{p_z}z,\ y = \frac{p_y}{p_z}z \tag{3}$$

It is convenient to calculate position at the given z values rather than at time intervals. This method avoids the involvement of time and therefore relativistic calculations.

The hits method of velo determines if a particle lies within the sensor. This is done by checking if a particle lies in one of 4 rectangular regions - these are seen in Figure 2, we treat each pair of 3 grey and 3 blue sensors as one rectangle. This method also smears the hits as described in Section 2.3 and we account for the hit efficiency using a random choice, by default 98% of hits are accepted.

To increase the accuracy of the task we extended the model to simulate particles generated at a range of values in x, y and z. We did this by adding uniformly distributed random numbers to each coordinate. We generated values uniformly distributed random numbers in the range $-15 < x, y < 15$ μm to account for the 30μm beam width at LHCb [12]. In the z direction we generated values over the range of $-63 < z < 63$ mm [4] to account for the luminous region in which collisions can take place.

3.2.1 Reconstruction Efficiency

First, we will calculate the reconstruction efficiency of our detector. We call our `velo` class and pass it z values for the left and right sensors to create a `velo` object.

We then generate several particles with uniformly sampled random values of η and ϕ. We generate ϕ values in the range $0 \leq \phi \leq 2\pi$ as this means a particle can be produced at any angle in the xy plane.

Pseudorapidity, η is slightly more complicated as we must find the range over which the reconstruction efficiency is non-zero. We first made a plot of reconstruction efficiency as a function of η. We did this for both particles generated at the origin and those with random start positions. From this, we found that the pseudorapidity went to zero outside of approximately $\eta = \pm$ 7.5.

3.2.2 Transverse Momentum & Resolution

Next, we calculate the transverse momentum of our particles, p_T. We once again generate a number of particles with randomly sampled pseudorapidities and azimuth angles of $\eta = \pm 7.5$ and $0 \leq \phi \leq 2\pi$. We are told by the lab manual that the magnitude of the momentum of our particles must be 10 GeV. To do this we make the following calculation, taking a unit vector and multiplying it by the desired magnitude:

$$\hat{p}_{\text{new}} = \frac{\hat{p}}{p} p_{\text{new}} \tag{4}$$

Where $\hat{p}$ and $\hat{p}_{\text{new}}$ are the current and new momentum vectors and p and p_{new} are their respective magnitudes.

Having done this we calculate where each particle hits one of the sensors and if it is reconstructed. Next, we fit the coordinates of the hits in the x, z and y, z planes, this returns the gradient of each slope, the intercepts of the x and y axes and the respective errors on these values. From this, we get 2 plots similar to those seen in Figure 5.

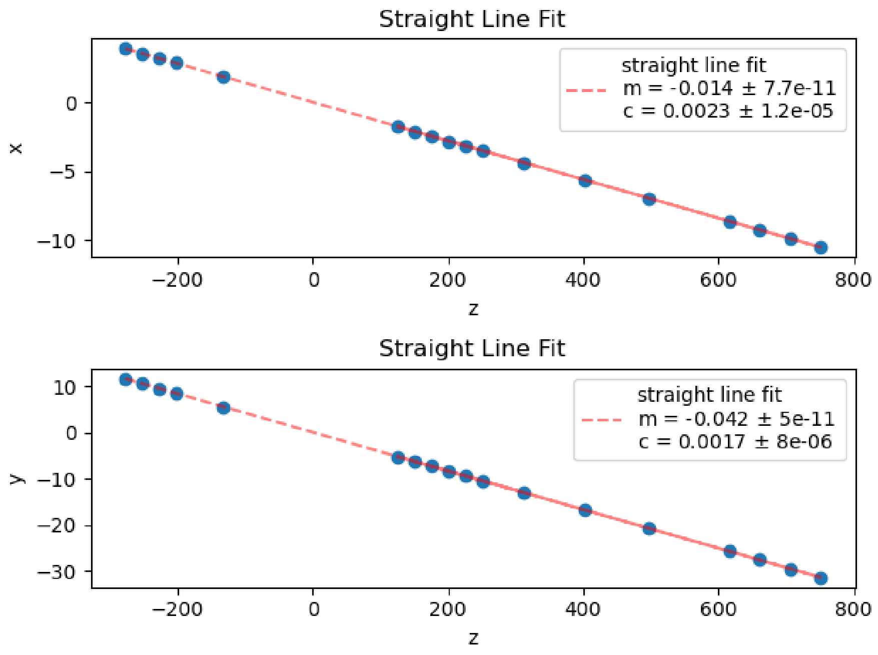


Figure 5: A straight-line fit of the smeared hit values. Note that the lines can have a positive or negative gradient depending on the direction the particle is travelling. A fit in each plane is performed for each particle.

We take the gradient and its error and use this with the following equation:

$$p_T = p\sqrt{\frac{m^2}{m^2+1}} \tag{5}$$

A derivation for this can be found in Appendix A. Using this equation with error propagation we can calculate the transverse momentum resolution of our particles. We derive the formula for transverse momentum resolution in Appendix B:

$$\sigma_{p_t} = \frac{1}{2}\left(\frac{\sigma_{p_x^2+p_y^2}}{p_x^2+p_y^2}\right)p_t \tag{6}$$

For peace of mind we can compare the true values of momentum (the initial randomly generated values) and see how these compare to the values generated using the straight-line fit.

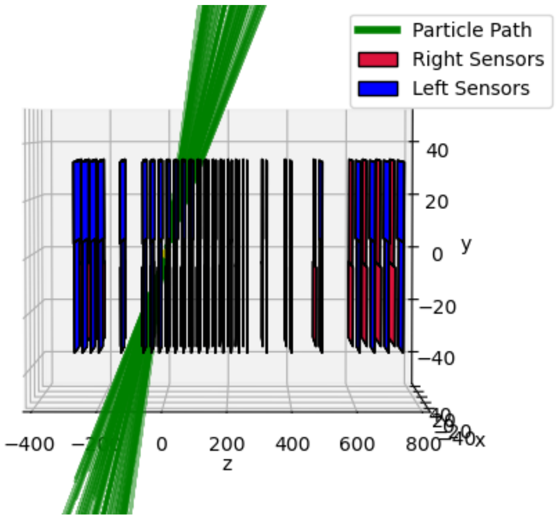
3.2.3 Impact Parameter & Resolution

Finally, we calculate the impact parameter. This is the distance of closest approach of a reconstructed track and the true origin of the particle [8]. To calculate this we find the length of the line between the true origin of the particle and the reconstructed track when the dot product of the lines is zero. The formula and its derivation can be found in Appendix C. Note: when calculating the impact parameter we used particles generated at the origin.

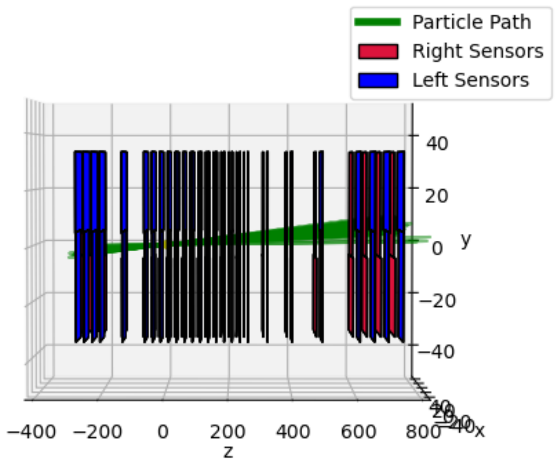
4 Results & Discussion

We must note that the results from each run will vary due to the use of random numbers, however, the values should stabilise as the number of particles is increased. For each run, 10000 particles are used.

Using the `plot_sensor_3d` function we checked if the tracks of the generated particles and returned hits were accurate, we can see in Figures 6a and 6b that at low values of η the particles travel at a much larger angle to the beam axis and vice versa for lower values. This is the behaviour that we would expect from the relation in Equation 1 [13].



(a) $1 \le \eta \le 2, 0 \le \phi \le 2\pi$.



(b) $5 \le \eta \le 6, 0 \le \phi \le 2\pi$.

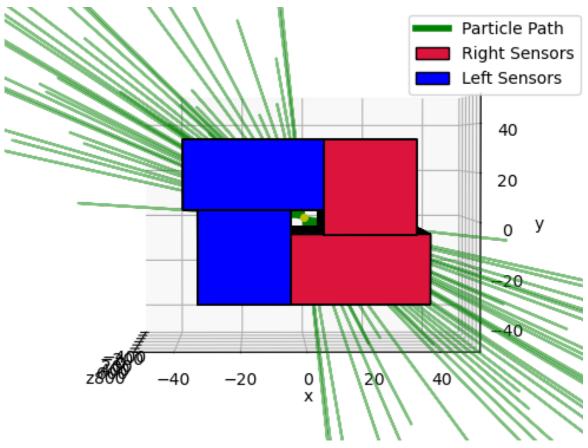
Figure 6: 3 Dimensional plots of the VELO at both low and high pseudorapidities (η). Each green line represents one particle with uniformly sampled, randomly generated values of η and ϕ

We can do the same for the azimuth angle, ϕ. I set the pseudorapidity to be in the range $-6 \le \eta \le 6$ and then set the azimuth angles, we can see that this changes the angle over which particle momenta are generated. These results verify that the way we generate our particles is correct.

From calculating the reconstruction efficiency over a range of pseudorapidities we got Figure 8.

There is little change in the reconstruction efficiency between the values generated at the origin and those generated using randomly sampled values. We would expect this as we have used actual values for the particle collision region, we would expect that the detector would have been designed such that particle tracks originating in this region would be reconstructed.

Our calculation of the transverse momentum for particles with $0 \le \phi \le 2\pi$ and $-7.5 \le \eta \le 7.5$ consistently returns a value of approximately $p_T = 1.75$ GeV. The error on this value was $10^{-7} \le \sigma_{p_T} \le 10^{-6}$ GeV. We can plot a histogram with the momentum of the particles as a function of pseudorapidity as seen below (Figure 9). We do the same for the azimuth angle and see no correlation with pseudorapidity, this is expected as the particle is roughly symmetric in the x, y plane.



(a) $-6 \leq \eta \leq 6, 0 \leq \phi \leq \frac{1}{2}\pi$.

(b) $-6 \leq \eta \leq 6, \frac{3}{2}\pi \leq \phi \leq 2\pi$.

Figure 7: 3 Dimensional plots of the VELO with different ϕ values. Each green line represents one particle with uniformly sampled, randomly generated values of η and ϕ.

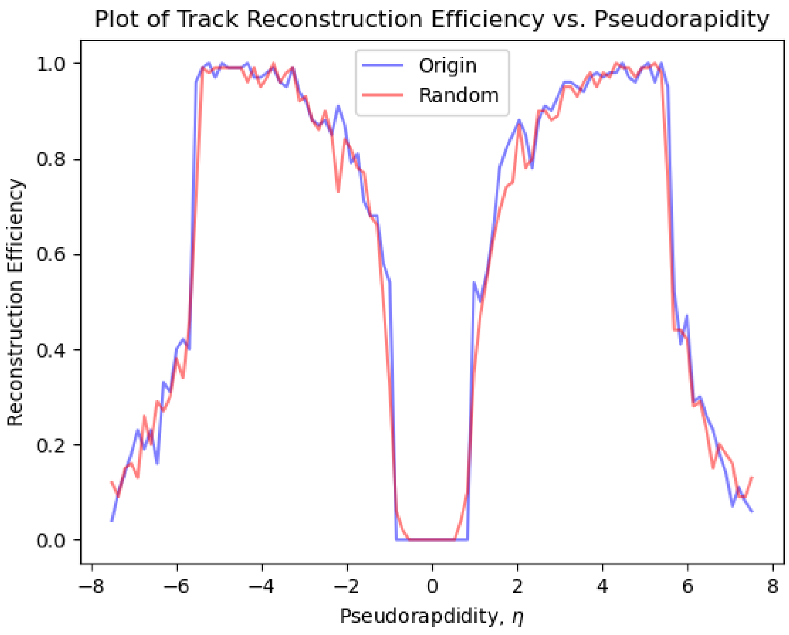


Figure 8: Plot of Reconstruction Efficiency vs. Pseudo Rapidity. The line in blue is the particles which were generated at the origin, those in red were generated using a random uniform distribution in the range $-7.5 \leq \eta \leq 7.5$.

This histogram is generally as expected. In the centre, it goes to zero as the particles travel nearly parallel to the sensors and are not reconstructed. Around this area, we see that the transverse momentum is maximised when the angle to the beam is large. the transverse momentum then decreases as pseudorapidity increases (the angle to the beamline decreases) and the transverse component decreases. Sometimes we see that some bins near the centre will be empty or that the chart is not centred around $\eta = 0$ this appeared to be the result of a division by zero error in our code.

For the impact parameter, we generally get a value of 0.1 mm. Unfortunately, when we tried to determine an error we got extremely large values which make our result insignificant. These are likely due to an error in the code or maths.

5 Conclusion

To conclude, I believe that we have not only met, but exceeded our original project aims. We successfully created a model of the vertex locator including the per-hit efficiency and the sensor resolution - these results were both verified through the testing of the model using extremely low and high values with a 3D plot.

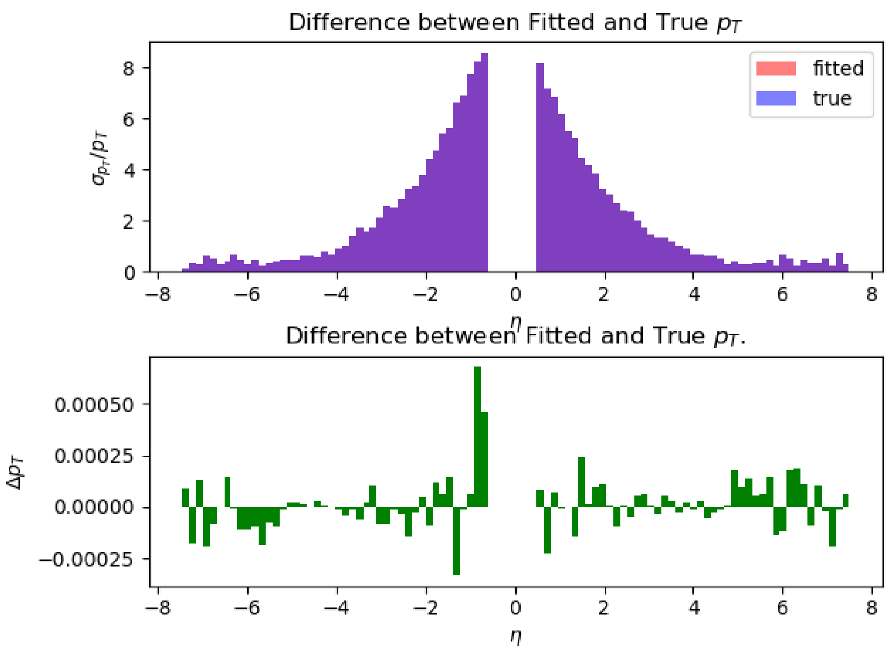


Figure 9: A plot showing the distribution of σ_{p_t}/p_t against η for both fitted and true values

Next, we used this model to calculate the track reconstruction efficiency as a function pseudorapidity and investigated how this was affected by particle starting position, finding that both were extremely similar.

We also successfully calculated a value of the transverse momentum and its error using a straight line fit obtaining the values $p_T \approx 1.75$ GeV and $10^{-7} \leq \sigma_{p_T} \leq 10^{-6}$ GeV. We verified this through both comparisons to the truth value and using our 3-dimensional sensor plot. We also tested this with particles generated at a range of start positions.

In addition to our original aims, we also calculated a value for the impact parameter obtaining a value of $IP \approx 0.1$ mm. To improve the project we could have obtained a correct value for the IP resolution however we did not have time.

References

[1] CERN. Lhcb. 2022. Website, (Link) (Accessed on 02/12/2022).

[2] The LHCb Collaboration. The lhcb detector at the lhc. *Institute of Physics Publishing SISSA*, 2008. Paper, (Link) (Accessed on 01/12/2022).

[3] CERN. Lhc run 3. 2022. Website, (Link) (Accessed on 03/12/2022).

[4] LHCb Collaboration. LHCb VELO Upgrade Technical Design Report. Technical report, 2013. Report, (Link) (Accessed on 03/12/2022).

[5] Alan Watson. Year 3 particle physics computing lhcb vertex locator project description. page 1, 2022. Lab Manual.

[6] F. Soomro and P. Campana. The LHCb experiment. *Scholarpedia*, 11(7):32452, 2016. revision #153343, Paper, (Link) (Accessed on 03/12/2022).

[7] R.Schmidt. Accelerator physics and technology of the lhc. page 20, 1999. Paper, (Link) (Accessed on 10/12/2022).

[8] P. C. Tsopelas. A silicon pixel detector for lhcb. page 23, 2016. Paper, (Link) (Accessed on 13/12/2022).

[9] Izaak Neutelings. Cms coordinate system. 2021. Website, (Link) (Accessed on 01/12/2022).

[10] M. Smith T. Szumlak . S. Borghi, C. Parkes. Study of the velo hit resolution in run 1. page 20, 2016. Paper, (Link) (Accessed on 06/12/2022).

[11] et. al. R. Geertsema. Charge collection properties of prototype sensors for the lhcb velo upgrade. *arXiv*, page 5, 2020. Paper, (Link) (Accessed on 01/12/2022).

[12] R. Matev G. Coombs, M. Ferro-Luzzi. Beam-gas imaging measurements at lhcb. *Int. Beam Instrumentation Conf*, 7th:460, 2018. Paper, (Link) (Accessed on 16/12/2022).

[13] E. Daw. Lecture 7 - rapidity and pseudorapidity. page 8, 1999. Lecture, (Link) (Accessed on 10/12/2022).

6 Appendix

A Derivation of Equation for p_T.

We know m_x and m_y. We can say:

$$m_x = \frac{p_x}{p_z} \text{ and } m_y = \frac{p_y}{p_z} \tag{7}$$

Summing the squares of these fractions will give us the slope squared in 3 dimensions, m^2.

$$m^2 = \left(\frac{p_x}{pz}\right)^2 + \left(\frac{p_y}{p_z}\right)^2 = \frac{p_x^2 + p_y^2}{p_z^2} \tag{8}$$

We know that the transverse momentum p_T is given by:

$$p_T^2 = p_x^2 = p_y^2 \tag{9}$$

Substituting this into Equation 8 we get the following:

$$m^2 = \frac{p_T^2}{p_Z^2} \tag{10}$$

We can also say, where P is the total momentum:

$$p^2 = p_T^2 + p_Z^2 \implies \frac{p_T^2}{p_Z^2} = \frac{p_T^2}{p^2 - p_T^2} = m^2 \tag{11}$$

Rearranging for P_T we get:

$$p_T = p\sqrt{\frac{m^2}{m^2+1}} \tag{12}$$

B Derivation of Equation for Error on p_T, σ_{p_T}.

We know:

$$p_x^2 + p_y^2 = p_T^2 \implies p_T = \sqrt{p_x^2 + p_y^2} \tag{13}$$

We know that to propagate errors with indices we multiply the relative error of the value by the power, we first calculate the error on $p_x^2 + p_y^2$:

$$\sigma_{p_x^2+p_y^2} = 2\left(\frac{\sigma_{p_x}}{p_x} + \frac{\sigma_{p_y}}{p_y}\right)(p_x^2 + p_y^2) \tag{14}$$

We then account for the root:

$$\sigma_{p_t} = \frac{1}{2}\left(\frac{\sigma_{p_x^2+p_y^2}}{p_x^2+p_y^2}\right)p_t \tag{15}$$

Next we need the errors of the momentum in the x, z and x, y planes which are σ_{p_x} and σ_{p_x} respectively.

$$\sigma_{p_x} = p_z\sigma_{m_x} \text{ and } \sigma_{p_z} = p_z\sigma_{m_y} \tag{16}$$

Where σ_{m_x} and σ_{m_y} are the errors on the gradients of the fits in their respective planes. Finally we substitute the values into Equation 15 to get our value.

C Derivation of Equation for Impact Parameter, IP.

We take 2 points from our fitted line and use them to determine the equation of the line $\vec{r}$, we call the true origin $\vec{O}$:

$$\vec{r} = t\begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \text{ and } \vec{O} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \tag{17}$$

We call the distance between the origin and the point on the line $\vec{b}$:

$$\vec{a} = \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix} \text{ and } \vec{b} = t\begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix} + \begin{bmatrix} x_2 - x_0 \\ y_2 - y_0 \\ z_2 - z_0 \end{bmatrix} \tag{18}$$

The distance of closest approach is when the path from the origin to the line and the line itself are perpendicular, i.e. when the dot-product is equal to zero.

$$\vec{a}\cdot\vec{b} = \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix} \cdot \left(t\begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix} + \begin{bmatrix} x_2 - x_0 \\ y_2 - y_0 \\ z_2 - z_0 \end{bmatrix} \right) \tag{19}$$

From this we get a value for t at which $\vec{a}$ and $\vec{b}$ are perpendicular.

$$t = -\frac{(x_2 - x_0)(x_1 - x_2) + (y_2 - y_0)(y_1 - y_2) + (z_2 - z_0)(z_1 - z_2)}{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{20}$$

We can then substitute t into $\vec{b}$ and take the magnitude of b to get the shortest distance IP between the true origin and the line.

$$IP = |\vec{b}(t)| \tag{21}$$

D Derivation of Equation for Error on the Impact Parameter, σ_{IP}.

We have error values for both the gradient of our straight-line fits and the y-intercepts of our straight-line fits.

$$\sigma_{\text{fit},x} = \frac{\sigma_{m,x}}{m_x}z + \sigma_{c,x} \text{ and } \sigma_{\text{fit},y} = \frac{\sigma_{m,y}}{m_y}z + \sigma_{c^y} \tag{22}$$

The error on t is then given by, since we add the relative errors of the numerator and the denominator:

$$\sigma_t = \left[\left(\frac{\sigma_{\text{fit,x}}}{x_2 - x_0}\right) + \left(\frac{2\sigma_{\text{fit,x}}}{x_1 - x_2}\right) + \left(\frac{\sigma_{\text{fit,y}}}{y_2 - y_0}\right) + \left(\frac{2\sigma_{\text{fit,y}}}{y_1 - y_2}\right) + 2\left(\frac{2\sigma_{\text{fit,x}}}{x_1 - x_2}\right) + 2\left(\frac{2\sigma_{\text{fit,y}}}{y_1 - y_2}\right)\right] t \tag{23}$$

Next we calculate the errors on the x and y components of $\vec{b}$, since t and are multiplied we add the relative errors:

$$\sigma_b^x = \frac{\sigma_t}{t} + \left(\frac{2\sigma_{\text{fit},x}}{x_1 - x_2}\right) + \sigma_{\text{fit},x} \text{ and } \sigma_b^y = \frac{\sigma_t}{t} + \left(\frac{2\sigma_{\text{fit},y}}{y_1 - y_2}\right) + \sigma_{\text{fit},y} \tag{24}$$

We must then calculate the error propagation for us calculating the magnitude of .

$$\sigma_{IP} = \frac{1}{2}\left(\frac{\sigma_{x,y,z}}{x^2 + y^2 + z^2}\right) \text{ where } \sigma_{x,y,z} = \frac{2\sigma_b^x}{x} + \frac{2\sigma_b^y}{y} \tag{25}$$

E GitHub Repository.

A GitHub repository of the code for this project can be found at: https://github.com/solomonsanderson/Year-3-Computing-Project.