University of Glasgow | Experimental Particle Physics

GLAS-PPE/2008-02
24th January 2008

Department of Physics and Astronomy
Experimental Particle Physics Group
Kelvin Building, University of Glasgow,
Glasgow, G12 8QQ, Scotland
Telephone: +44 (0)141 330 2000 Fax: +44 (0)141 330 5881

Reconstruction of Cluster Positions in the LHCb VELO

T. Szumlak[1], C. Parkes[1], T. Ruf[2],

[1] University of Glasgow, Glasgow, G12 8QQ, Scotland
2 CERN, Meyrin, CH-1211 Geneve 23, Switzerland

Abstract

This note describes the 'Velo Cluster Position Tool'. This software is used in the GAUDI framework to estimate the hit position of a particle traversing the silicon sensors of the LHCb VELO and to estimate the uncertainty on this position. This estimate and its uncertainty are used in the LHCb track fit. The definition of the cluster centre is given and the baseline linear approximation method presented. The position error is strongly dependent on the angle of incidence of the particle on the silicon sensors measured perpendicularly to the strips – known as the projected angle – and on the silicon sensor pitch at the point of incidence, and is parametrised in terms of these variables. Pull plots are presented to show the quality of the current tuning implemented for simulation events.

Contents

1 Introduction

This note describes the procedure used to make a precision estimate of the VELO cluster position in the LHCb software. This precision estimate is used by the LHCb track fit and hence is relevant to the off-line physics performance of the experiment.

The VELO TELL1 boards perform the raw signal processing and as their final stage reconstruct clusters. The cluster data is stored in the RawBank [4] structure of GAUDI [3]. The TELL1 processing board also produces an estimate of the cluster position with three bit precision relative to the inter-strip pitch [2], this estimate is decoded in the `VeloLiteCluster` object in Gaudi and is used in the pattern recognition, this is not discussed further here. The complete list of strips that have been identified by the TELL1 to come from a cluster are decoded into the `VeloCluster` object in Gaudi. In the simulation, the VELO clusters are produced at the last stage of the LHCb detector response simulation program Boole [1] and stored in the RawBank format. In both the real and simulated data cases the clusters are represented by the same `C++` classes. The `VeloCluster` is the starting point for the VELO Cluster Position tool described in this note.

The baseline simple linear algorithm for calculating the VELO cluster position is given in section 2 of this document and the cluster position defined. Section 3 defines the projected angle of a track and describes the calculations for R and Phi sensors. An exact and an approximate calculation are presented. Section 4 presents the projected angle and pitch parametrisations used to estimate the uncertainty on the cluster position. The global to local frame transformations required are described in the Appendix. The software implementation of the cluster position tool is presented in section 6 and a summary and conclusions are given in section 7.

This note provides a description of the baseline cluster position tool that performs the position calculation using a linear approximation for describing the charge sharing between a sensor's strips. Work is underway in the VELO on the use of non-linear η dependent models on testbeam data, this should allow an improved cluster position estimate.

The error parametrisations described here have been tuned for the simulated LHCb events, any discrepancy between the data and simulation resolutions will require additional parametrisation tunings to be obtained for the data.

2 VELO Cluster Position

A particle that traverses the VELO silicon sensor deposits charge within the sensor that is subsequently collected on its strips. The measured strip signals are digitized and processed – referred to as ADC counts – and subjected to a clusterization procedure inside the TELL1 processing board. The measured signal heights can be used to reconstruct the crossing point of the particle.

The clusters are decoded into a dedicated `C++` class called `VeloCluster`. Each object of type of `VeloCluster` contains:

- VELO sensor identifier
- `VELOLiteCluster`

and for each strip in the cluster:

- channel identifier of each strip in the cluster
- ADC Counts of strip (after all TELL1 processing has been applied)

This is discussed further in reference [7].

The functionality of the TELL1 clustering algorithm is illustrated in Fig. 1, for the examples of two and three strip clusters. The algorithm is described in [8], it requires that the ADC value of at least one strip passes a seeding cut (SC), and adjacent strips are then included if they pass a lower inclusion cut (IC).

In the first case shown (Fig. 1 a) a two strip cluster would be formed and the reconstructed cluster centre position is between strip 0 and 1. In the case of Fig. 1b a three strip cluster would be created, since both adjacent strips would pass the inclusion cut, and the reconstructed cluster position is between strip 1 and strip 2.

The cluster position tool, described in this note, reconstructs the position from the ADC counts of the strips in the cluster. Diagramatically the cluster position, *i.e.* the centre of the reconstructed cluster, is indicated by the green arrow in both cases in Fig. 1. The cluster centre is considered as the best estimate of the location of the traversing particle at the centre plane of the silicon sensor.

The information on the distribution of the charge over more than one strip allows us to produce a more accurate position measurement than would be obtained in a binary system. The cluster position is determined from the pulse height weighted average of the strips contributing to the cluster:

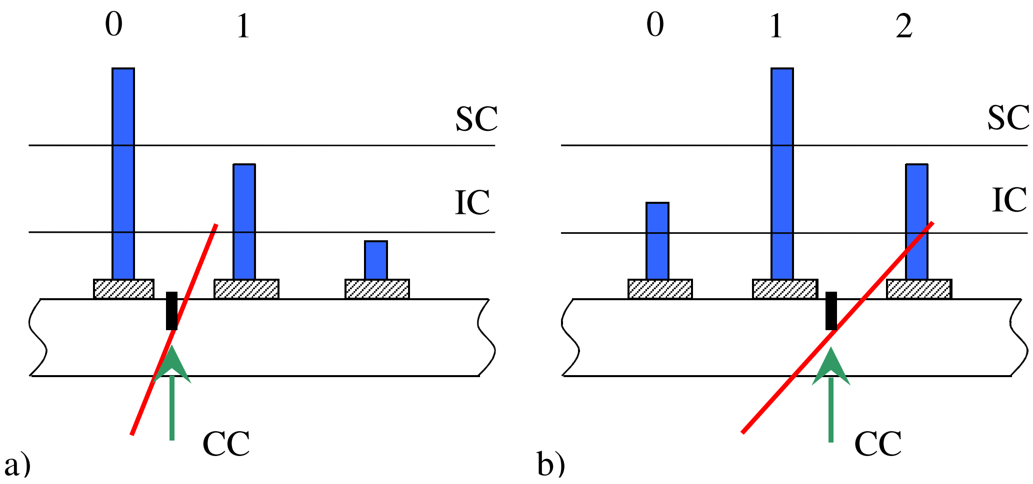


Figure 1: Two examples of VELO clusters are shown schematically. Blue rectangles depict the charges collected on the strips, black lines represent seeding (SC) and inclusion (IC) cuts, the reconstructed cluster centre (CC) is indicated by arrows and the particle trajectory is given by the red sloping lines.

$$\text{Cluster Position} = \sum_i w_i \cdot d_i, \tag{1}$$

where w_i is a ADC count for a given strip i divided by total cluster charge:

$$\mathrm{w_i} = \frac{q_i}{\sum_i q_i}, \tag{2}$$

d_i is the channel number of the strip in the cluster and index i denotes a sum over the cluster's strips.

This simple algorithm assumes that the weighting field inside the silicon sensor gives rise to perfect charge sharing, so that there is a linear relation between the inter-strip impact position of the particle and the signal measured on the adjacent strips. As this is not the case, more sophisticated algorithms can give a better estimate of the cluster position and are under development.

In order to pass information on the cluster centre position to subsequent reconstruction algorithms the identification number of one of the cluster's strips is used along with a fractional inter-strip position.

Each strip has a unique identifier known in the Gaudi framework as a `VeloChannelID`. The object is in principle equivalent to an integer number, that encodes all the information needed to identify uniquely each channel (strip) of the VELO detector. Geometrically adjacent strips have channel identifiers that differ by 1.

Software versions subsequent to $v2r3$ of the VELO Cluster Position Tool, adopt the convention described here for the integer channel ID and fractional position that are used. The scheme is the same as is used in the TELL 1 cluster position calculation and decoded into the `VeloLiteClusters`.

The scheme can be explained using Fig. 1 again. In the case of the two strip cluster the position is denoted by the `VeloChannelID` of the strip number 0 and a fraction of a strip pitch (in the range 0 to 1) corresponding to the inter-strip position indicated by arrow. In the case of the three strip cluster its position would be built as the `VeloChannelID` of the strip number 1 and the appropriate inter-strip fraction.

In addition the cluster position tool returns an error estimate on the position. The majority of this note describes the parametrisation of this error estimate.

3 Projected Angle

The cluster position tool is required to return an estimate of the uncertainty on the reconstructed cluster position for use in the tracking algorithms. This hit resolution of the silicon sensor depends strongly on both the inter-strip pitch and the angle at which a particle crosses the sensor. Hence the resolution has been parametrised in terms of these variables.

The projected angle is the quantity that plays the primary role in the VELO resolution studies rather than, for example, the number of strips in the cluster. The definition of the projected angle and two algorithms for its calculation are presented in this section. The first algorithm provides an exact calculation whereas the second algorithm provides a simpler good approximation. Results from both algorithms are presented.

In order to calculate the projected angle it is necessary to use an estimate of the track state. A space point and the track slopes given at this point are needed. The tracks slopes are needed in the local frame of the sensor, the transformations applied to move from the global LHCb frame to the local sensor frame are described in Appendix A.

Clearly in order to use this method the hits on the track must already have been identified. The pattern recognition algorithm itself does not need this level of precision for the cluster position error estimate. The accurate cluster position uncertainty can then be used for refining the existing track.

3.1 Definition of the Projected Angle

The precision of the measurement of the particle's position in the VELO sensors is, naturally, related to the inter-strip pitch. The smaller the sensor pitch the better position resolution is expected.

There is, however, another important element in the position resolution - the sharing of the deposited charge between silicon strips. The cluster position is estimated using the ADC count weighted strips, as described in section 2. A system in which all charge is measured on a single strip gives the so-called binary resolution of pitch/$\sqrt{12}$, this may be improved by making use of the charge sharing between strips.

The charge distribution strongly depends on the angle at which the particle crosses the strips since as this angle increases there is a larger chance to form a multi-strip cluster. Thus we expect that the position measurement precision will also depend on the angle.

The relevant angle is the crossing angle of the particles projected perpendicular to the strips. The geometrical definition of this projected angle is shown in Fig. 2. This angle is defined in the same way for both the Phi and R sensors. In the case of physics tracks coming from the primary vertex this projected angle is similar to the track polar angle for the R sensors in the VELO but for the Phi sensors the projected angle is usually a small angle.

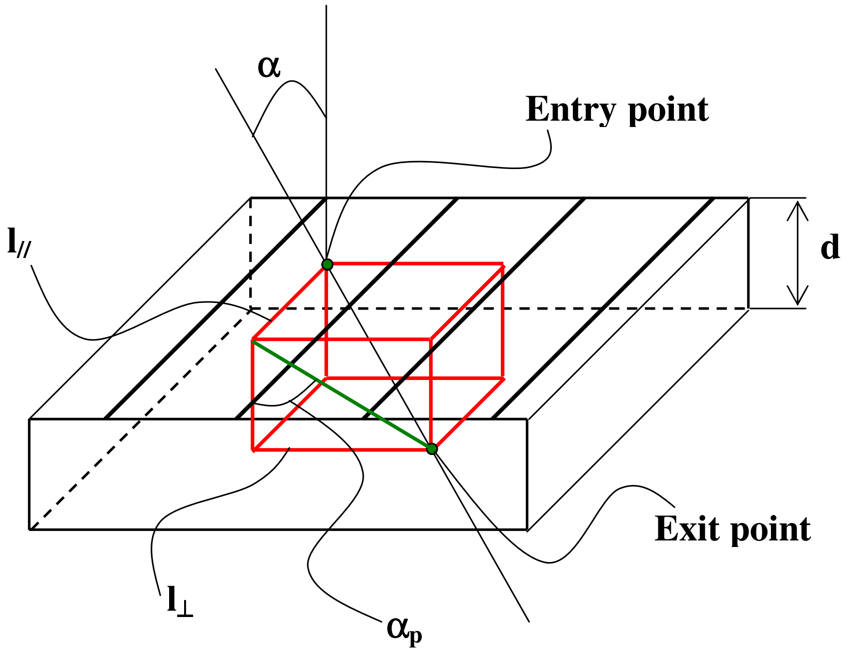


Figure 2: Geometrical definition of the projected angle. α and α_p are the track angle and projection angle respectively. $l_\perp$ and $l_{//}$ are the track's component parallel and perpendicular to a strip. The sensor's thickness is designated by d.

The projected angle can be calculated using the following formula:

$$\tan(\alpha_p) = \frac{d}{l_\perp}, \tag{3}$$

where: d is the sensor thickness and $l_\perp$ is the track component perpendicular to a strip. The component parallel to the strip is not taken into account since that segment of the track does not contribute into the charge sharing.

Two independent methods for the projected angle determination have been devised during the implementation of the tool. The first is based on the definition of the projected angle that was given in the previous section and will be referred to as the exact method. The second one is based on a few simplifications and is less CPU time consuming.

3.1.1 Exact Calculation of the Projected Angle

Due to the different strip geometries the determination of the projected angle is sensor type specific. However, the procedure has a common part that includes the calculation of the vector parallel to the track using the track slopes expressed in the local coordinate system, $\mathbf{v_t}$. this vector is then used for the projected angle calculation for both the Phi and R strip geometry.

- R Sensor Geometry

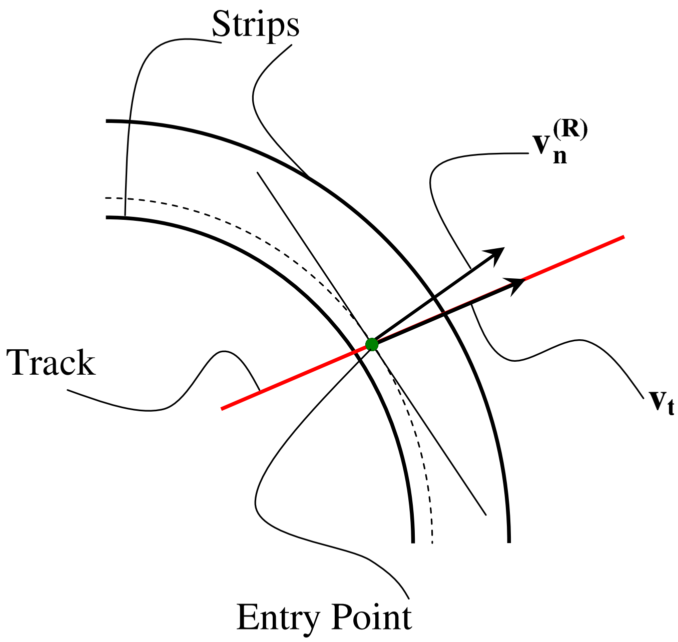


Figure 3: Graphical representation of the projected angle α_p for the R geometry strips.

The R sensor projected angle calculation is illustrated in Fig. 3. The vector parallel to the track $\mathbf{v_{xz}}$ and the entry point $P_{loc}(x_0, y_0, 0)$ are known and expressed in the local reference frame. The vector perpendicular to the circular strip segment at the particle entry point is:

$$\mathbf{v_n^{(R)}} = [\mathbf{x_0}, \mathbf{y_0}, \mathbf{0}]. \tag{4}$$

The projected angle may now be calculated:

$$\alpha_p^{(R)} = arctan(\frac{v_{tn}}{v_{tZ}}), \tag{5}$$

where α_p is the projected angle, v_{tn} is the projection of the vector parallel to the track ($\mathbf{v_t}$) on the vector $\mathbf{v_n^{(R)}}$ and v_{tZ} is the projection of the vector $\mathbf{v_t}$ on the local Z axis Z_{loc} . The value of v_{tn} can be computed using:

$$v_{tn} = |\mathbf{v_t}| cos(\alpha_{tn}), \tag{6}$$

where $cos(\alpha_{tn})$ can be expressed by:

$$cos\angle(\mathbf{v_t}{}^\wedge\mathbf{v_n^{(R)}}) = cos(\alpha_{tn}) = \frac{\mathbf{v_t} \cdot \mathbf{v_n^{(R)}}}{|\mathbf{v_t}|\ |\mathbf{v_n^{(R)}}|}. \tag{7}$$

- Phi sensor geometry

The principles of calculating the projected angle for the *Phi* sensor are exactly the same as for the *R*. The only difference is the determination of the vector perpendicular to the strip since for *Phi* type sensors strips are pseudo-radial line segments. The general idea is depicted in Fig. 4. Again, we assume that $\mathbf{v_t}$ and $P_{loc}(x_0, y_0, 0)$ are known. In order to determine $\mathbf{v_n^{(Phi)}}$ one needs to find the equation of the line that contains the end points of the strip that is closest to the reconstructed cluster's centre. This can be retrieved via the VELO detector element iterface:

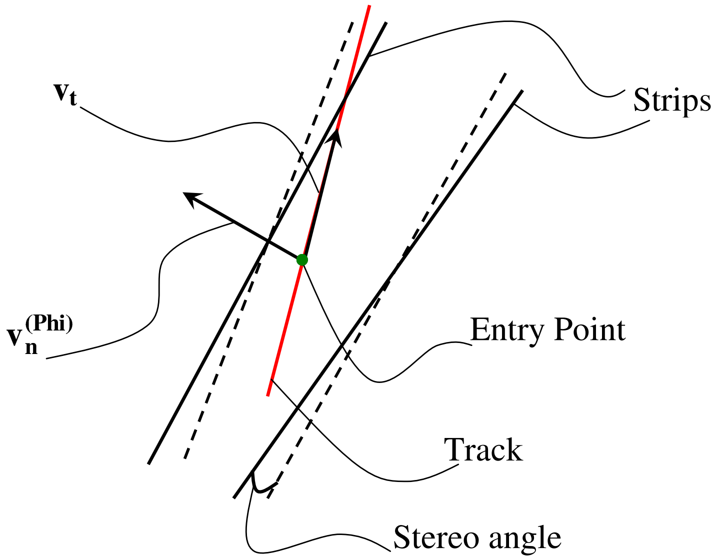


Figure 4: Graphical representation of the projected angle α_p for the Phi geometry strips.

$$pair < Gaudi :: XYZPoint, Gaudi :: XYZPoint > localStripLimits(unsigned\ int\ strip)\ const$$

with the strip number retrieved from a `VELOCluster` object. Knowing both end points of the strip the vector $\mathbf{v_n^{(Phi)}}$ can be calculated. The calculation then proceeds in the same manner as that presented for the R sensor (see formula (6)).

3.1.2 Approximate Projected Angle

- R Sensor Geometry

The approximate projected angle algorithm for the R sensors requires only a cluster and the slopes in the global reference frame. The central channel and the local pitch can be retrieved from the cluster and the projected angle is determined as:

$$\alpha_p \approx \alpha_t \approx \sqrt{s_{xz}^2 + s_{yz}^2}, \tag{8}$$

where s_{xz} and s_{yz} are track slopes in XZ and YZ planes respectively and α_t is a track angle. This calculation is exact in the case that the track origin is on the Z-axis (x,y=(0,0)), there are no sensor misalignments and the track slope is small.

- Phi Sensor Geometry

The approximate calculations for the Phi sensor is somewhat more complicated and requires an `LHCb::Trajectory` object that is built using the centre strip and the reconstructed cluster centre with respect to this strip (*i.e.* the fractional position). The trajectory is retrieved via `VeloDet` detector element interface.

The stereo angle of the trajectory is calculated using the dot product of the unit vector parallel to the trajectory and the unit vector parallel to a line segment between the origin of the local reference frame and the crossing point of track. The projected angle may be approximated subsequently as:

$$\alpha_p \approx \alpha_t * \alpha_s = \sqrt{s_{xz}^2 + s_{yz}^2} * \alpha_s, \tag{9}$$

where α_s is stereo angle of the linear trajectory, s_{xz} , s_{yz} and α_t have the same meaning as in equation 8. Again, this calculation is exact in the case that the track origin is on the Z-axis (x,y=(0,0)), there are no sensor misalignments and the track slope is small.

3.2 Comparison of the Projected Angle Algorithms

In order to evaluate the quality of the approximate calculation additional studies were performed. Correlation plots and a comparison of the projected angle distributions are presented below.

3.2.1 Projected Angle Distributions

The projected angle distributions for R and Phi sensors are expected to be significantly different. For R sensors tracks that come from the vicinity of the interaction point should have projected angles almost identical to their polar angles. In the case of the Phi geometry we expect that projected angle distribution should be dominated by small angles as most of tracks come from interaction region thus travel almost parallel to the Phi sensor's strips in the XY plane. This can be seen on Fig. 5. where the track angle and projected track angle distributions are shown superimposed for both R and Phi sensors.

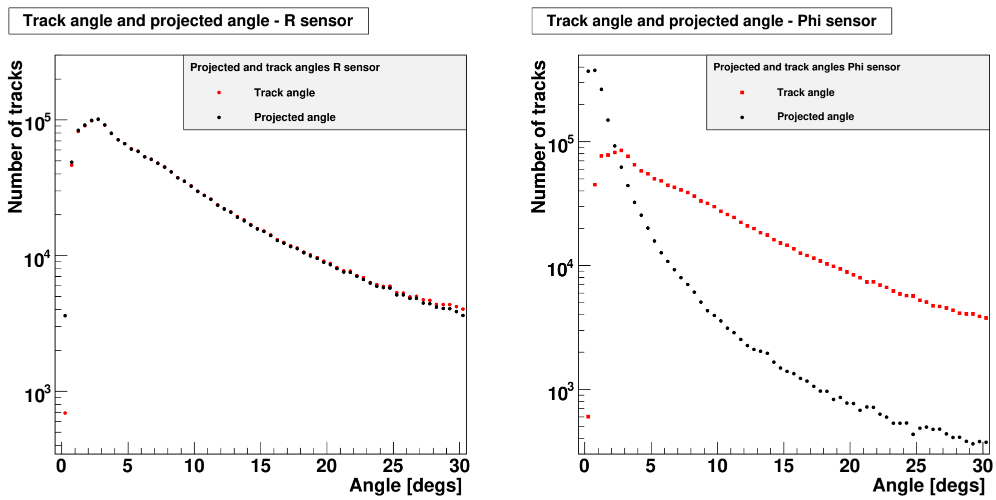


Figure 5: The left hand plot shows the superimposed distributions of the track and projected angle for the R sensor geometry, the same distributions are shown on the right hand plot for the Phi sensor.

In the case of the R type sensor both distributions are very similar (apart from a slight difference for the first bin). For the Phi type the shape of the projected angle distribution is, of course, completely different from the distribution of the track polar angle.

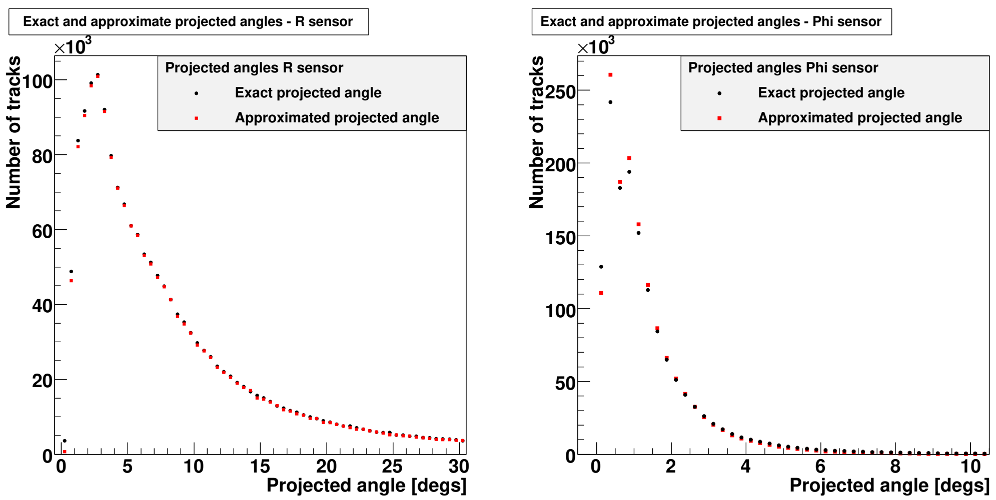


Figure 6: Distributions of the exact and approximated projected angle are presented for R (left hand side) and Phi (left hand side).

An event by event comparison between the exact and approximate values of the projected angle is shown on Fig. 6. As can be seen from the figure the approximation works excellently for the R sensor and is reasonably good for the Phi geometry, the only notable differences being for the first few bins. This conclusion is confirmed by the correlation plots on Fig. 7. Only 2% of tracks have a difference between the exact and approximate methods of over one degree. A difference of one degree between the exact and approximate methods results in an error of less than 1 μm on the estimated position error.

The difference between the exact and approximate angle $\Delta\alpha = \alpha_p^{exact} - \alpha_p^{app}$ is shown in Fig. 8 where the

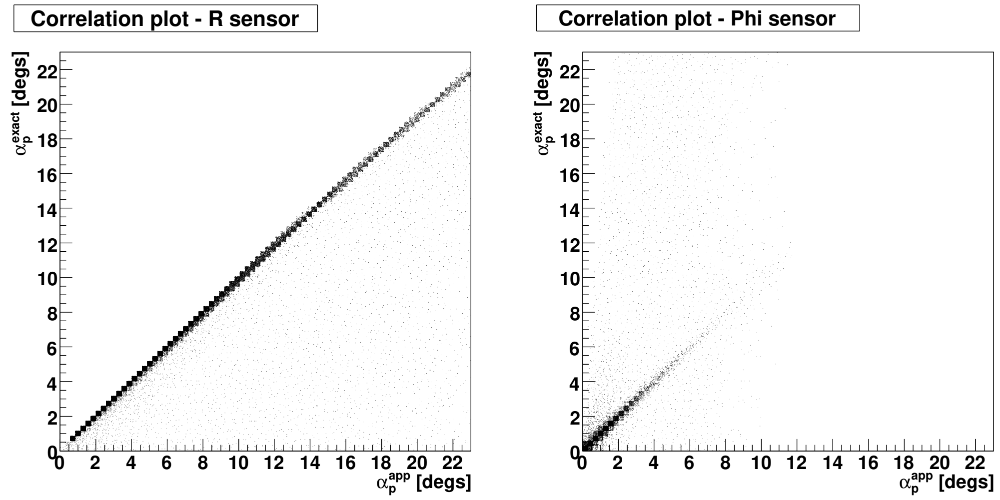


Figure 7: Correlation plots for projected angle for both types of the VELO sensors are shown (additional cut for the track angle $\alpha_t < 400$ mrad was applied in both cases).

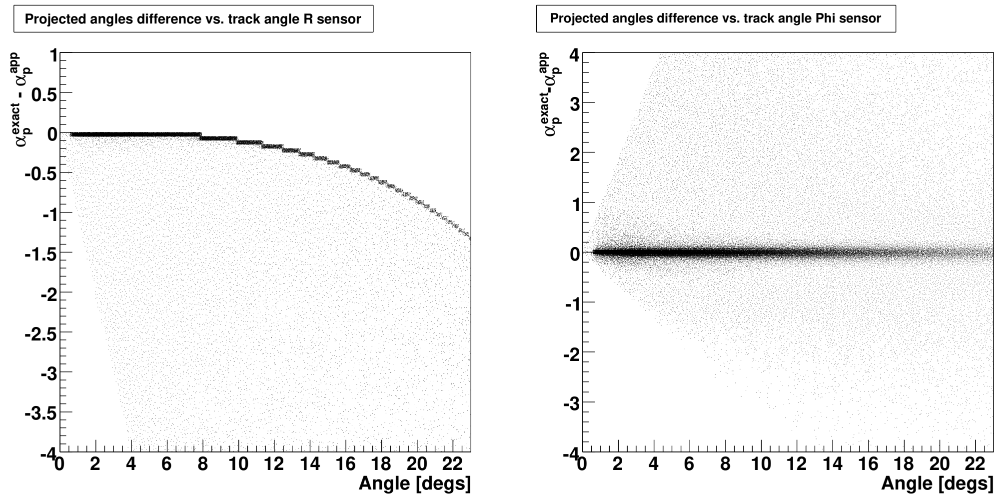


Figure 8: Difference between exact values of the projected α_p^{exact} and approximate α_p^{app} angles plotted against track angle α_t. Left hand side plot corresponds to R geometry and the right hand side one to Phi.

variable is plotted against the track angle. The main component of the difference for the R sensor is due to the small angle approximation in calculating the polar angle slope of the track.

The CPU time taken by the method is of importance due to the heavy usage of the tool during the reconstruction of the LHCb events. The code is called many times for each event depending on the number of reconstructed clusters, there are typically 1000 clusters per event. The extra time taken by the exact compared to the approximate method is approximately 3 μs per call measured on an Intel Duo Core 2 processor running SLC 4.4 Linux.

4 Error Parametrizations

The cluster position tool error estimate is parametrised in terms of the local pitch of the sensor and the projected angle of the track. This 2D error parametrization may be created in two ways : either a collection of 1D functions of projected angle determined for different pitch bins; or 1D functions of pitch for different projected angle bins. Both approaches, along with resolution determinations, will be described below.

4.1 Residuals and Intrinsic Resolution

The residual is defined as distance between the particle entry point in the silicon and the reconstructed cluster centre. The distance is measured perpendicular to the direction of the strips in the sensor, and can be both positive and negative.

In the study presented here the particle entry point is taken from the true entry point of the simulated particle, this was used to decouple the study from problems with the LHCb track fit at the time of development. For future data studies the unbiased residual using the fitted tracks (without the measurements in the sensor under study included) will be used.

Examples of residual distributions for R and Phi sensors for all pitches and all projected angles, hence giving rise to the non-Gaussian shape, are presented in Fig. 9.

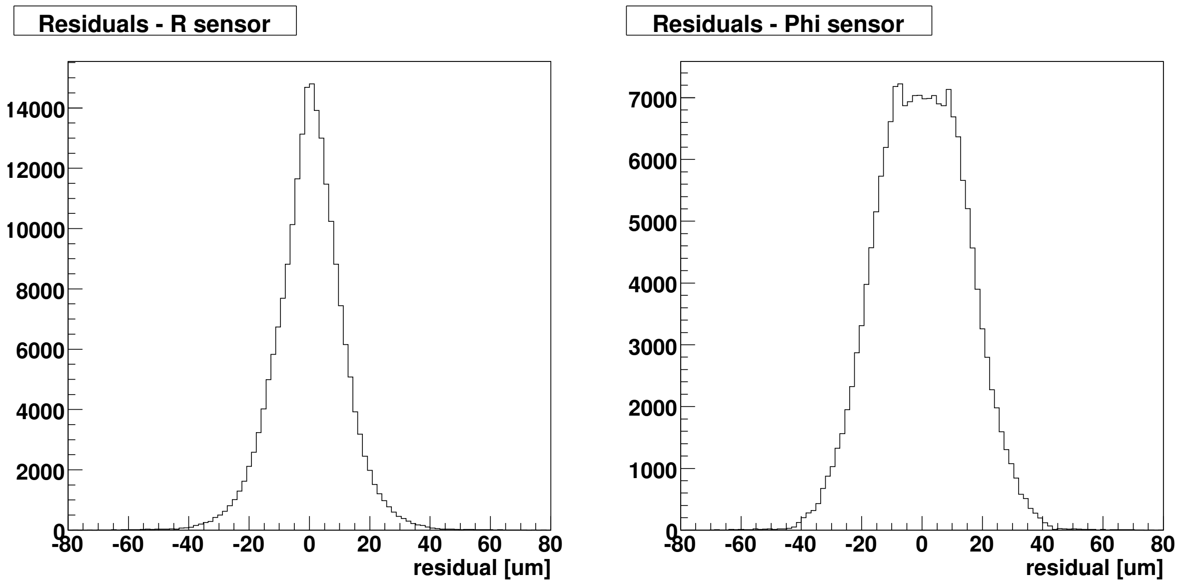


Figure 9: Residual distributions for R and Phi sensor.

Here the resolution of the VELO sensor has been defined as the RMS of the residual distribution. Alternatively the σ of a Gaussian curve fitted to the residual distribution may be used. The two estimations can differ significantly if there are many outliers. The outliers are due to tracks with large angle. In order to get rid of those outliers a cut for the track angle $\alpha_t < 400$ mrad was applied in both cases.

In the following part of this section the definition of the error parametrizations used and their extraction procedure from simulated data are given. The errors have been calculated both as projections on pitch and on projected angle. Both types of parametrizations are available in the tool and may be used as a cross-check.

As no difference is observed between the R and Phi sensor resolution as a function of pitch and projected angle in test-beam data studies (or in the simulation) only the R sensor resolution plots are presented here.

4.1.1 Resolution as Function of the Pitch

Conventionally the VELO plots the resolution as a function of pitch − in this projection the resolution is a simple linear function of the sensor pitch. Typical resolution plots for (three) different projected angle bins as a function of the sensor's pitch are presented in Fig. 10.

The most obvious feature of the resolution is its perfectly linear dependence on the pitch. The resolution improves as a function of the projected angle up to $\alpha_p \approx 15°$ where the best resolution is obtained. For larger projected angles the resolution starts to degrade since the number of clusters with more than two strips rises significantly. For projected angles greater than 20° the resolution saturates and become a flat function of the pitch. VELO testbeam data was taken in 2004 with a range of projected angles and, within the limitations of the data taken, the analysis of this has verified the prediction of the simulation.

In order to provide a full 2D continuous resolution parametrization as a function of the pitch and projected angle a collection of linear resolution functions determined for each projected angle bin are used. Each resolution plot can be fitted with the following analytical function:

$$res^{1D}(pitch)_{\alpha_p=const} = p_0 + p_1 * pitch \equiv err^{1D}_{pitch}. \tag{10}$$

For each projected angle bin a pair of p_0 and p_1 parameters are obtained. The dependency of these extracted parameters on the projected angle can be found, as shown in Fig. 11.

This leads us to continuous 2D resolution parametrization of following form:

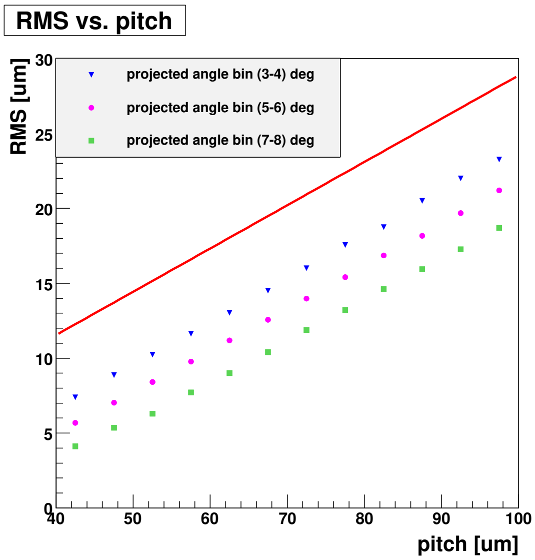


Figure 10: VELO silicon sensor resolution plotted against the pitch for three projected angle bin (red line represents binary resolution).

$$res^{2D}(pitch, \alpha_p) = P_0(\alpha_p) + P_1(\alpha_p) * pitch \equiv err^{2D}_{pitch}. \tag{11}$$

Although the resolution as a function of the pitch has very convenient linear form, the p_0 and p_1 parameters are complex functions of the projected angle. An accurate fit of these functions is obtained using splines.

4.1.2 Resolution as Function of the Projected Angle

Another option is to build resolution parametrizations using projections along pitch bins. Figure 12 presents typical resolution plots as a function of the projected angle (for a comparison see Fig. 10).

Each such curve shows a single minimum at a different projected angle which clearly depends on the pitch. For pitches in the range $55 - 60$ μm resolution the best resolution in the simulation is obtained at $\alpha_p \approx 12°$ while for pitches in the range $95 - 100$ μm the best resolution is at $\alpha_p \approx 20°$. The curves show significant resolution degradation for large projected angles.

In order to build a continuous 2D resolution parametrization one can use the recipe described in the previous section. The resolution curves are fitted with the following function:

$$res^{1D}(\alpha_p)_{pitch=const} = p_0 + p_1 * sin(p_2 * \alpha_p + p_3) \equiv err^{1D}_{\alpha_p}. \tag{12}$$

For each pitch bin four parameters p_i are extracted. Again, by plotting them against pitch we can express them as functions $P_i = P_i(pitch)$ and use them to build 2D resolution parametrizations. This time a convenient analytic expression can be used to fit the parameters:

$$P_i(pitch) = a^i_0 + a^i_2 * pitch, \tag{13}$$

and the corresponding 2D parametrization can be written out as:

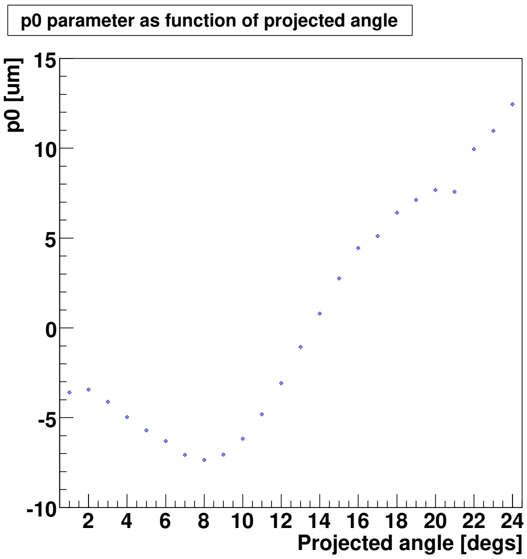
$$res^{2D}(pitch, \alpha_p) = P_0(pitch) + P_1(pitch) * sin(P_2(pitch) * \alpha_p + P_3(pitch)) \equiv err^{2D}_{\alpha_p}. \tag{14}$$

4.2 Pull Distributions

The error parametrizations err^{2D}_{pitch} and $err^{2D}_{\alpha_p}$ have been tested by producing the pull distribution for residuals with simulated data, where the pulls are defined as:

$$pull = \frac{residual}{err}. \tag{15}$$

In order to make sure that our error parametrizations are consistent the pulls are presented both as a function of projected angle and of pitch. According to equation 15 we expect to get for each projected angle (pitch) bin pull distributions with a mean value close to 0 and with an RMS close to 1. These values are plotted in Fig. 13



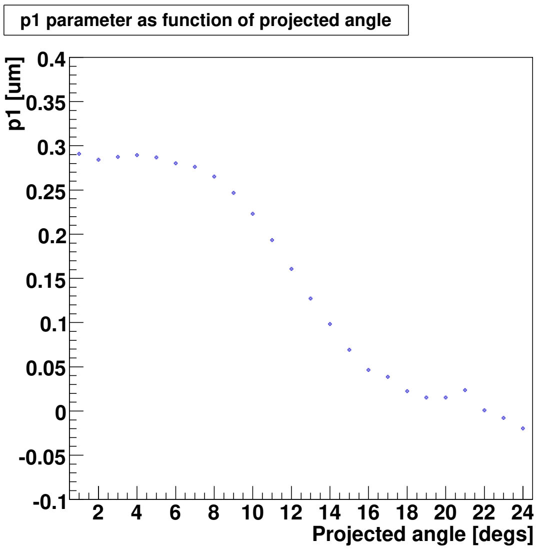


Figure 11: VELO silicon sensor resolution in the simulation plotted against the pitch for three projected angle bins (the red line represents binary resolution).

and 14 for both err^{2D}_{pitch} and $err^{2D}_{\alpha_p}$ parametrizations. Good agreement is obtained for both parametrisations, with a slight preference for the err^{2D}_{pitch} parametrisation.

5 C++ Code

This section presents the most important details of the VELO cluster tool's C++ code implementation. The tool follows the standard Gaudi framework rules for the tool pattern.

5.1 Interface

The first part of the tool is a pure virtual interface that inherits from the IAlgTool class. The VELO cluster tool's interface is of type $IVeloClusterPosition$ and is located in the $TrackInterfaces$ package. All the tool's functionality needs to be declared inside the class as pure virtual methods. $IVeloClusterPosition$ contains:

```
virtual toolInfo position(const LHCb::VeloCluster* aCluster) const;

virtual toolInfo position(const LHCb::VeloCluster* aCluster,
                          const Gaudi::XYZPoint& aPoint,
                          const Direction& aDirection) const;

virtual toolInfo position(const LHCb::VeloCluster* aCluster,
                          const LHCb::StateVector& aState) const;
```

These three methods each return a $SiPositionInfo$ templated structure.

```
typedef LHCb::SiPositionInfo<LHCb::VeloChannelID> toolInfo;
```

The structure contains three data members: VeloChannelID of the seeding strip of $VeloLiteCluster$ object; fractional position; and fractional error. The $SiPositionInfo$ object may also be used to define the position of a cluster in the silicon tracker as well as in the VELO.

The accuracy of the position error estimate returned by the tool depends on which of the three methods is called. For the first method, when only the clusters are available, the error estimate is based on a parametrization determined for the mean value of the projected angle of tracks in LHCb. This may lead to a very distorted pull distribution as a function of the projected angle. Whenever possible, i.e. once a track estimate is available, the two other position methods should be used. With additional information about the track's state (slopes s_{xz}, s_{yz} and a space point) the error estimate is much more accurate and a better performance can be obtained from the track fitting procedure.

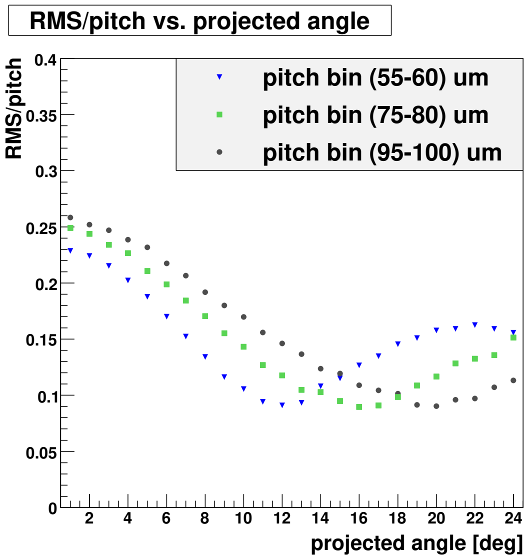


Figure 12: Resolutions plotted against the projected angle.

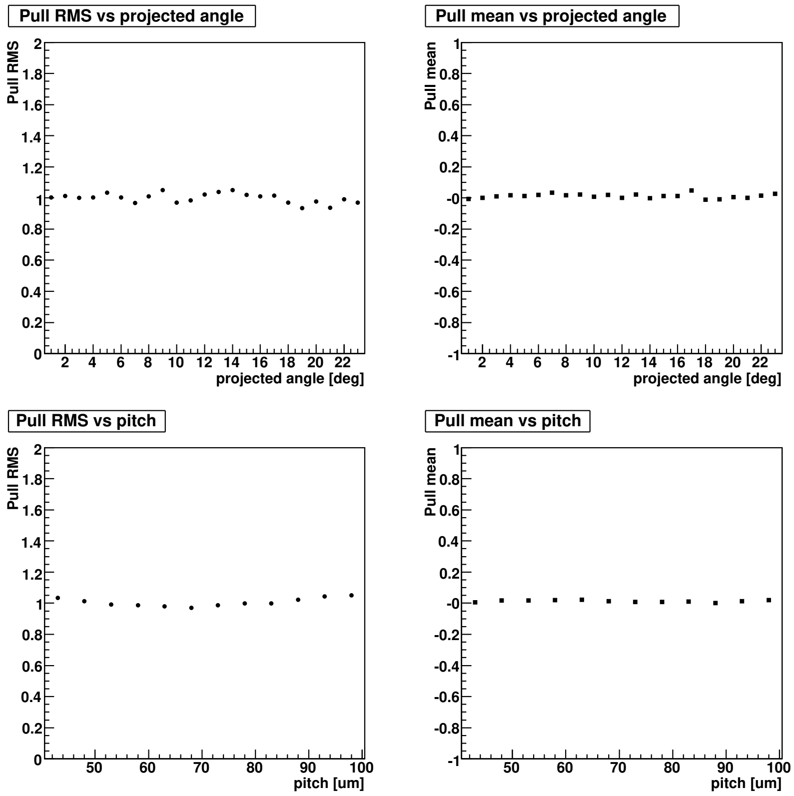


Figure 13: RMS and mean values of the residual pulls from simulation plotted as function of the projected angle (upper plot) and the pitch (lower). The RMS values correspond to err^{2D}_{pitch} parametrization.

5.2 Implementation and Use Case

In order to use the tool one needs to include in one's header algorithm file the cluster tool interface

```
#include Kernel/IVeloClusterPosition.h
```

and define a pointer to the object of type $IVeloClusterPosition$ as a data member of the algorithm. The tool can then be retrieved inside the implementation file of the algorithm in the standard way using:

```
m_clusterTool=tool<IVeloClusterPosition>("VeloClusterPosition");
```

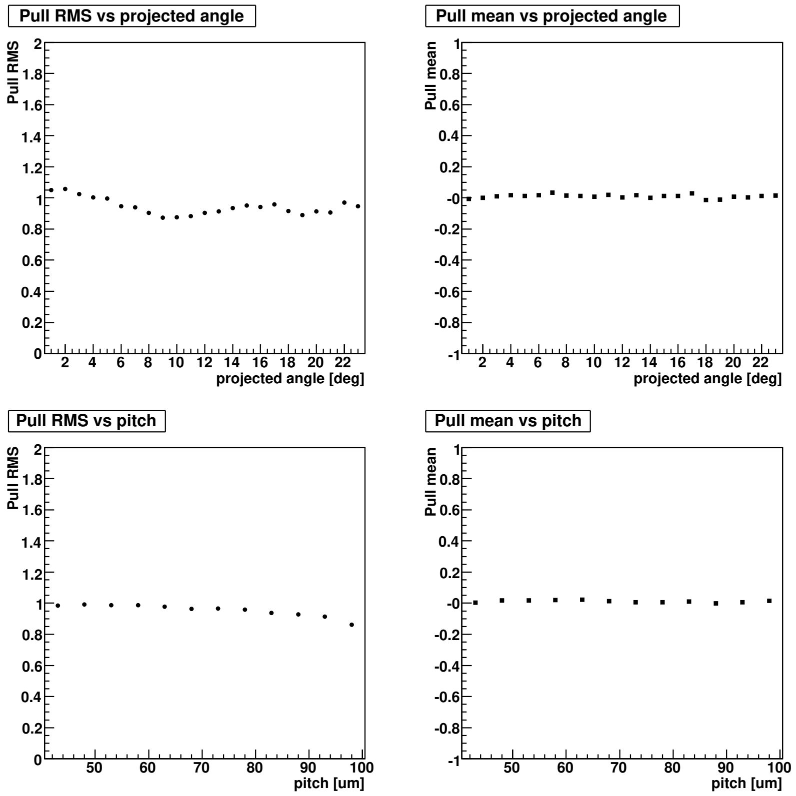


Figure 14: RMS and mean values of the residual pulls from simulation plotted as functon of the projected angle (upper plot) and the pitch (lower). The RMS values correspond to $err^{2D}_{\alpha_p}$ parametrization.

where m_clusterTool is defined as a pointer to the tool interface. To retrieve the position and error information one may use the toolInfo definition from the interface:

```
IVeloClusterPosition::toolInfo anInfo;
```

and subsequently:

```
anInfo=m_clusterTool->position((*cluIT), globalEntry, aDirection);
```

where the first argument is a pointer to a cluster, the second is a space point expressed in the global LHCb reference frame and the last one represents the track slopes. The global space point is of the type:

```
Gaudi::XYZPoint
```

and the slopes are encapsulated inside an object defined inside interface class:

```
IVeloClusterPosition::Direction aDirection;
```

where the *Direction* is of the type:

```
std::pair<double, double>
```

Having retrieved a toolInfo object, one can determine the absolute position and its uncertainty from the fractional position and fractional error by using the local pitch value. The local pitch can be extracted from the detector element via the channel identifier of the seeding strip. Hence, both the fractional position and error can be expressed in μm for both R and Phi sensors.

6 Conclusions

This note provides a detailed description of the VELO cluster position tool and its pull performance on simulation events.

Studies of the residuals demonstrated their strong dependence on both the silicon sensor's pitch and projected angle. A parametrization of the VELO cluster position error as a function of the pitch only, as previously used in the LHCb tracking, is not sufficient for high quality track parameter determination.

Continuous two dimensional paramterisations of the cluster position error are provided. Two versions have been implemented. In the first, a collection of one dimensional parametrizations as a function of pitch are produced in bins of projected angle. In the second, a collection of one dimensional parametrizations as a function of projected angle are produced in bins of pitch. In both versions two dimensional parametrizations are then obtained. The parametrisations have been extracted from simulation events and pull distributions shown. Both parametrizations are available through option file settings. The default option is that obtained using pitch parametrisations in projected angle bins.

Additionally, due to heavy usage of the tool during reconstruction of the LHCb events some simplifications have been introduced to decrease the CPU time. The most important ones are related with the computation of the projected angle. Approximate methods are demonstrated for both the R and Phi sensors. Detailed studies of the approximations' quality have also been performed and this proved them to be acceptable for both VELO sensor geometries. Both the exact and approximate projected angle methods are accessible via options file settings, the default setting is to use the faster approximate calculation.

Future work will be carried out in the VELO group to improve the accuracy of the cluster position accuracy, and to extract the cluster position error parametrisations from data.

A Reference Frame Transformations

In the track state all the geometrical properties related with a track are expressed in the global reference frame [6]. In order to use this information for the cluster resolution calculation on a single sensor appropriate transformations from the global to the local sensor reference systems must be applied. These transformations are described in this section.

A.1 Global LHCb Reference Frame

The global reference frame is the standard cartesian LHCb reference frame which is defined as follows: the Z_{global} axis is along the nominal beam axis; the Y_{global} axis points up with respect to the cavern floor; and the X_{global} axis is defined to make a right-handed reference frame. In this reference frame the left half (or A side) of the VELO detector is always at $x > 0$. It is customary to call the direction towards increasing Z "downstream" and the opposite direction "upstream".

A.2 Local VELO Reference Frames

Neglecting misalignments, the VELO sensors have four distinct orientations in the global reference frame.

The cartesian VELO sensor local reference frame has the Z_{local} axis perpendicular to sensor surface - the axis direction is such that the implanted strip surface is at $z_{local} > 0$ and the back plane at $z_{local} < 0$. The local X_{local} is chosen so that (almost) all strips are at $x_{loc} > 0$ and the Y_{loc} is defined to make the local reference system right-handed. This is shown in Fig 15.

In the global frame the sensor strip plane may face in both the upstream or downstream directions, and the sensor is mounted in the left or right-half of the VELO. These four orientations of the local reference frame in the global frame are shown in Fig 16. Each local reference frame is designated, for the sake of simplicity of further discussion, by two letters - the first one corresponds to a detector half (L for left and R for right) and the second describes whether the strip implant plane of a sensor faces upstream or downstream (U and D respectively).

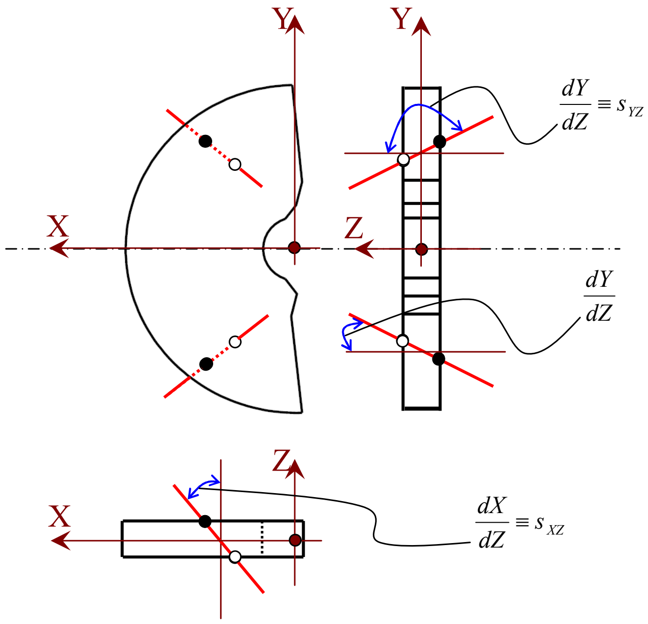


Figure 15: Definition of the local reference frame's origin point is shown along with the slopes for two different tracks (Phi sensor is depicted). The transformation from the global to local coordinate frame may cause a change of the actual value of the slopes.

A.3 Transformations from Global to Local reference frames

A space point and the track slopes are obtained from the tracks class in the global frame. These must be transformed into the local frame for use in the projected angle calculation.

The transformation of space points from local to global geometry is available via the `DeVelo` class interface. This transformation for slopes is not available and needs to be implemented within the VELO cluster position tool.

In order to get the appropriate values for the slopes of a track in the local reference frame the following general transformation is made:

$$S_i^{(local)} = T_{ij}^{(XX)} \cdot S_j^{(global)}, \tag{16}$$

where $S^{(local)}$ and $S^{(global)}$ are the slope vectors expressed in the local and global reference frame respectively, $T_{ij}^{(XX)}$ is a transformation matix and XX signifies the local frame type (see Fig. 16). Simple geometrical analysis leads to following transformation matrices:

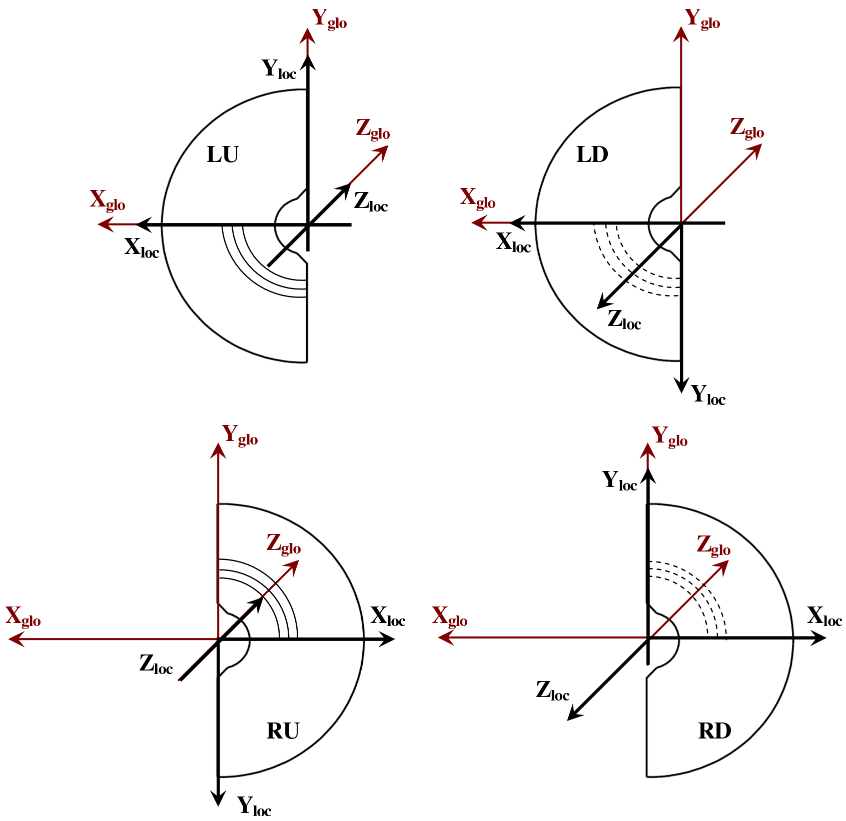


Figure 16: Definition of the local reference frames for the VELO sensors.

$$T_{ij}^{LU} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \; T_{ij}^{LD} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$T_{ij}^{RU} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \; T_{ij}^{RU} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

This neglects the effect of misalignments. The level of the relevant misalignments, rotations around the X or Y axes, in the assembled VELO system is of order of few mrads and hence can be neglected.

References

[1] http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/boole

[2] G. Haefeli at al, 'TELL1 - specification for a common read out board for LHCb', LHCb-2003-007

[3] http://lhcb-comp.web.cern.ch/lhcb-comp/Frameworks/Gaudi/GaudiTutorial.htm

[4] VELO raw data format and strip numbering, EDMS note 637676 v.2

[5] http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/brunel

[6] Definition of the Coordinate System, EDMS note LHCB-C-EN-0001 v.1

[7] T. Szumlak, C. Parkes, 'Velo Event Model', LHCb-2006-054

[8] G. Heafeli, A. Gong, LHCb Velo and ST clusterization on TELL1, EDMS 690585

[9] T. Huse et al., Position Reconstruction and Charge Distribution in LHCb VELO Silicon Sensors, LHCb-2007-119