LHCb 2008-026

LPHE 2008-004

Identification of Ghost Tracks using a Likelihood Method

M. Needham

Laboratoire de Physique des Hautes Energies,
École Polytechnique Fédérale de Lausanne

May 28, 2008

Abstract

The implementation of an algorithm based on a likelihood method to discriminate between real and ghost tracks is described. This algorithm is shown to perform slightly better than a cut on the χ^2/ndof.

1 Introduction

The high track densities expected in the LHCb detector present a challenging enviroment for track reconstruction. During the pattern recognition step many pseudo-random combinations of hits, refered to as ghosts, are formed. The efficiency of LHCb tracking algorithms are limited by the number of ghosts that can be tolerated. This is particularly true for luminosities higher than the nominal LHCb running value of 2×10^{32} cm^{-2}s^{-1}. In this note a likelihood based method to identify ghost tracks is presented. Such an approach is easy to implement and generic.

The layout of this note is as follows. First, the algorithm is described. Next the performance for long tracks is summarized for luminosities up to 20 $\times 10^{32}$ cm^{-2}s^{-1} and compared to a simple cut on the χ^2/ndof. The performance for other track types is then discussed. Finally, the implementation in the LHCb software is described in an appendix.

2 Algorithm

Ghost tracks are characterized by having a low probability of χ^2 from the track fit and missing hits [1]. In addition, ghosts due to spillover from previous crossings can be distinguished by having lower deposited energy in the silicon detectors. To create a discriminant this information is combined into a log-likelihood using the approach described in [2]. For each detector layer it is checked whether the track is expected to give a hit. A binomial counting term is then calculated according to the number of hits found (n) compared to the number expected (N). This gives a contribution to the log-likelihood:

$$lnL = ln\frac{N!}{n!(N-n)!} + nln\epsilon + (N-n)ln(1-\epsilon) \tag{1}$$

where ϵ is the product of the detector efficiency and the efficiency to add the hit to the track. For the silicon detectors similar counting terms are added to the likelihood based on how many hits with the high threshold bit are observed compared to the number expected based on the total number of clusters on the track. Table 1 - 2 summarize the efficiencies assumed for the various types of tracking hits.

For the Outer Tracker the cell efficiency decreases towards the wall of the straw and a layer is insensitive to tracks that pass through the gap between two cells.

Term	**Probability (ϵ)**	**Source**
Velo r	0.97	PatChecker
Velo ϕ	0.94	PatChecker
TT	0.9	LHCb note 2007-024
IT	0.99	LHCb note 2007-024
OT	0.94	PatChecker

Table 1: Probabilities assumed for the various counting terms.

Term	**Probability (ϵ)**	**Spillover Probability (ϵ)**	**Source**
Velo	0.99	0.10	Private code
TT	0.91	0.16	LHCb note 2007-024
IT	0.94	0.06	LHCb note 2007-024

Table 2: Probabilities assumed for the counting terms related the high threshold bit. For comparison the corresponding numbers for spillover clusters are shown.

Therefore, in layers with missing hits the track trajectory should be such that it has large distance to the centre of the straw. This leads to a term in the likelihood of the form [2]:

$$L = 0.261 + exp(5.1r - 11.87) \tag{2}$$

where r is the distance of the track from the wire. This contribution is shown in Fig 1.

Finally, the probability of χ^2 given by the track fit is added to the likelihood. For real tracks this contribution has the form shown in Fig. 2. The accumulation of tracks with low probability of χ^2 generates a long tail in the log-likelihood distribution. Potentially, this spoils the performance of the algorithm. In [1] it was suggested to de-weight this contribution by a factor that is empirically determined to be around three. Another approach, which will be investigated here, is to use the shape of the distribution in Fig. 2. This can be modeled by a β function :

$$\beta(p) = \frac{1}{Z(u_1, u_2)} p^{u_1 - 1}(1-p)^{u_2 - 1}$$

with $u_1 = 0.63$ and $u_2 = 0.86$. The contribution of the χ^2 to the likelihood is then:

$$L = \int_0^{p(\chi^2)} \beta(p(\chi^2))dp \tag{3}$$

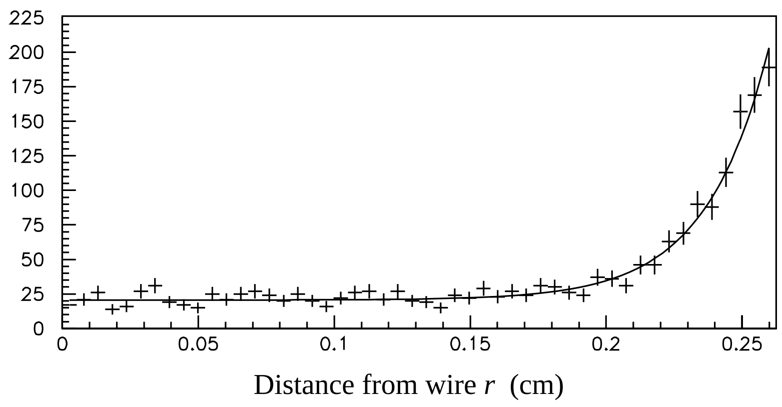

Figure 1: Histogram of the distance of the track trajectory from the wire, for OT cells that did not give a hit; the parametrization used in the likelihood calculation is superimposed.

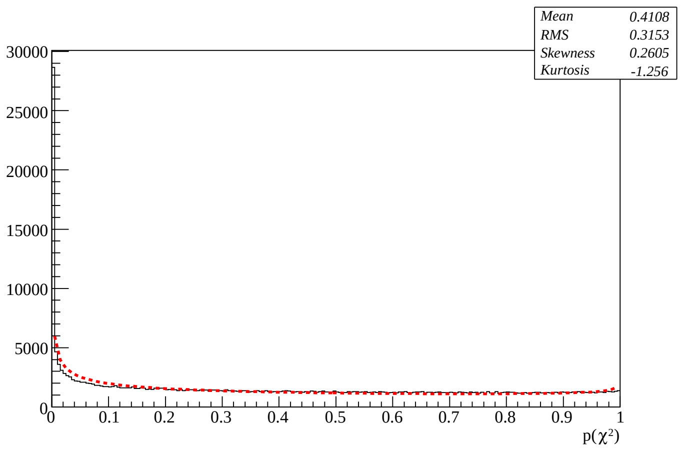

Figure 2: Probability of χ^2 for long tracks. The dotted line is the result of a fit to a β function.

3 Results

The performance studies were done using the following data samples:

- A sample of 24000 $B_d \rightarrow J/\psi(\mu^+\mu^-)K_S(\pi^+\pi^-)$ events generated at the default LHCb luminosity of 2×10^{32} cm^{-2}s^{-1}.
- Samples of 500 inclusive b events generated at luminosities of 5, 8, 10 and

20×10^{32} cm^{-2}s^{-1}.

All the samples were reconstructed with Brunel v31r11.

3.1 Long Track Studies

For these studies the definitions of long track acceptance, efficiency and ghost rate given in [1] were used. Fig. 3 shows the likelihood distribution for real long tracks within the LHCb acceptance and ghosts. A clear separation is observed. From this plot it seems that there is a component of the ghost's that look like the real tracks. This is consistent with the observation [3] that there is some fraction of the ghost rate due to real tracks, which are not correctly associated to Monte Carlo truth.

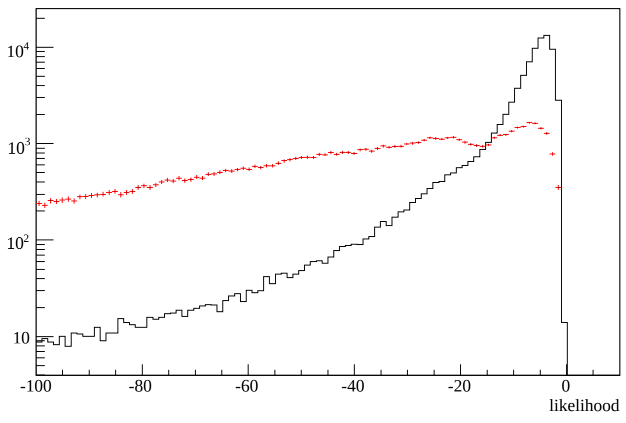

Figure 3: Likelihood distribution for reconstructed long tracks within the LHCb acceptance (solid line) and ghosts (points).

Fig. 4 shows the efficiency versus ghosts rate for the various cuts on the likelihood method and also for a cut on the χ^2/ndof. It can be seen that the difference in performance of the two methods is small. This shows that the χ^2/ndof is already a powerful discriminant and that the additional gain in performance from the likelihood technique is small. In Fig 5 the effect of re-weighting the χ^2 contribution according to Equation 3 is shown. The effect of the re-weighting is small. Therefore, this approach will not be considered further in this note [1].

[1]The approach used in [2] of simply dividing the χ^2 contribution by three was also studied. It was also found to have only a small effect on the results.

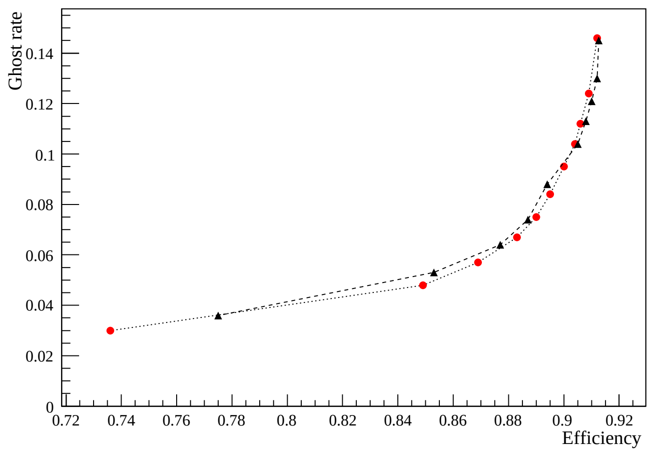

Figure 4: Efficiency versus ghost rate as a function of a cut on the χ^2/ndof (black triangles) and the likelihood (red points). The points on the χ^2/ndof graph from left to right correspond to cuts at 1.5, 2, 2.5, 3 , 4, 6, 8, 10, 15, ∞. Those on the likelihood graph correspond to cuts at -10, -20, -30, -40, -50, -60, -70, -80, -90, -100 , $-\infty$.

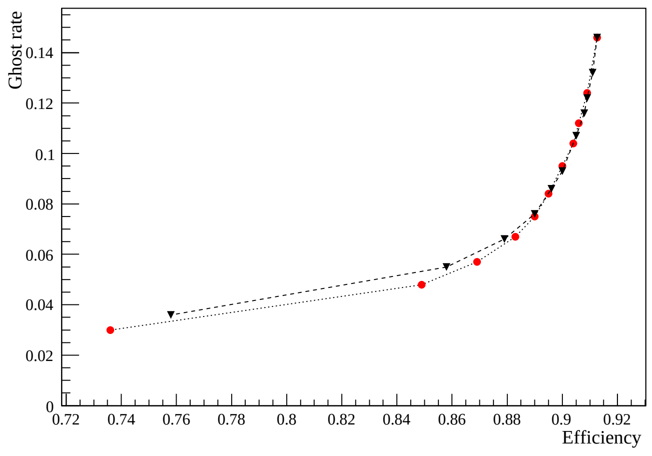

Figure 5: Efficiency versus ghost rate as a function of a cut on the likelihood (red points) and the case of reweighting the contribution from the track $p(\chi^2)$ with a β function (black triangles).

3.2 Performance as function of luminosity

The performance of the likelihood method and the χ^2/ndof cut have also been compared as a function of luminosity. Based on the studies in the previous section it was decided to apply a cut on the χ^2/ndof at 6 or a cut on the likelihood at -60. In both cases at a luminosity of 2×10^{32} cm^{-2}s^{-1} this reduces the ghost rate to 10.4 % for a 1 % loss in efficiency. The efficiency versus luminosity is shown in Fig. 6. For both methods, compared to the case of no cut, an increasing loss in efficiency is observed at higher luminosities. Due to the larger hit density there is an increased probability to wrongly assign hits to a track. This leads to a degradation of the fit quality which can be seen in Fig. 7 where the mean probability of χ^2 is observed to decrease as the luminosity increases.

Fig. 8 shows the ghost rate versus luminosity. For luminosities above 10^{33} cm^{-2}s^{-1} the likelihood method gives a lower ghost rate than the χ^2/ndof cut.

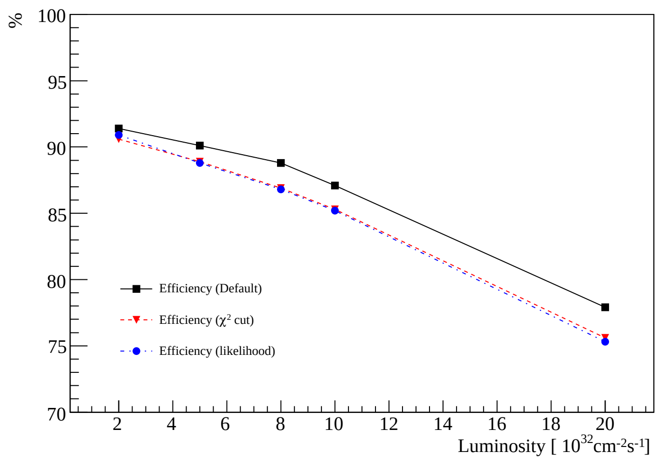

Figure 6: Efficiency versus luminosity. Note the suppressed scale.

3.3 Performance for other track types

Since the algorithm is generic it can be applied, without modification, to the output of other tracking algorithms. Fig 9 - 11 show the results for the Downstream tracking [4], the Velo-TT tracking [5] and the standalone T-seeding based on the Tsa reconstruction framework [2, 6, 7]. It can be seen that in all cases the algorithm allows the ghost rate to be reduced for a small loss in inefficiency. As

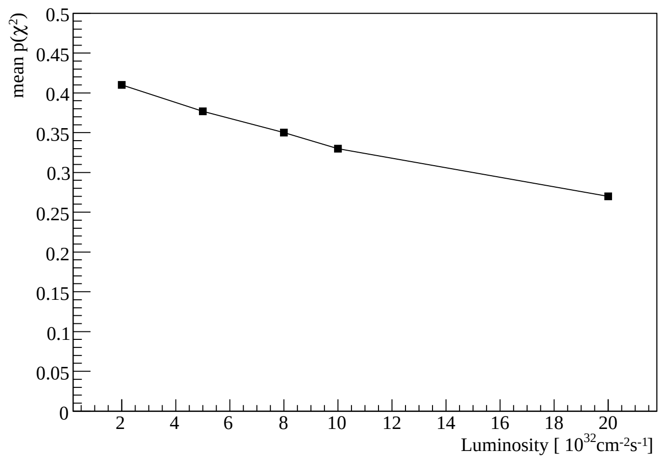

Figure 7: Mean $p(\chi^2)$ versus luminosity.

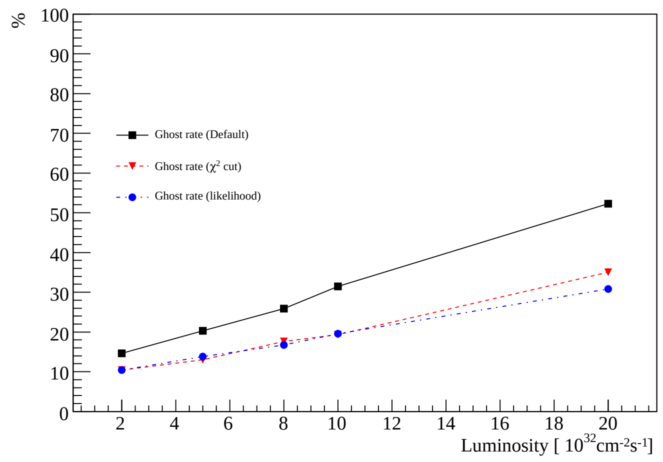

Figure 8: Ghost-rate versus luminosity.

was seen for the case of the long tracking the power of the algorithm is diluted by the tail in the probability of χ^2 distribution. The biggest separation of power is seen for the Tsa seeding and the least for the Velo-TT tracking. The fact that poor separation is seen in the latter case is not surprising. First, this track type provides limited information since it is formed only from Velo and TT hits. In addition, inside the algorithm the track fit is run and used to discriminate between competing candidates. This means that internally the algorithm already cuts on one of the quantities used in the likelihood. However, the fact that some separation is seen indicates that the use of the available information by the algorithm is not optimal. Therefore, the performance could be improved by implementing a competition between candidates based on the likelihood

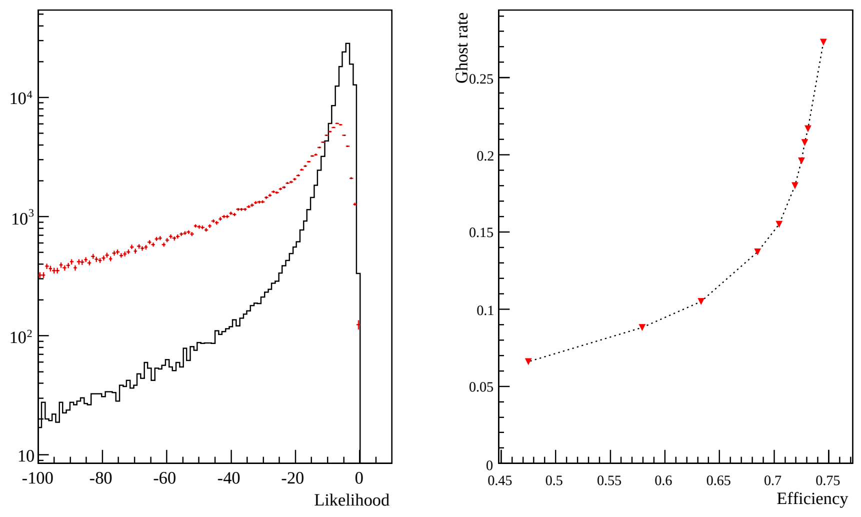

Figure 9: Likelihood for ghosts (points) and real (solid) tracks for the Downstream Tracking algorithm (left) and efficiency versus ghost rate (right). The points from left to right correspond to cuts at -6, -8 , -10, -15, -20, -30, -40, -50, -60, $-\infty$.

4 Summary

In this note an algorithm for discriminating between real and ghost tracks based on a likelihood technique has been studied with simulated data. It has been shown that this approach leads to an improvement in the discrimination between real and ghost tracks compared to a simple cut on the χ^2/ndof of the track. However, the improvement is small because the χ^2/ndof is already a powerful discriminant. With real data the power of the χ^2/ndof may be reduced because of incorrect modeling of hit resolution and material in the detector. In this case the likelihood approach, by adding more information could be more performant.

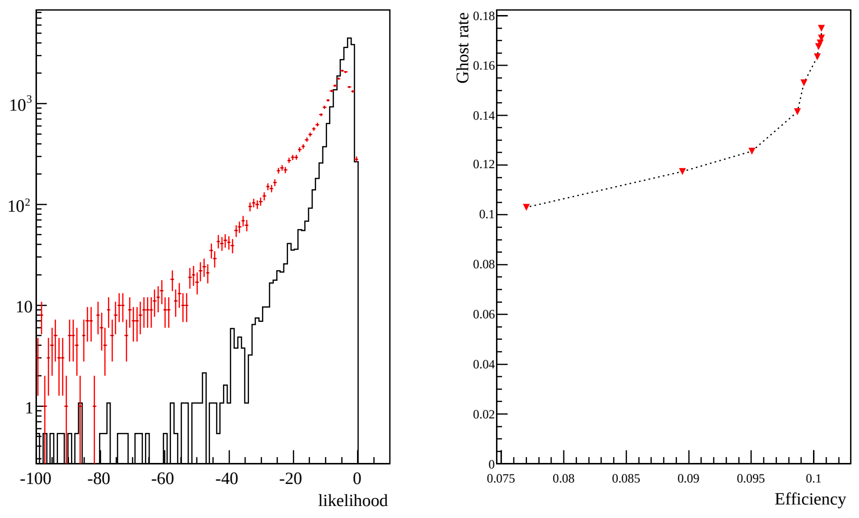

Figure 10: Likelihood for ghosts (points) and real (solid) tracks for the Velo-TT algorithm (left) and efficiency versus ghost rate (right). The points from left to right correspond to cuts at -6, -8 , -10, -15, -20, -30, -40, -50, -60, $-\infty$. *Nota Bene*, the low efficiency of this algorithm is because VELO track segments used in the long tracking are excluded before the algorithm is run whilst the efficiency is normalized to all tracks in the Velo-TT acceptance.

In both approaches the performance is limited by the tail in the probability of χ^2/ndof distribution. A better treatment of this tail would lead to improved performance. Two effects are known to contribute to this tail:

- Wrongly resolved L/R signs in the Outer Tracker [1].
- Wrongly assigned hits from nearby tracks (Section 3.2).

Further work is needed to improve the treatment of both effects in the track fit.

References

[1] M. Needham. Performance of the LHCb Track Reconstruction Software. LHCb-note 2007-144.

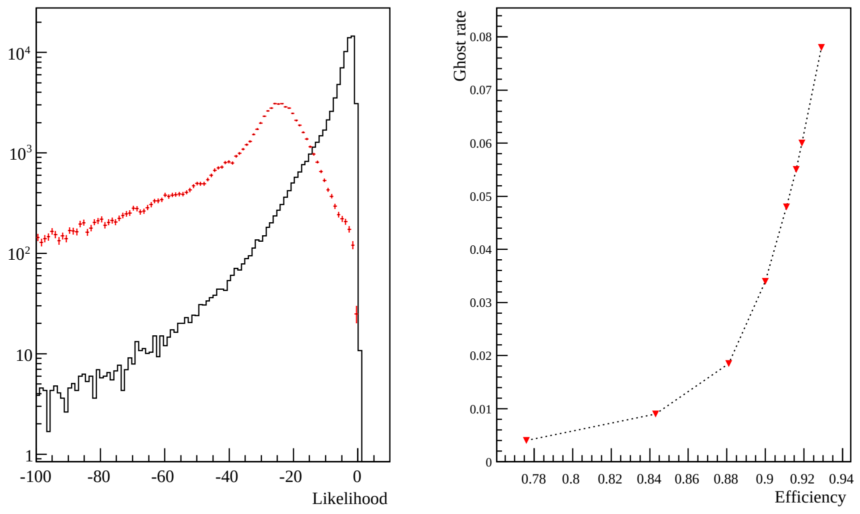

Figure 11: Likelihood for ghosts (points) and real (solid) tracks (left) and efficiency versus ghost rate (right) for the Tsa seeding algorithm. The points from left to right correspond to cuts at -10, -15, -20, -25, -30, -35, -40, $-\infty$

[2] R. Forty M. Needham. Standalone Track Reconstruction in the T-Stations. LHCb-note 2007-022.

[3] M. Needham. Classification of Ghost Tracks. LHCb-note 2007-129.

[4] O. Callot. Downstream Pattern Recognition. LHCb-Note 2007-026.

[5] O. Callot *et. al.* VELO-TT track reconstruction. LHCb-note 2007-010.

[6] R. Forty and M. Needham. Updated Performance of the T seeding. LHCb-note 2007-023.

[7] M. Needham. The Tsa Reconstruction Framework. LHCb-Note 2007-037.

A Implementation of the Likelihood tool

The algorithm described above has been implemented as a tool in the TrackTools package. The code fragments below illustrate how to get and use the tool:

```
#include ``TrackInterfaces/ITrackManipulator.h''

// get the tool
ITrackManipulator* likTool = tool<ITrackManipulator>(``TrackLikelihood'');

// use it
Track aTrack;
double lik = likTool->execute(aTrack);
```

For a long track the calculation takes 0.5 ms on a 64 bit 2.8 GHz Opteron processor.

Internally the tool delegates the task of calculating the number of expected hits to two tools: the **IHitExpectation** in the case of IT, TT and OT and the **IVeloExpectation** tool for the VELO. These tools are described in Appendix B. The quality of the results given by these tools depends on what **States** are present on the **Track**. Therefore, the accuracy of the result will be higher in Brunel where **States** at the z positions of all measurements are present. In addition, the information on the number of hits in the Silicon detectors above the high threshold is extracted from the **Measurements** on the track — which are not stored on the DST. These considerations mean the algorithm is run in Brunel and the output added to the **ExtraInfo** map of the track class by the **TrackAddLikelihood** algorithm and stored on the DST. For completeness the number of hits expected in each of the tracking detectors are also stored. The following code fragment show how gets and use this information from the track:

```
Track aTrack;
double lik = likTool->info(LHCb::Track::Likelihood, 9999);
if ( lik < cutValue){
  // likely to be a ghost
}
```

B Expected Hit Tools

Two interfaces have been written that allow the number of expected hits in the tracking detectors to be estimated. The first is used by the OT, IT and TT and has the interface below:

```
class IHitExpectation: virtual public IAlgTool{

public:

  /** small struct returning hit info....
  */
  typedef struct {
    unsigned int nExpected;
    unsigned int nFound;
    double likelihood;
  } Info;

  /// Retrieve interface ID
  static const InterfaceID& interfaceID() { return IID_IHitExpectation ; }

  /** Returns number of hits expected, from zFirst to inf
  *  @param aTrack Reference to the Track to test
  *  @return unsigned int number of hits expected
  */
  virtual unsigned int nExpected ( const LHCb::Track& aTrack ) const = 0;

  /** Returns number of hits expected
  *  @param aTrack Reference to the Track to test
  *  @return Info info including likelihood
  */
  virtual Info expectation( const LHCb::Track& aTrack ) const = 0;
};
```

As can be seen two methods are defined in the interface. The first simply returns the number of hits expected in a given detector. The second returns a structure, which contains the number of expected hits, the number found and the likelihood contribution.

For the VELO a second interface (**IVeloExpectation**) is used. This allows the number of r and ϕ hits that are expected to be calculated separately. For the VELO account must be taken of the fact that stations can be missed because the particle is the product of a hyperon decay occurring after the start of the VELO acceptance. The interface below takes account of these requirements and in addition provides methods that allow to calculate how many stations the particle traverses before the first hit seen on the track. The interface is as follows:

```
class IVeloExpectation: virtual public IAlgTool{

public:

  /** Helper struct
  */
  typedef struct{
```

```
  unsigned int nR;
  unsigned int nPhi;
} Info;

/// Retrieve interface ID
static const InterfaceID& interfaceID() { return IID_IVeloExpectation ; }

/** Returns number of hits expected, from zFirst to endVelo
*  @param aTrack Reference to the Track to test
*  @return number of hits expected */
virtual int nExpected ( const LHCb::Track& aTrack ) const = 0;

/** Returns Info on hits expected, from zFirst to endVelo
*  @param aTrack Reference to the Track to test
*  @return Info */
virtual IVeloExpectation::Info
expectedInfo ( const LHCb::Track& aTrack ) const = 0;

/** Returns number of hits expected, from zStart to zStop    *
 *  @param aTrack Reference to the Track to test
 *  @param zStart --> start of scan range
 *  @param zStop --> end of scan range
 *  @return number of hits expected */
 virtual int nExpected ( const LHCb::Track& aTrack ,
                         const double zStart ,
                         const double zStop) const = 0;

/** Returns Info on hits expected, from zStart to zStop
*
*  @param aTrack Reference to the Track to test
*  @param zStart --> start of scan range
*  @param zStop --> end of scan range
*  @return Info */
virtual IVeloExpectation::Info
  expectedInfo (const LHCb::Track& aTrack ,
                const double zStart,
                const double zStop) const = 0;

/** Returns number of hits missed, from zBeamLine to firstHit
*  @param aTrack Reference to the Track to test
*  @return number of hits missed before first hit */
virtual int nMissed( const LHCb::Track& aTrack ) const = 0;

/** Returns number of hits missed, from z to firstHit
 *  @param aTrack Reference to the Track to test
 *  @param z --> start of scan range
 *  @return number of hits missed before first hit */
virtual int nMissed( const LHCb::Track& aTrack,
                     const double z ) const = 0;

 /** Returns true if track passses thro a given sensor
 *
 *  @param aTrack Reference to the Track to test
    @param sensorNum ---> sensor number
 *  @return true if inside sensor */
 virtual bool isInside(const LHCb::Track& aTrack,
                       const unsigned int sensorNum) const = 0;
```