LHCb note 2007-128

Classification of Ghost Tracks

M. Needham
CERN *

October 5, 2007

Abstract

A tool for classifying ghost tracks together with the results obtained with Brunel v31r8 is described.

*Present Addresss: Laboratoire de Physique des Hautes Energies, Ecole Polytechnique Fédérale de Lausanne

1 Introduction

The efficiency of the majority of the LHCb tracking algorithms is limited by the ghost rate in the detector that can be tolerated. This is particulary true for luminosities greater than the nominal LHCb running value of 2×10^{32} cm^{-2}s^{-1}. In this note a tool that classifies ghost tracks is presented. This tool allows the origin of ghost tracks to be better understood. Thus, it is hoped thus better strategies for identifying and removing ghosts can be developed.

This note is structured as follows. First, the proposed classification scheme and its implementation is discussed. This is followed by a discussion of the results found running the tool on Brunel v31r8.

2 Classification Scheme

Based on studies of the origin of ghost tracks the following categorization scheme is proposed. *Nota Bene,* the scheme proposed is not unique: some tracks can be placed in more than one of the categories. Therefore, the categories have been ranked according to how informative they are. Each track is checked against each category in turn until the first match is found. The categories are:

Spillover and noise: 70% of the **LHCbIDs** assigned to the track are not related to any **MCParticle** [1].

Decay in flight: 70% of the **LHCbIDs** assigned to the track are due to a kaon or pion and its daughter muon.

Conversion: 70% of the **LHCbIDs** assigned to the track are due to an electron pair produced by a photon conversion within the detector.

EM: 70% of the **LHCbIDs** assigned to the track are due to an electron.

Hadronic interaction: Two or more of the **MCParticle** that give **LHCbIDs** assigned to a track originate in hadronic interactions in the detector.

[1] Clusters from both spillover and noise have no associated **MCParticle**. In fact the noise rates simulated in the tracking detectors are low. Hence, all the tracks in the category are due to spillover.

Phi decay products: 70% of the **LHCbIDs** assigned to the track are due to a $K^{\pm}$ pair produced in a decay of a ϕ.

Ghost Parent: One or more of the ancestors tracks [1] used to construct the track is a ghost.

After this generic procedure has been run algorithm specific categorizations are considered:

Inconsistant parts Many of the LHCb tracking algorithms build tracks in independent projections before combining these to give space tracks. It can be that the **LHCbIDs** corresponding to each projection are consistant but that two projections are not related to the same **MCParticle**. In the case of the VELO tracking this corresponds to the hits in the r and ϕ projections being from different **MCParticles**. In the case of the T seeding this corresponds to the hits in the x and stereo layers being from different **MCParticles**. This category is also used to flag a ghost in the produced by the track matching and the forward tracking where the T and VELO seed tracks are not related to the same **MCParticle**.

Combinatoric In the VELO r tracking combinations of three co-linear r hits are considered valid track candidates. If the three hits assigned to the track are from unrelated **MCParticles** then this is categorized as combinatoric.

Finally, the tool can return two trivial cases: if no classification was possible or if the tool has been called for a real track.

3 Implementation

The categorization scheme described above has been implemented as a Gaudi tool with the interface given below:

```
1  static const InterfaceID
2    IID_ITrackGhostClassification( "ITrackGhostClassification", 0,0 );
3
4  class ITrackGhostClassification: virtual public IAlgTool {
5
6  public:
7
8     typedef std::vector<LHCb::LHCbID> LHCbIDs;
```

```
9       /// Retrieve interface ID
10      static const InterfaceID&
11        interfaceID() { return IID_ITrackGhostClassification; }
12
13      /**
14      *  Information on what a ghost track is....
15      *  @param LHCb::Track to link
16      *  @param LHCb::GhostInfo filled with ghost info
17      *  @return StatusCode
18      */
19      virtual void StatusCode info(const LHCb::Track& aTrack,
20                                   LHCb::GhostTrackInfo& tinfo) const= 0;
21
22      /**
23      *  Information on a list of LHCbIDs
24      *  @param LHCbIDs::const_iterator start
25      *  @param LHCbIDs::const_iterator stop
26      *  @param LHCb::GhostInfo filled with ghost info
27      *  @return StatusCode
28      */
29      virtual void StatusCode info(LHCbIDs::const_iterator& start,
30                                   LHCbIDs::const_iterator& stop,
31                                   LHCb::GhostTrackInfo& tinfo) const= 0;
32  };
```

The tool has two methods that can be called. The first takes a **LHCb::Track** as an input. The second takes iterators that define a range of LHCbIDs. This allows the tool to be used in cases — such as the pattern recognition where no **LHCb::Track** object has yet been built. In both cases the result of the categorization is filled into a **LHCb::GhostTrackInfo** object [2]. This contains both the result of the categorization procedure and also a map containing the raw information on which MCParticles were linked to the track.

Fig. 1 shows the class diagram for tools inheriting from this interface. A concrete base class (**GhostTrackClassificationBase**) provides the generic part of the categorization procedure together with member functions for manipulation of the raw **MCParticle** map. If only the generic categorization is required this base class can be called directly. On the otherhand if an algorithm specific classification is need a specialized implementation inheriting from the base class can be made. This has been provided for the current 2-D, Velo 3-D, T-seeding and long tracking algorithms.

The following code fragments illustrate how to use the tool:

```
1  #include ''MCInterfaces/IGhostTrackClassification.h''
2  #include ''GaudiKernel/toStream.h'' // turns enums to strings
3
4  // get the tool
5  IGhostTrackClassification* gTool =
6     tool<IGhostTrackClassification>(''LongGhostClassification'');
```

[2]The header file is located in the MCEvent package.

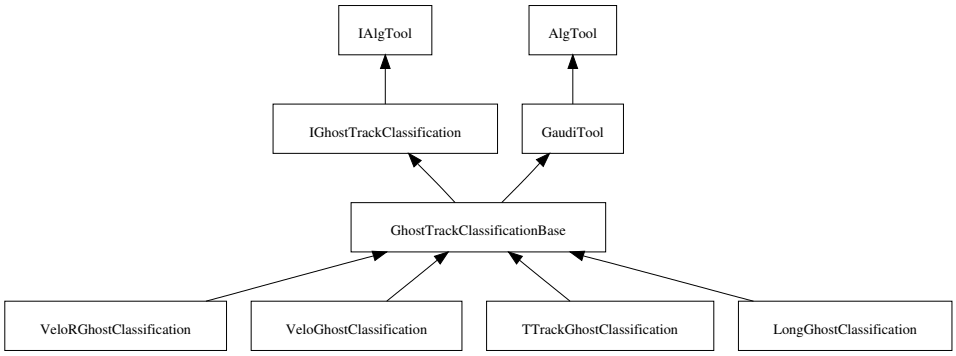

Figure 1: Class diagram for the ghost track classification tool.

```
7
8  // call it
9  LHCb::Track* aTrack;
10 LHCb::GhostTrackInfo tInfo;
11 StatusCode sc = gTool->info(aTrack, tInfo);
12
13 // get the classification
14 std::cout << ``Classified as ''
15           << Gaudi::Utils::toString(tinfo.classification()) << std::endl;
```

4 Results

he performance of the algorithm has been studied using 13000 $B_d \to J/\psi(\mu^+\mu^-)K_S(\pi^+\pi^-)$ events generated for the DC' 06 production and reconstructed with Brunel v31r8. Fig 2-7 show the results of running the algorithm for the Velo 2-D, Velo 3-D, Tsa seeding, Forward Tracking, Track Matching and Dowstream tracking. It can be seen that:

- For all algorithms there is a sizeable fraction of ghost tracks that are due to electromagnetic interactions of photons and electrons within the detector.
- For the Velo 2-D tracking it can be seen that the largest source of ghosts is tracks with three hits that originate from different sources (combinatoric).
- The only algorithm where the contribution from spillover is sizeable is the Tsa seeding. This was known from previous studies [2].

- In the case of the 'long' tracking algorithms the biggest source of ghosts is random matches of the T and Velo parts
- More work is needed to improve the classification scheme. In particular for the Velo 3-D and Tsa seeding the number of tracks which are not classified is high.

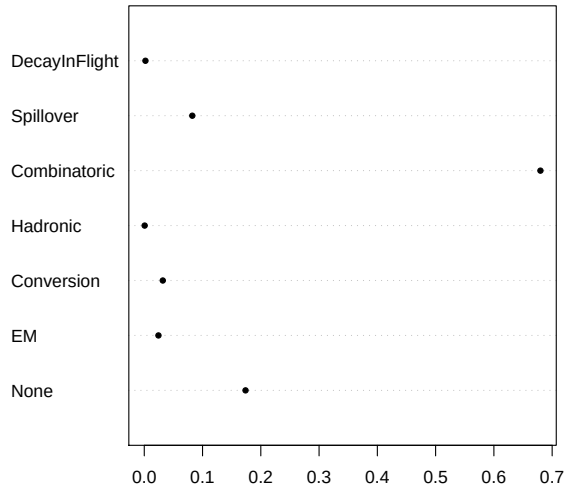

Figure 2: R dotchart [3] showing the composition of ghost tracks in % for the Velo 2-D tracking [4].

References

[1] J. Hernando Morata and E. Rodrigues. Tracking event model. LHCb-note 2007-007.

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

[3] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0.

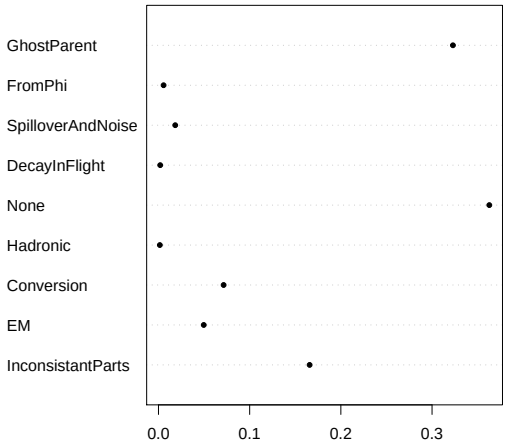

Figure 3: R dotchart [3] showing the composition of ghost tracks in % for the Velo 3-D tracking [4].

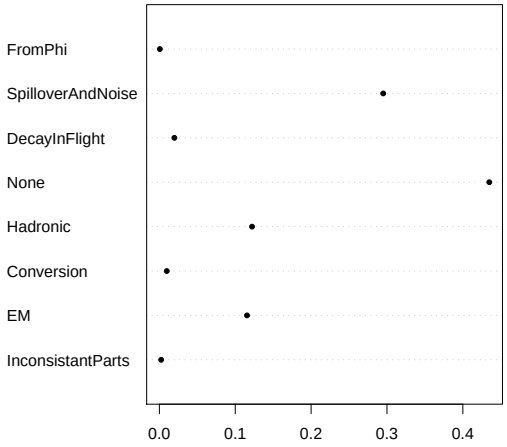

Figure 4: R dotchart [3] showing the composition of ghost tracks in % for the Tsa seeding [2].

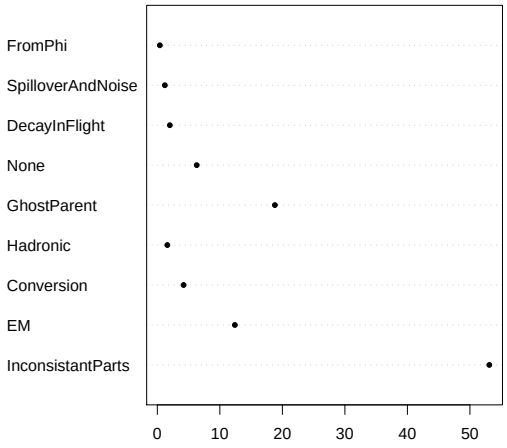

Figure 5: R dotchart [3] showing the composition of ghost tracks in % for the forward tracking [5].

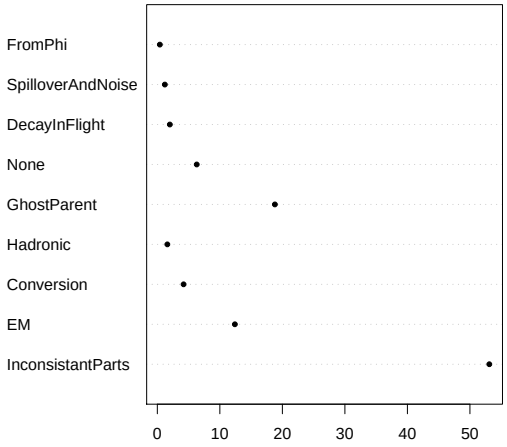

Figure 6: R dotchart [3] showing the composition of ghost tracks in % for the track matching [6].

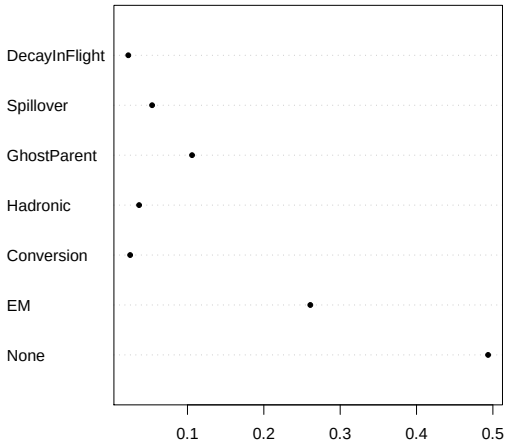

Figure 7: R dotchart [3] showing the composition of ghost tracks in % for the downstream tracking [7].

[4] D. Hutchcroft *et al.* VELO Pattern Recognition. LHCb-note 2007-013.

[5] O. Callot and S. Menzemer. Performance of the forward tracking. LHCb-note 2007-015.

[6] M. Needham. Performance of the Track Matching. LHCb-note 2007-129.

[7] O. Callot. Downstream pattern recognition. LHCb-Note 2007-026.