PatSeeding — A Standalone Track Reconstruction Algorithm

Public Note

Issue:	1
Revision:	0
Reference:	LHCb-2008-042
Created:	August 6, 2008
Last modified:	August 6, 2008

Prepared by:	Olivier Callot[a], Manuel Schiller[b]
	[a]Laboratoire de l'Accélérateur Linéaire, Orsay, France
	[b]Physikalisches Institut, Heidelberg, Germany

Abstract

A standalone pattern recognition algorithm for the LHCb T stations is presented. Its performance on $B_d \longrightarrow J/\psi K_s$ and minimum bias Monte Carlo events is evaluated, and tunings for reconstruction without magnetic field, with misalignments or for reconstruction of cosmics are shown. Furthermore, this notes describes changes to the algorithm which make it suitable for trigger applications, especially for Level 0 confirmation (the performance of these modifications is evaluated in an upcoming LHCb note).

Document Status Sheet

1. Document Title: PatSeeding — A Standalone Track Reconstruction Algorithm			
2. Document Reference Number: LHCb-2008-042			
3. Issue	**4. Revision**	**5. Date**	**6. Reason for change**
1	Final	August 6, 2008	Initial release.

Contents

List of Figures

List of Tables

1 Introduction

This note describes PatSeeding, one of the two competing standalone pattern recognition algorithms for the main tracking stations (consisting of Inner and Outer Tracker) in the LHCb experiment[a]. The code was originally written by one of the authors (Olivier Callot) and is now maintained by the other author.

Apart from evaluating the performance of the algorithm on signal and minimum bias Monte Carlo events, this note shows how the algorithm can be tuned to reconstruct tracks under suboptimal circumstances like those expected for the very first data taken by the experiment.

In this note, names of classes in the software are typeset in a font without serifs (e.g. PatSeeding), and job options are indicated with a typewriter font (e.g. `UsePatForward`). Unless explicitly noted otherwise, job options belong the the algorithm or tool described in the section in question.

To keep some consistency with the source code, the three stations in the main tracking stations will be enumerated 0, 1 and 2 for stations T1, T2 and T3 (in that order).

[a] For a description of the other algorithm, Tsa seeding by Roger Forty and Matthew Needham, we refer the interested reader to [1] and [2].

2 Description of the algorithm

The algorithm performs the pattern recognition using a tool, PatSeedingTool, which can be reused by other algorithms (e.g. in the trigger). Therefore, this section effectively describes PatSeedingTool. A list of options with their default values is given in the Appendix.

The algorithm consists of five distinct steps:

- hit preparation
- track search per detector region
- track search for tracks migrating from IT to OT
- track search per Outer Tracker region for tracks at large $|y|$
- validation of low quality tracks left over from the per region search

Each layer of the main tracking stations is divided into six different regions (cf. Fig. 1). Because there are few tracks migrating between regions, the second step is executed per detector region, thus reducing combinatorics.

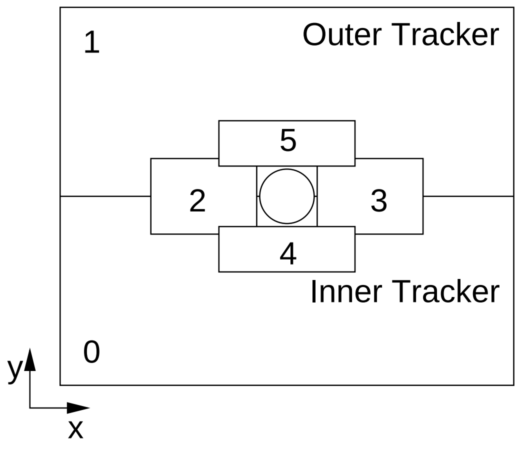


Figure 1: *All layers in the main tracking stations are divided into six regions (OT: 0-1, IT: 2-4). Not to scale.*

Once all tracks staying within a single region have been found, tracks are formed from the remaining hits. This step allows for tracks which leave the Inner Tracker and continues to search for hits in the Outer Tracker.

The remaining steps recover tracks that are difficult to find in previous stages.

2.1 Hit preparation

PatSeedingTool obtains a range of hits from the T station hit manager (Tf::TStationHitManager<PatForwardHit>). The hit manager has already marked potential clusters (i.e. hits in neighbouring straws) using a flag. The hits returned by the hit manager have also been sorted by increasing x coordinate (at $y = 0$).

Each hit has a flag indicating if it is used. Per default, this flag is initialised to indicate an unused hit, unless

- the hit has an unphysical drift time (in case of OT hits, a cut is placed on the drift radius obtained from the drift time such that `DriftRadiusRange[0]` $< r <$ `DriftRadiusRange[1]`)

- if `UseForward` is true, hits used by a previous run of the PatForward[b] algorithm are marked used
- if `ReusePatSeeding` is false, hits used by a previous run of PatSeedingTool are marked used

2.2 Track model and fit method

The track model and the fit are basic building blocks of the pattern recognition strategy. They are explained ahead of the actual pattern recognition algorithm to have them available when needed.

2.2.1 *Field shape*

Without magnetic field in the main tracking stations, tracks would just be straight lines. The field in the main tracker has some global properties that influence the shape of the track segments that the pattern recognition will try to find:

- The field is decreasing with z, which means that the curvature between the first and the second station of the main tracker is higher than the one between the second and the third.
- The stations have six x layers. The distance between two x layers in the same station is already too big such that the local slope cannot be ignored. A track is thus measured not in three points but in six.
- The ratio of the field in the two gaps between stations (i.e. T1-T2 and T2-T3) is reasonably constant within the surface of the stations.
- The field varies with y, and in the uppermost (and lowermost) part of the T stations, it is essentially zero.

2.2.2 *Track model*

The algorithm employs a straight line description of the trajectory of a particle in yz projection. For xz projection, a cubic model with three track parameters a, b, c is used:

$$x(z) = c(z - \texttt{zReference})^2(1 + \texttt{dRatio}(z - \texttt{zReference})) + b(z - \texttt{zReference}) + a$$

$$y(z) = b_y z + a_y$$

`dRatio` describes the correlation between the quadratic and cubic terms and is determined from Monte Carlo studies. The shift by `zReference` makes the fit more numerically stable (`zReference` is in the middle of the T stations, by default). We will sometimes also refer to the cubic track model in xz projection as "parabola" because the latter is easier to remember and the cubic admixture is small.

Under the assumption of a track coming from the primary vertex and due to the magnetic field in the T stations being roughly proportional to the integral of the field along the trajectory of a particle through the magnet, one can derive a linear relation between the x coordinate of a line joining two x hits in stations 0 and 2 at $z = 0$, $x_{intercept}$, and the curvature. This is used in several places of the algorithm. It is parametrised by the x deviation from the straight line from above in station 1, Δx (cf. Fig. 2):

$$\Delta x = \texttt{InitialArrow} \cdot x_{intercept}$$

This relation holds rather well in the centre of the T stations, but not near the top or the bottom where the field inside the T stations is essentially zero. Therefore, the algorithm has an extra pass over the hits with `InitialArrow` put to zero to recover tracks at high $|y|$ (see 2.5).

Different field maps require a different tuning of the parameters governing the track model. The sets of parameters for the field map used in the DC06 Monte Carlo productions as well as the sets of parameters for the measured field map can be found in the appendix.

[b]PatForward is a tracking algorithm for the main tracker that starts from seeds in the vertex detector. (See [6].)

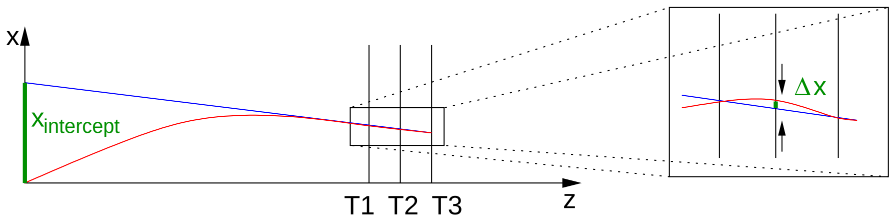


Figure 2: *Assuming a track comes from the origin, one can estimate the position of a hit in T2 relative to a straight line joining hits in T1 and T3. This displacement Δx (best visible in the magnified portion on the right) is proportional to the intercept of the straight line with the x axis, $x_{intercept}$. Not to scale.*

2.2.3 *Fit method*

The fit is a standard weighted least-squares fit; the resolution of the hits is provided by the Tf-framework (see [3]). Throughout the fitting procedure, the coordinates of the hits are updated according to the changes in track parameters to account for shifts in x and z coordinates due to stereo angles and the 3.6 mrad tilt of the main tracker coordinate frame with respect to the LHCb one.

In a first step, initial parameters are obtained by just fitting first the xz and then yz projection separately, using only x hits in the first case and only stereo hits in the second.

Then, both fits are iterated up to ten times each to account for correlations and resolve ambiguities in the Outer Tracker (see below for details). Again, the fit in xz projection comes first. The algorithm minimises

$$\chi^2 = \sum_{\text{hits}\, i} \frac{1}{\sigma_i^2} \left(\frac{x_i - x_{track}(z_i)}{\cos(\alpha)} \pm r_{drift} \right)^2$$

where (x_i, z_i) is the x coordinate of the i-th hit at the $y(z_i)$ predicted from the track model[c], the factor $\cos^2(\alpha)$ accounts for the slope of a track in xz projection[d], σ_i is the hit position uncertainty, and r_{drift} is the drift radius for Outer Tracker hits or 0 for the Inner Tracker. The sign is chosen such that the smaller contribution to χ^2 is taken.

In yz projection, the expression is similar, but the sum goes only over hits in stereo layers, and the distance (i.e. $x_i - x_{track}(z_i)$) is divided by an additional factor of $-\tan\theta_i$ to convert the x distance into a distance in y (θ_i is the stereo angle of the i-th layer).

If the changes in parameters fall below the following values, the iterations are stopped early:

$$\begin{array}{ll} |\Delta a| < 5 \cdot 10^{-3} & |\Delta a_y| < 5 \cdot 10^{-2} \\ |\Delta b| < 5 \cdot 10^{-6} & |\Delta b_y| < 5 \cdot 10^{-5} \\ |\Delta c| < 5 \cdot 10^{-9} & \end{array}$$

During the initial fit, left-right ambiguities in the OT are resolved separately for x and stereo layers in the following way:

- in each station, search for the hit with the largest drift time
- if all three have a drift radius smaller than 0.1 mm, the resolution of ambiguities is done later by iterating the fit
- otherwise, all 8 combinations to resolve the ambiguity of the 3 hits with the largest drift distance are tried, and the solution which gives lowest χ^2 in the projection in question is chosen

[c]The x coordinates of hits on a track are shifted by $y \cdot dx/dy$ as soon as an estimate of y at the z position of the hit is available.
[d] In the Outer Tracker, the nominal hit position (x_i, z_i) is the wire position, while the true hit is somewhere on a drift circle around that wire. Dividing the distance in x between track and wire at nominal z_i by $\cos(\alpha) = \frac{1}{\sqrt{1+b^2}}$ converts it into the distance of closest approach, which is what is needed for the fit. In the inner tracker, this factor is unity.

For the fit in the stereo layers, in fact only the two largest of the three selected hits are considered (only two fit parameters need to be determined), bringing the number of possible combinations down to 4. Resolving ambiguities like that ensures that the fit needs fewer iterations to converge.

During iterations, the ambiguities are resolved towards the track described by the current set of track parameters.

After, the iterations, a track χ^2 is recalculated (using the formula from above for xz projection). If the χ^2 contribution of the worst fitting hit is above a threshold for outlier removal, it is removed from the fit, and the fit is restarted from scratch. If the number of layers[e] with hits falls below a minimum, the fit returns, indicating an error. Each track is assigned a track χ^2 which is the χ^2 just calculated, divided by the number of degrees of freedom.

Outlier cuts and minimum number of hits are set to different values in different places of the algorithm; the values used will be described there. Also, the fit in yz projection can be skipped when no stereo hits are available at a certain stage of the algorithm.

2.3 Track search per detector region

The track search consists of two phases: First, track candidates are searched in xz projection. Then, stereo hits are collected, and a final track selection is made. The resulting track is fitted using a weighted least squares method to estimate a reference state for a subsequent fit with the Kalman-based fitter.

2.3.1 *Track search in xz projection*

From each hit in one of the x layers of station 0 and each suitable x hit in station 2, a straight line is constructed. Suitable means that the combination of hits must satisfy both of the following constraints:

- The hit in station 2 must be inside a window formed by lines joining the first hit with $(x = \pm maximpact, z = 0)$, where

$$maximpact = \frac{1}{\texttt{MinMomentum}} \left(\frac{1\mathrm{mm}^2}{\texttt{MomentumScale} \times \texttt{InitialArrow}} + 210\mathrm{MeV}\,2000\mathrm{mm} \right)$$

  The first term translates a maximum curvature (due to a minimal track momentum) and the hypothesis of the track coming from the primary vertex into an impact parameter estimate. The second term accounts for an additional displacement at the origin due to a potential K_S decay. Both contribution are proportional to the inverse minimum momentum.

- The lines joining the hit in station 0 with the points $(x = \pm\texttt{xMagTol}, \texttt{zMagnet})$ form an additional search window in station 2. ("centre of magnet compatibility").

In case of Outer Tracker hits, the hit positions are taken to be the wire positions. If the hit manager indicates that two hits might form a cluster, a weighted mean of the hit positions is used. The weight of an Outer Tracker hit is $w = \frac{1}{\sqrt{r^2+\sigma^2}}$ where r is the drift distance and σ is the detector resolution. The idea behind this is that hits with small drift radii contain more information about the true particle position than those with larger drift time. They should therefore give the greater contribution. As the measured drift radii can also occasionally be negative (which would obviously spoil the weights we want to calculate), the modulus of the drift radius is used. This is done by squaring and taking the square root afterwards. The reason for introducing the addition of the squared hit resolution under the square root is that one wants to avoid divisions by zero in case of one of the drift radii being zero. This procedure also has the nice property that it degenerates into the arithmetic mean if drift information is not used or not present (in this case, the drift radii for all hits read zero). Note that a similar procedure was also used in [4].

[e]For the Outer Tracker, a layer consists of both monolayers of straws inside a module.

A hit in station 1 is selected whose distance to the track is less than `TolCollectIT` (or `TolCollectOT`, respectively)[f]. From these three points in (x, z)-space, a parabola is formed.

For the Outer Tracker, only combinations of hits in either the first or the last x layer of each station are considered at this stage, because layers are sufficiently separated in z that a track passing trough the inefficient region[g] near $y = 0$ are unlikely to go through this region in all layers of a station.

For the Inner Tracker, the separation in z of layers in the same station is much smaller, so all combinations of x layers are considered to reduce the risk of losing the track.

Then, hits from all other x layers are added, provided that their distance to the parabola is less than `TolCollectIT` or `TolCollectOT`, respectively.

From these hits, a track candidate is constructed, if

- the resulting candidate has hits in all but `MaxMisses` x layers
- the track has hits in at least `MinXPlanes` x layers after the weighted least squares fit in xz projection (inside the fit, a cut on outliers is placed at a per-hit contribution to the track χ^2 of `MaxChi2HitIT` or `MaxChi2HitOT`, respectively)
- the fit converged
- the fit χ^2 is below `MaxTrackChi2`
- in case of IT-only tracks, at least 3 hits must have enough charge found in the IT to be above the IT readout high charge threshold
- in case of an OT-only low multiplicity track (i.e. with less than 8 hits), the following additional requirements have to be met:
  - the outlier cut is tightened to a per-hit χ^2 contribution of `MaxFinalChi2`
  - the track χ^2 is below `MaxTrackChi2LowMult`
  - none of the potential OT clusters (flagged as such by the hit manager) may have a hit removed by the outlier removal

The resulting track candidate is stored for further inspection. In case the x position in the middle of a track candidate stored previously is within `CloneMaxXDist` of the track to be added, only the candidate with higher quality is kept. Two tracks only compete if the shorter one shares at least a fraction of `CommonXFraction` of its hits with the longer one.

Track quality is defined as

$$Q = \texttt{QualityWeights}[0] \times n_{Hits} + \texttt{QualityWeights}[1] \times \chi^2$$

The track with lower quality loses, the other is kept. `QualityWeights` is `{ 1.0, -0.2 }` per default.

2.3.2 *Track search in stereo layers*

For each track candidate in xz projection, the algorithm searches for hits in the stereo layers of the corresponding region. The algorithm uses a Hough transform to find the track in yz projection among all compatible stereo hits.

Hits are considered further if they satisfy the following requirements:

[f]Due to the tilt of the main tracker coordinate frame with respect to the LHCb one, it is necessary to slightly widen narrow search windows. This widening depends on the distance from the last known point in z, the most extreme slopes t_y expected in the detector region, and the tilt dz/dy. See the code for details.

[g]Both Outer and Inner Tracker have an inefficient area near $y = 0$. In case of the Inner Tracker, this is caused by a dead area in between two silicon ladders. For the Outer Tracker, modules are electrically divided in the middle to permit separate readout of top and bottom half. The resulting dead area of a monolayer in an Outer Tracker module has been staggered, resulting merely in decreased efficiency for the whole Outer Tracker module near $y = 0$.

- $$y_{min}\tan(\theta) - \texttt{TolCollect} < x_{hit} - x_{track} < y_{max}\tan(\theta) + \texttt{TolCollect}$$

  In this expression, θ is the stereo angle, y_{min} and y_{max} describe the extension of the sensitive area of the stereo layer, x_{hit} is the coordinate of the hit in the stereo layer at $y = 0$, and x_{track} is the x coordinate of the track candidate at the z of the stereo layer. Depending on the subdetector considered, you have to substitute `TolCollectIT` or `TolCollectOT` for `TolCollect` in the expression above.

- If the candidate in xz projection is entirely in the Outer Tracker and has less than 7 hits, the algorithm assumes that the track goes through the detector at small $|y|$, where only one of the two monolayers is efficient. Therefore, the algorithm looks for stereo hits in both the upper and lower half of the Outer Tracker, and it restricts the sensitive region of both halves to the range $-5\text{cm} < y < 5\text{cm}$.

As this is just a pre-selection of possible stereo hits, more work is required. The algorithm therefore projects the y coordinate of all hits to `zForYMatch` (assuming a straight line in y from the origin through the hit). This plane forms the Hough space in which the algorithm searches for a cluster of hits. The spread of such a cluster must be below `MaxRangeIT` (or `MaxRangeOT` in case of the Outer Tracker). A cluster of stereo hits is combined with the track candidate in xz to form a track, if

- the cluster of stereo hits together with the track candidate would form a track in which at least `MinTotalPlanes` layers have hits
- if several such clusters exist, the algorithm considers only those in which the number of stereo layers with hits is maximal
- if there is still more than one such cluster, the one with the smallest spread in the projection plane is selected

The algorithm continues to add stereo hits at both sides of the cluster which are less than $\frac{\texttt{TolCollectIT}}{|\tan(\theta)|}$ (or $\frac{\texttt{TolCollectOT}}{|\tan(\theta)|}$, respectively) away from the cluster in projection plane. This step is skipped, if the cluster already contains hits from six stereo layers, or if it overlaps with the interval $[-5\text{cm}, 5\text{cm}]$ in the projection plane.

The resulting combination of hits must satisfy the following conditions to make it to the algorithm's output:

- After the initial fit in yz projection, $|y_{corr}(0)| < \texttt{maxYAtOrigin}$ where

  $$y_{corr}(0) = a_y \pm \frac{b_y^2\, c^2}{\texttt{yCorrection}}$$

  The correction is due to the magnetic field slightly changing the slope in y (positive sign for $b_y < 0$, negative otherwise). Outer Tracker tracks failing this criterion ($\texttt{maxYAtOrigin} \leq |y_{corr}(0)| \leq \texttt{maxYAtOriginLowQual}$) are saved for further potential use during the last stage of the algorithm.

- for Outer Tracker tracks, demand at least 2 stereo hits per station (unless $|y| < 5\text{cm}$ in the projection plane from above)
- for Outer Tracker tracks with fewer than `OTNHitsLowThresh` hits, demand that for each hit marked as part of a potential cluster, the other hit in the potential cluster is also part of the track
- a refit of the track is made; the outlier cut is set to `MaxFinalChi2`, and we demand `MinTotal-Planes` planes
- if the fit converged, the criterion for Outer Tracker tracks with less than `OTNHitsLowThresh` hits from above still holds and the track χ^2 is below `MaxFinalTrackChi2`, the track is stored for a final selection pass

The tracks stored are then sorted by decreasing number of hits. For tracks with the same number of hits, lower χ^2 values come first. Starting with the first of the sorted tracks, tracks are put into the output destination if less than `MaxUsedFractPerRegion` of their hits are used. The track is discarded otherwise. Hits are tagged used and a state estimate is produced (with the FastMomentumEstimate tool) while adding tracks to the output container. This process continues until all tracks sorted above have either been discarded or added to the output container.

2.4 Track search for tracks migrating from Inner to Outer Tracker

In this stage, tracks overlapping regions are constructed from the remaining unused hits. The strategy starts in the Inner Tracker. In each station of each IT region, we search for four hits satisfying the following constraints:

- two x hits are in different x layers and satisfy the centre of magnet constraint from the last section
- a stereo hit in the first stereo layer that is compatible with the x part of the track found (i.e. within $\pm$`TolCollectIT` of the x position predicted from the straight line between the two x hits)
- a second stereo hit in the second stereo layer compatible with the previous three hits (the x coordinate is predicted using the straight line segment in x and line joining the first stereo hit with the origin; a hit must be within ± 1.5`TolCollectIT` to be accepted

From these four hits, parameters for the full track model are estimated, assuming that it came from the primary vertex. In xz projection, one has thus three points, enough to derive the parameters for the track model. In yz projection, an average y is calculated, and the slope is also estimated under the primary vertex assumption.

This produces a track candidate to which neighbouring hits in the same station and region are added, provided that they are within `TolCollectIT` of the hits already found.

Unused Outer Tracker hits are added to the candidates, if

- they are in a different station than the four Inner Tracker hits
- they are in the upper half of the Outer tracker for stereo hits in the Inner Tracker with $y > 0$ and vice-versa
- they are within `TolExtrapolate` of the position predicted from the estimated track parameters

For every station added to the track with hits in three or more layers, there is a refit. An outlier cut is placed at `MaxChi2HitOT`. A track is abandoned if it lacks hits in more than one layer among the layers in stations contributing hits to the track.

If the resulting candidate still does not have hits in all three stations, the same procedure is applied to the Inner Tracker, looking for hits there.

If the fit converges and the track candidate has hits in all three stations, the track is put into the output container (after the same selection procedure that is used after the per-region track search, with the maximal fraction of used hits on a track set to `MaxUsedFractITOT` this time).

For tracks passing through the Inner Tracker at small $|y|$, the track is also accepted if it has hits in only two of the three stations to avoid losing it if it passes through the inefficient area in the Inner Tracker (as explained during the per-region search description).

This stage is actually repeated a second time, after hits found by the first first iteration have been tagged used. For the second iteration, all Inner and Outer Tracker regions are considered once a track segment in the Inner Tracker has been found. This enables the algorithm to also find tracks that migrate between different regions in the Inner Tracker itself.

2.5 Track search in the Outer Tracker at high $|y|$

As explained in 2.2.1, tracks at high $|y|$ in the Outer Tracker are much less curved than in the rest of the detector. Therefore, a second iteration of the track search per region is performed on the unused hits in the Outer Tracker. For this iteration, `InitialArrow` is temporarily set to zero to make the algorithm expect zero curvature for all tracks found during this stage, and `MaxUsedFractPerRegion` is set to 0.05, so that only tracks are found that share very few hits among each other.

2.6 Track search among low quality candidates

This step builds upon work done during the per region track search by using low quality Outer Tracker tracks if they have enough unused hits. Essentially, these tracks are processed in the same way as those that were not flagged low quality tracks during the per region track search, but there are a few modifications. For brevity, only the modifications are listed below:

- The fraction of hits on a track used by previous stages must be below `MaxUsedFractLowQual`. Up to 3 used hits are removed before we start counting. The algorithm removes the used hits it encounters first. This removing procedure is done to avoid losing tracks which have a used hit in very few layers.
- After removing hits, the initial straight line fit in yz projection is redone, and we require $|y(z = 0)| <$ `MaxYAtOriginLowQual`.
- When the refit is performed, demand at least `MinTotalPlanes - 1` layers with hits, with the outlier cut set to `MaxFinalChi2` and the maximum track χ^2 smaller than `MaxFinalTrackChi2`.

Surviving tracks enter a competition like at the very end of the per region track searching step; the fraction of used hits per track must be below `MaxUsedFractLowQual` for a track to be accepted.

3 Performance

The performance of the algorithm has been evaluated on 10,000 simulated events of a $B_d \longrightarrow J/\psi K_S$ sample from the DC06 production[h]. The simulated events correspond to a luminosity of $2{\cdot}10^{32}\mathrm{cm}^{-2}\mathrm{s}^{-1}$.

The following conditions have to be fulfilled by a (Monte Carlo) particle to be included in the efficiency denominator:

- it must be charged
- it must neither be an electron nor a positron
- it must be reconstructible in the main tracking stations (i.e. it must have at least one x and one stereo hit in each station)
- for long tracks, reconstructibility in the vertex detector (Velo) is demanded

Using these criteria, an efficiency of 70.3% is observed for all tracks. This figure rises to 94.9% for long tracks and 96.0% for long tracks with momenta over 5 GeV. The fraction of tracks which can not be attributed to a physical particle in the detector (the "ghost fraction") is 8.2% (track-averaged). Calculating this figure per event and averaging the results gives 6.1% ghost tracks. The fraction of duplicate tracks is about 0.1% of all tracks (these are also called "clone tracks", i.e. several tracks match the same Monte Carlo particle, one track contributes to the efficiency calculation, the others are called "clones"). Table 1 summarises the situation.

efficiency			ghost fraction		clone fraction
all	long	long, $p > 5$ GeV	track-av.	event-av.	
70.3%	94.9%	96.0%	8.2%	6.1%	0.1%

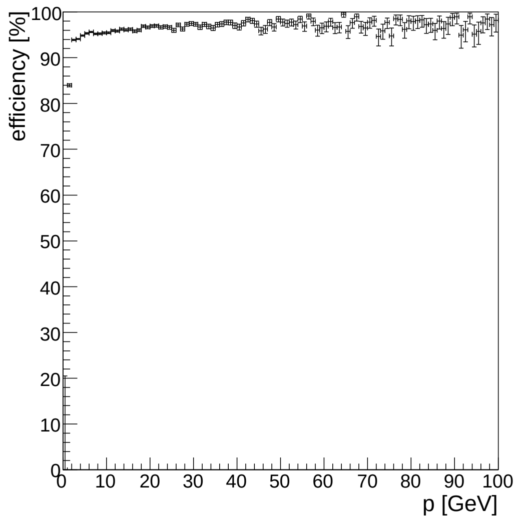
Table 1: *Efficiency, ghost and clone fraction of the algorithm evaluated using 10,000 $B_d \longrightarrow J/\psi K_s$ events.*

Fig. 3 shows the behaviour of the efficiency versus p, p_T and η for charged long tracks. The dependency of efficiency and ghost fraction versus the number of visible interactions is also shown. As can be expected, tracks with high momentum (or transverse momentum) are seen to be easier to reconstruct than low momentum ones; efficiency drops to about 84% for momenta between 1 and 2 GeV. The efficiency of the algorithm decreases and the ghost fraction increases with rising number of visible interactions which is also what one would expect.

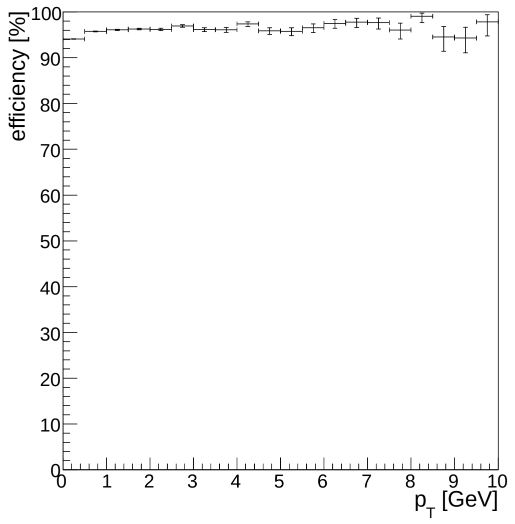
Looking specifically at decay products of B mesons under the constraints given above, the algorithm achieves an efficiency of 96.0%. Figure 4 shows the corresponding distributions.

It is interesting to also look at the performance of the algorithm for tracks from strange particle decays (K^0_S, Λ) which can not be found in the vertex detector (Velo), because these tracks have to be reconstructed in the T stations or are not found at all. In the sample studied, this group of tracks accounts for 39.3% of all reconstructible tracks from strange particle decays. PatSeeding manages to reconstruct 78.9% of these tracks, subject to the definitions laid out in the beginning of the chapter. Figure 5 shows the distribution of efficiency versus p and p_T.

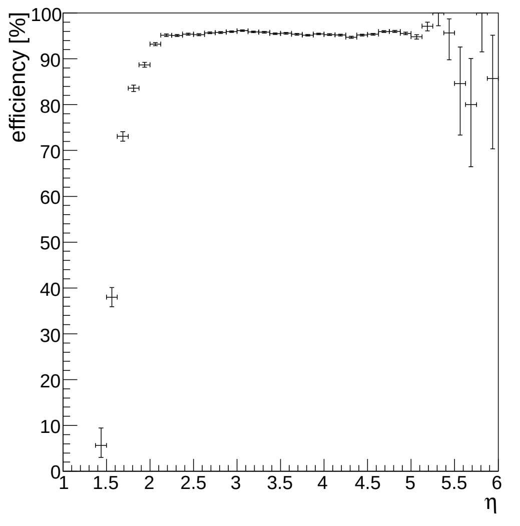
[h]Boole v12r10, Gauss v25r8 and v25r9, Brunel v32r8, with modifications to PatSeeding available publically after Brunel v32r8



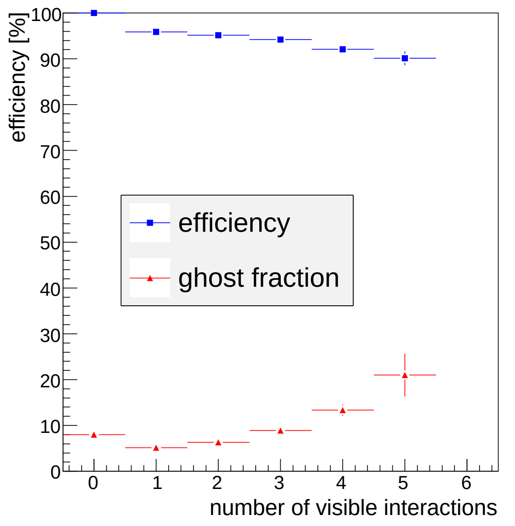
(a)



(b)

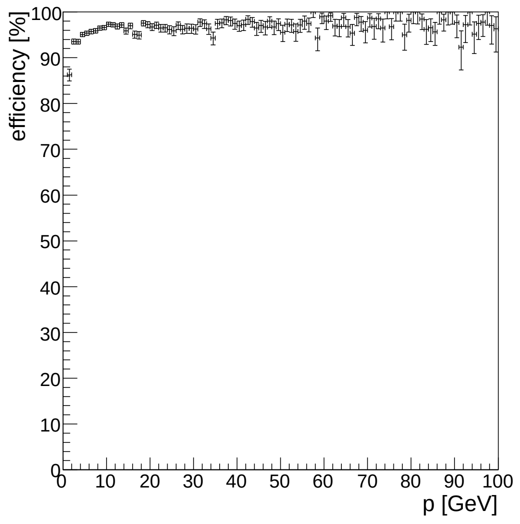


(c)

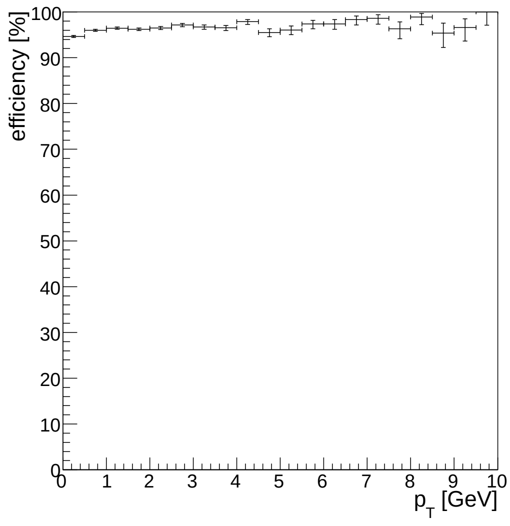


(d)

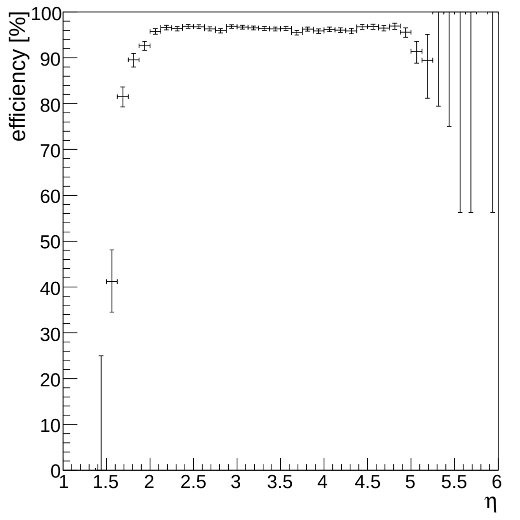
Figure 3: *Efficiency versus (a) p, (b) p_T, (c) η, and (d) efficiency and ghost fraction versus number of visible interactions.*



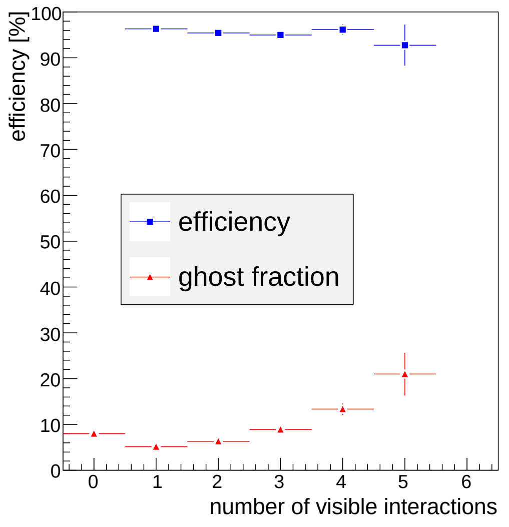
(a)



(b)

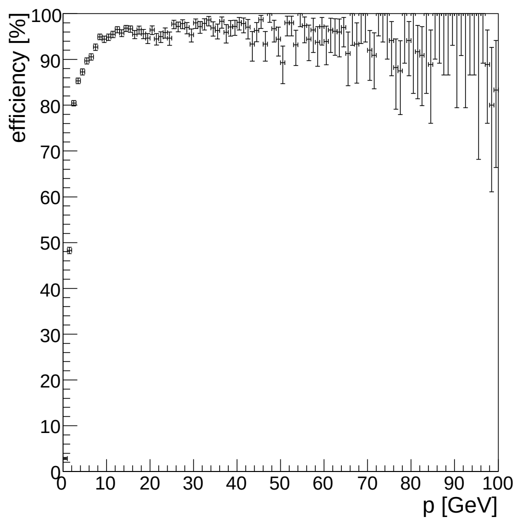


(c)

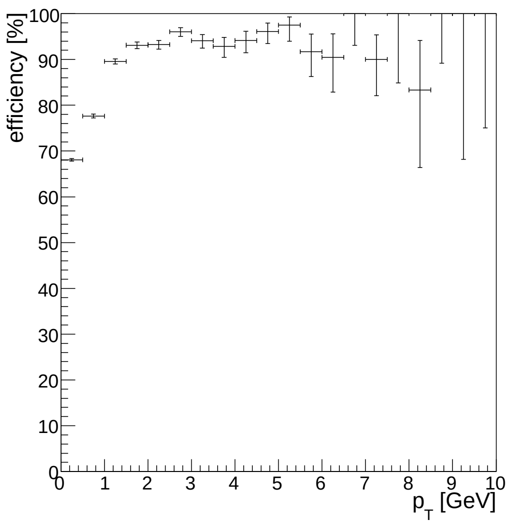


(d)

Figure 4: *Efficiency for B daughters versus (a) p, (b) p_T, (c) η, and (d) efficiency and ghost fraction versus number of visible interactions.*



(a)



(b)

Figure 5: *Efficiency for tracks from strange particle decays that can not be found in the vertex detector versus (a) p and (b) p_T.*

3.1 Timing

On average, the algorithm took 34.5 ms per event on a machine with an AMD Opteron processor running at 2.312 GHz. Fig. 6 shows the time required per event as a function of the number of hits in the main tracker.

The execution time of the algorithm scales roughly cubically with the number of hits in the main tracking station and can be parametrised as:

$$t = 5.7 \cdot 10^{-3} n_{Hits} - 2.1 \cdot 10^{-6} n_{Hits}^2 + 4.9 \cdot 10^{-10} n_{Hits}^3 \text{ [ms]}$$

If the behaviour for high T station hit multiplicity events is considered a problem, two possibilities to reduce the worst case timing spring to mind, both of which would be extremely easy to implement:

- The algorithm could perform hit cleaning as is presently done in a different reconstruction algorithm for the main tracker (see [1]).
- The algorithm could also refuse to reconstruct events with more than a certain number of hits in the main tracker (see also [1]).

Considering the small average execution time and the fact that relatively clean and thus low multiplicity events are required for the precision physics that LHCb wants to do, it may well turn out that the long execution time for these relatively few hot event is not such a serious limitation.

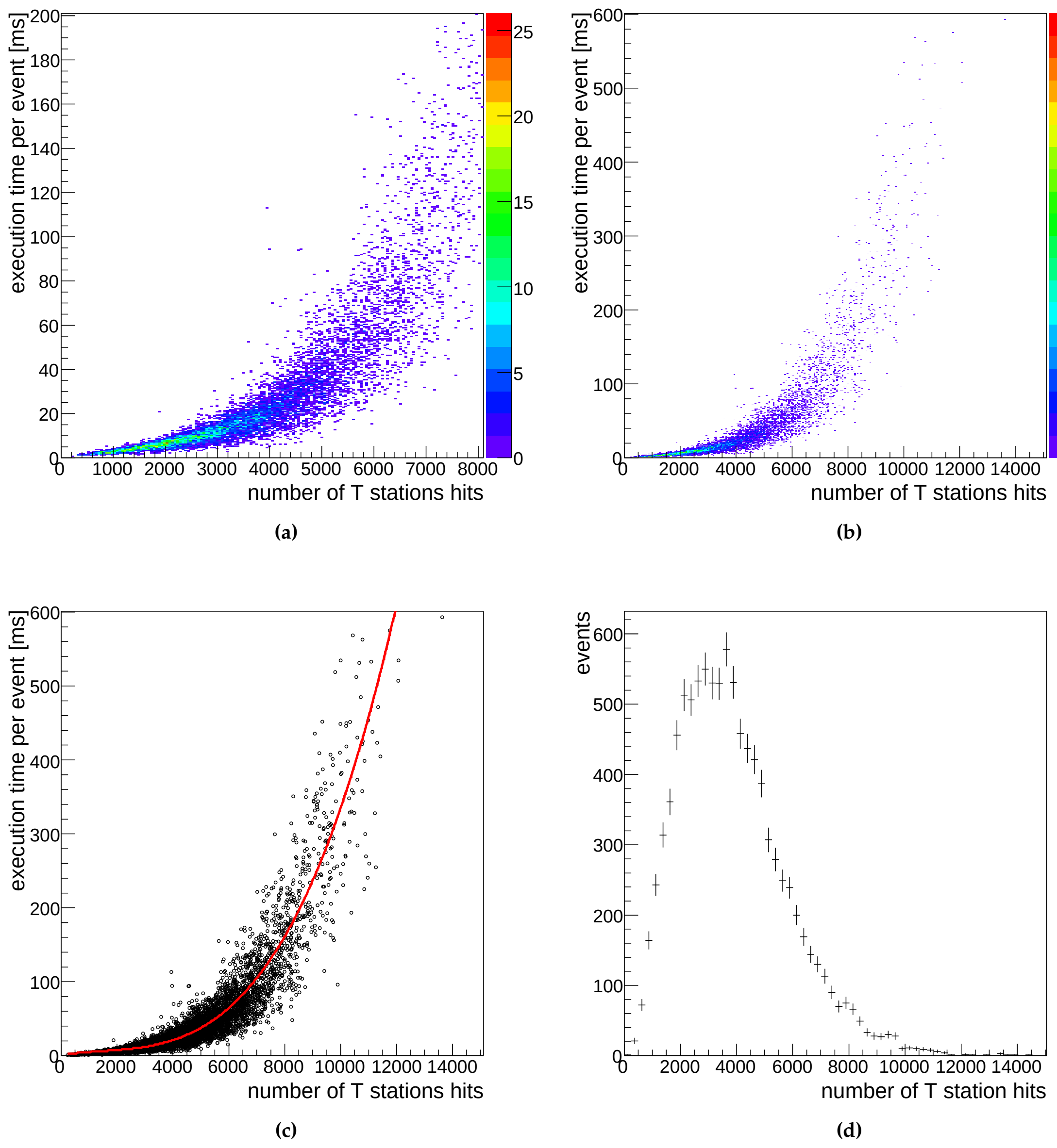


Figure 6: *Time required for reconstruction per event versus number of hits in the LHCb main tracker, (a) detailed view up to 8,000 hits in the main tracker, (b) for all 10,000 events, (c) parametrisation (see text), (d) distribution of number of main tracker hits per signal event.*

3.2 Working in tandem with PatForward

The algorithm can work reusing information about tracks found by PatForward (see [6]), a pattern recognition algorithm that starts from seeds in the vertex detector to reconstruct tracks in the T stations. Obviously, it is not necessary to find tracks already found by PatForward a second time. To this end, PatSeeding can be configured to ignore hits used for PatForward tracks (cf. 2.1). The average execution time per event drops to 24.0 ms in that case.

To measure combined efficiencies, it is possible to extract the main tracker segments of the tracks found by PatForward, optionally killing clones among these segments (which arise because PatForward is tuned to most efficiently match main tracker hits to vertex detector seeds, so it sometimes splits the main tracker part of a singe track to match two seeds in the vertex detector). Running these track extraction and clone killing stages adds 0.3 ms per event of execution time on average, bringing the total to 24.3 ms per event. In the latter case, the execution time can be parametrised as:

$$t = 1.5 \cdot 10^{-3} n_{Hits} + 1.9 \cdot 10^{-10} n_{Hits}^3 \text{ [ms]}$$

`UseForwardTracks` can be set to `true` to enable this track extraction. The clone merging procedure is activated by setting `ForwardCloneMergeSeg` to `true`. When activated, tracks with track parameters differing by less than `ForwardCloneMaxXDist`, `ForwardCloneMaxYDist` and `Forward-CloneMaxTXDist` in x, y, and slope in x direction, t_x, respectively, are considered clones (track parameters are evaluated at $z =$ `zReference`).

For tracks sharing more than a fraction of `ForwardCloneMaxShared` of their hits, the track with more hits is kept, otherwise, the hits of both tracks are merged to produce a single track.

The combined efficiency to find tracks in the main tracker is shown in table 2. Without the clone killing among main tracker segments inherited from PatForward, the clone fraction is on the order of 5%. One should also note that ghost tracks found by PatForward can cause some inefficiency in PatSeeding because the hits are unavailable for PatSeeding when running in this mode.

efficiency			ghost fraction		clone fraction
all	long	long, $p > 5$ GeV	track-av.	event-av.	
70.8%	97.4%	98.6%	7.0%	5.5%	0.4%

Table 2: *Efficiency, ghost and clone fraction of the algorithm evaluated using 10,000 $B_d \longrightarrow J/\psi K_s$ events, reusing tracks already found by PatForward.*

4 Tunings and modification for special applications

4.1 Tracking without magnetic field

Tracking with and without magnetic field is not substantially different from the point of view of this algorithm, apart from the fact that one expects straight tracks instead of curved ones. To account for this fact, the set of options in table 3 has been prepared.

option	value
`xMagTol`	`400 mm`
`zMagnet`	`0 mm`
`FieldOff`	`true`
`MinMomentum`	`50000 MeV`

Table 3: *Options for running with magnet off data*

Effectively, it demands straight tracks by turning the centre of magnet compatibility cut into a pointing constraint at $z = 0$ and enforces relatively straight tracks by artificially introducing a cut on minimal momentum. It also applies an additional cut on the track curvature c:

$$\frac{\texttt{TolCollect}}{dz^2} > |c|$$

where dz is half the distance between the first and the last hit in z and `TolCollect` has to be substituted by either `TolCollectIT` or `TolCollectOT`, depending on the subdetector.

4.1.1 Performance

The performance of this tuning has been evaluated on a minimum bias sample with 10,000 events prepared for alignment purposes by Steven Blusk (see [5]). The beam energy was 7 TeV, and the events were generated without spillover or magnetic field.

The results are summarised in Table 4 and Fig. 7. Efficiency for long tracks is lower (89.1%) than what was seen with magnetic field, and significantly lower when considering all reconstructible tracks (35.6%). This has two reasons: First, the magnetic field usually deflects tracks with very low momentum away from the main tracker. These tracks are more difficult to reconstruct because of the increased importance of multiple scattering. Second, the total T station multiplicity increases as more and more low momentum tracks make it through the magnet. These factors both add to the difficulties encountered by the algorithm.

efficiency			ghost fraction		clone fraction
all	long	long, $p > 5$ GeV	track-av.	event-av.	
35.6%	89.1%	90.1%	16.0%	5.8%	0.1%

Table 4: *Efficiency, ghost and clone fraction of the algorithm evaluated using 10,000 minimum bias events without magnetic field.*

It should also be noted that the tuning is not particularly optimised. It was designed to quickly give acceptable performance for long tracks for alignment and calibration studies which can be done before the magnet is first ramped up.

The average reconstruction time per event was about 9.5 ms on this particular sample of events.

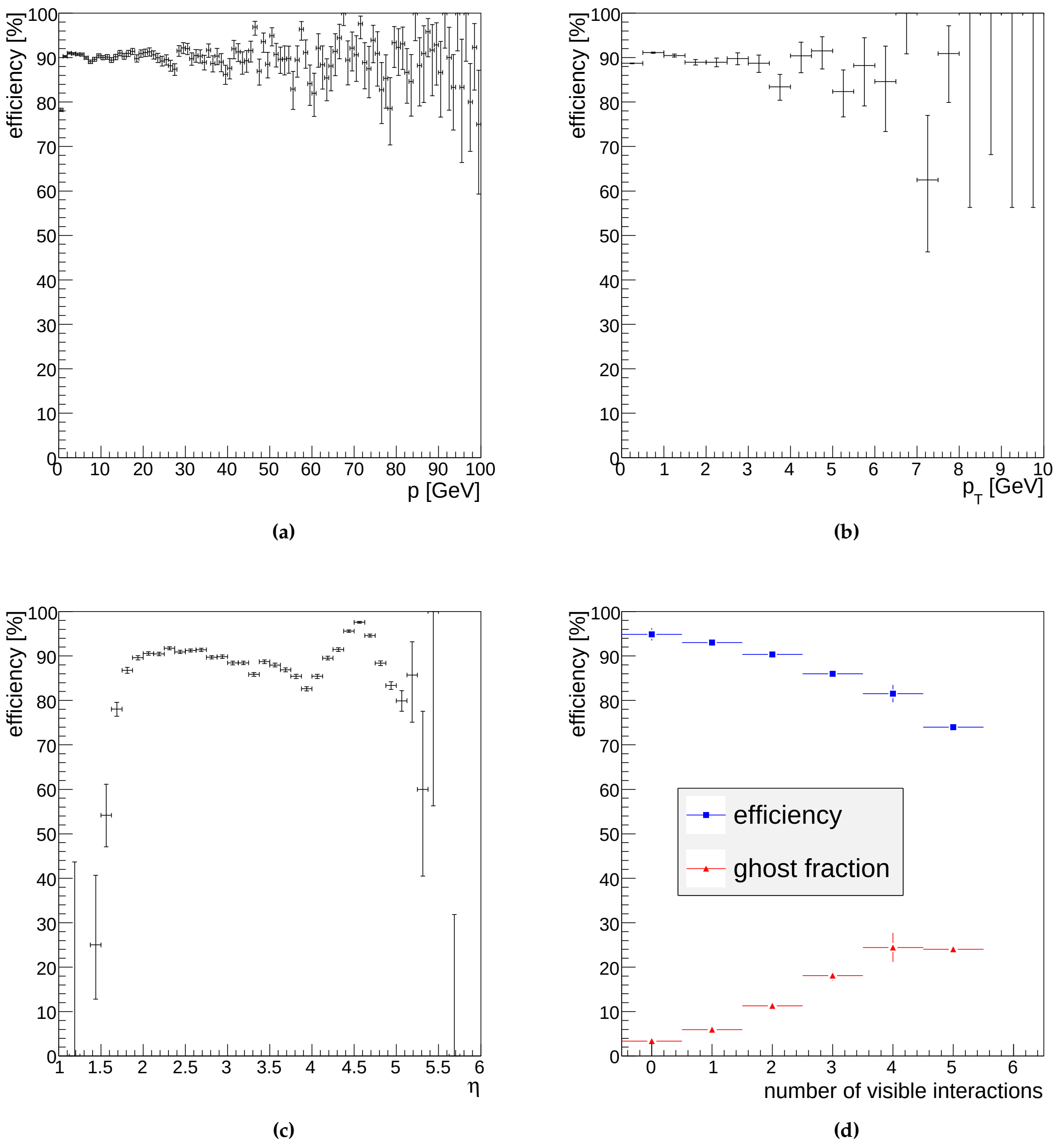


Figure 7: *Efficiency versus (a) p, (b) p_T, (c) η, and (d) efficiency and ghost fraction versus number of visible interactions (all for minimum bias events without magnetic field).*

4.2 Tracking with a misaligned detector

In this subsection, the effect of a misaligned detector on the pattern recognition is studied. The issue is discussed with producing tracks for detector alignment in mind.

When reconstructing with a misaligned detector, one must take care of two things: First, one needs to widen search windows, χ^2 and outlier cuts, and second, it is usually necessary to suppress ghost tracks which would be killed by the tighter cuts applied in case of perfect alignment.

For the first category, a set of options will be given in 4.2.2. The second category, killing ghosts, is tackled by requiring tracks to be isolated in the detector. This comes of course with a substantial loss in efficiency, but this is fine because for alignment, it is more important to have a clean track sample.

4.2.1 *Isolated tracks*

A track is called isolated in a layer if there are no hits within `ITIsolation` (or `OTIsolation`, respectively) to the left and to the right of the hit in this layer. If the hit manager has marked a hit to potentially belong to a cluster, the other potential hit in the cluster is ignored for the check above.

A track is said to be isolated if:

- During the xz search per region, the three points forming the initial parabola must be isolated, and non-isolated hits are skipped when picking up further x hits around the parabola.
- While collecting stereo hits per region, a hit must either be isolated to be collected, or, if it is not, it must be the first violating hit in a strip of $\Delta y = 15\mathrm{cm}$ (strips start at $y = 0$). This segmentation of stereo layers is done to avoid losing hits which are clearly separated in y, even if this is not apparent from their x coordinate. (Clearly, it would be better to demand the hit to be the only isolation-violating hit instead of the first, but the code would be more complex and slower.)
- When searching tracks spanning several regions, all hits must be isolated in their respective layers and regions.

To turn on the code enforcing these isolation criteria, the option `EnforceIsolation` must be set to true.

4.2.2 *Performance*

For this study, the same sample as in 4.1.1 was used. More specifically, we studied four misalignment scenarios in which both Inner and Outer tracker layers were shifted in x direction and rotated around all axes by a random amount, according to a Gaussian distribution. The width of these distributions (i.e. the parameter commonly called σ) is shown in Table 5. To avoid very large misalignments, values differing from zero by more than 2.5σ were rejected, and another random number was drawn.

All four scenarios were tested using 10,000 events, the two sets of options used can be found in Table 6 (scenarios 2 and 4 with their large rotation angles require cuts to be wider). Please note that the set options is not particularly well tuned.

scenario	σ(shifts OT)	σ(shifts IT)	σ(rotations OT and IT)
1	1.0 mm	0.3 mm	1 mrad
2	1.0 mm	0.3 mm	10 mrad
3	4.0 mm	1.0 mm	1 mrad
4	4.0 mm	1.0 mm	10 mrad

Table 5: *Misalignment scenarios: Inner and Outer Tracker layers were shifted in x direction and rotated around all three axes according to a Gaussian distribution with mean σ.*

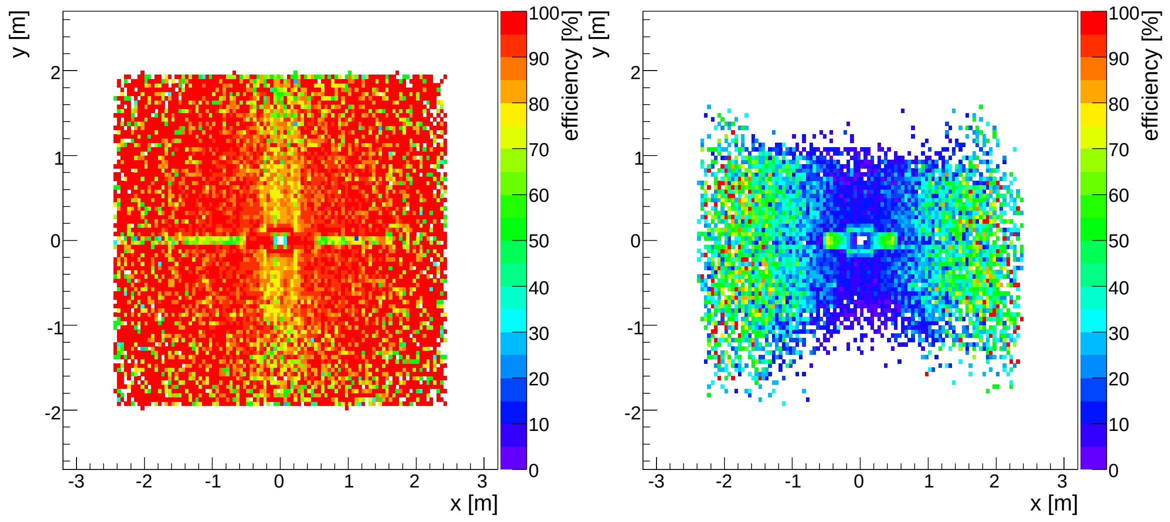
The aim of this study was to achieve very low a ghost fraction for alignment purposes, efficiency is considered less important in such a context. Table 7 shows the results for the four scenarios.

option	value (scenarios 1, 3)	value (scenarios 2, 4)
`CurveTol`	`12.0 mm`	`12.0 mm`
`TolCollectOT`	`5.5 mm`	`5.5 mm`
`TolCollectIT`	`1.5 mm`	`3.0 mm`
`MaxRangeOT`	`200. mm`	`200. mm`
`MaxChi2HitOT`	`400.`	`3600.`
`MaxChi2HitIT`	`400.`	`1800.`
`MaxFinalChi2`	`400.`	`3600.`
`MaxTrackChi2`	`400.`	`1800.`
`MaxTrackChi2LowMult`	`300.`	`1800.`
`MaxFinalTrackChi2`	`300.`	`1800.`
`MinTotalPlanes`	`11`	`11`
`CloneMaxXDist`	`20.0 mm`	`20.0 mm`
`QualityWeights`	`1.0, 0.0`	`1.0, 0.0`
`EnforceIsolation`	`true`	`true`
`MaxYAtOriginLowQual`	`600. mm`	`600. mm`

Table 6: *Options used to reconstruct a misaligned detector*

		ghost fraction		
scenario	efficiency	track av.	event av.	clones
1	26.1%	1.8%	1.2%	3 in 67392
2	24.1%	2.8%	2.3%	3 in 49419
3	15.9%	2.0%	1.5%	0 in 38648
4	20.5%	2.3%	1.8%	2 in 44472

Table 7: *Performance for different misalignment scenarios (efficiency for long tracks with momenta above 5 GeV, number of clones among all reconstructed tracks).*



(a) **(b)**

Figure 8: *Efficiency versus track position at $z = 7500$ mm. (a) is without misalignments, (b) is for scenario 4. The inefficiency at central x in the Outer Tracker is caused by high track multiplicity because the absence of the magnetic field means a larger number of low momentum particles can reach the main tracker. In (b), the inefficiency in the outer regions of the Outer Tracker is caused by rotations around the centre of the layers, leading to larger displacements of hits in these regions.*

There has been some concern in the collaboration that the isolation cuts used in this setup might cause the algorithm to fail completely in the most occupied regions of the detector, thus making alignment of these regions using this algorithm effectively impossible. Therefore, we investigated the efficiency of the algorithm as function of the track position at $z = 7500\mathrm{mm}$ (cf Fig. 8) for scenario 4 and for perfect alignment, also using the sample from 4.1.1.

In case of perfect alignment, the effect of the higher multiplicity in the main tracker (due to the absence of the magnetic field as filter for low momentum particles) can be seen in the central region of the Outer Tracker. In the misaligned case, the efficiency is clearly lower than under ideal conditions, but the algorithm manages to find tracks also in the dense regions of the detector. The plot for the misaligned case is not completely symmetric because Inner Tracker boxes can move independently, so that the actual misalignments applied are not symmetric themselves.

It must also be said that for the scenarios with large rotations (10 mrad), the algorithm has trouble reconstructing tracks in the Outer Tracker at large $|x|$ and particularly $|y|$. This is due to the large lever arm caused by the spatial extension of the Outer Tracker. However, rotations of 10 mrad, in particular around the z axis, seem to be quite big so that we should know rotations to a better accuracy from survey measurements. (10 mrad over the length of half an Outer Tracker module mean displacements of 2.5 cm which should definitely be measurable.)

4.3 Modification to reconstruct cosmics

To reconstruct cosmics taken during the commissioning effort, three minor modifications to the algorithm are required:

- For the Outer Tracker, neither the timing of the Outer Tracker wrt. the bunch clock nor the relation between drift radius and drift time are known exactly. Therefore, the framework supplying the hits is instructed to return zero drift distance in the Outer Tracker (with the resolution set to $\frac{5\mathrm{mm}}{\sqrt{12}}$)[i].
- Cosmics do not point back to the primary vertex which is used extensively to find the pattern in the stereo layers during the per region track search. For cosmics, this is replaced by a combinatorial approach which counts the number of hits inside a window around a line connecting two of these hits. The combinations of two hits that gives most hits inside this window is taken.
- Because cosmics traverse the detector at much steeper slopes than tracks coming from the primary vertex, one can not expect to hits in more that one station of the Inner Tracker. Therefore, only the Outer Tracker is used to supply the three hits for the initial parabola. When collecting hits in a window around this parabola, all regions are searched for hits, if the track traverses them (including Inner Tracker regions).

Using these modifications, it is possible to reconstruct cosmic tracks. An example is shown in Figure 9.

The job options used to reconstruct this event are shown in table 8. As there was no magnetic field, the options from 4.1.1 were used as well; in case of conflicting options, those given here should take precedence.

At the time of this writing, reconstructing cosmics is more a proof of principle than a finished product. PatSeeding is not optimised for this use case yet, and reliable figures on performance and efficiency of the algorithm are not yet available.

4.4 Modification for application in the trigger

The standard PatSeedingTool settings are not fast enough to be run in trigger applications. Therefore, one uses information from elsewhere to restrict the algorithm to look at a small region of interest, see

[i]Note that if these tracks are to be fitted, the fitter has to be told as well that drift times are to be ignored. The standard reconstruction sequence does a so-called pre-fit on tracks in the main tracker, ignoring drift times. Thus, options to switch off drift times in the fitter are best looked up there.

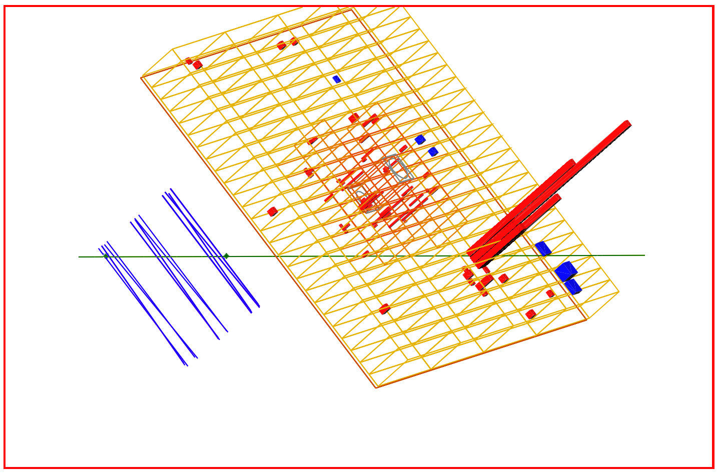


Figure 9: *Cosmic track reconstructed from run 24941, event 138. Data taken July 1st, 2008, triggered on energy deposit in the calorimeters. In this run, one half of the Outer Tracker (C side) was read out together with the calorimeters. Reconstructed track in dark green, Outer Tracker straws on the track in blue, clusters in electromagnetic and hadronic calorimeters in red and blue, respectively.*

option	value
`PatSeeding.PatSeedingTool.Cosmics`	`true`
`PatSeeding.PatSeedingTool.MinXPlanes`	`4`
`PatSeeding.PatSeedingTool.MinTotalPlanes`	`8`
`PatSeeding.PatSeedingTool.OTNHitsLowThresh`	`9`
`PatSeeding.PatSeedingTool.MaxMisses`	`2`
`PatSeeding.PatSeedingTool.MaxYAtOrigin`	`400000`
`PatSeeding.PatSeedingTool.MaxYAtOriginLowQual`	`800000`
`PatSeeding.PatSeedingTool.xMagTol`	`400000`
`PatSeeding.PatSeedingTool.MinMomentum`	`1e-4`
`PatSeeding.PatSeedingTool.QualityWeights`	`1.0, 0.0`
`ToolSvc.OTHitCreator.NoDriftTimes`	`true`

Table 8: *Options used to reconstruct cosmics. The last entry puts to zero the Outer Tracker drift radii for all pattern recognition algorithms, as explained in the text.*

below. It is also possible to ignore hits which have been tagged used by previous runs of PatSeedingTool or other algorithms, see 2.1 for details.

This subsection only discusses how the algorithm deals with such a region of interest, its performance and time requirements are best studied in the context of the trigger framework. Consequently, the performance discussion is left to a different LHCb note (currently in preparation).

4.4.1 Selecting a region of interest

One can specify a region of interest to PatSeedingTool by giving a state and a covariance matrix at some reference z_{ref}. Track curvature is neglected . Let $\vec{S}(z_{ref})$ be the state and $C(z_{ref})$ be the covariance matrix at z_{ref}:

$$\vec{S}(z_{ref}) = \begin{pmatrix} x \\ y \\ t_x \\ t_y \end{pmatrix} (z_{ref})$$

$$C(z_{ref}) = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xt_x} & \sigma_{xt_y} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yt_x} & \sigma_{yt_y} \\ \sigma_{xt_x} & \sigma_{yt_x} & \sigma_{t_x}^2 & \sigma_{t_xt_y} \\ \sigma_{xt_y} & \sigma_{yt_y} & \sigma_{t_xt_y} & \sigma_{t_y}^2 \end{pmatrix} (z_{ref})$$

Then, centre and extension of the region of interest in the main tracker is then calculated using the following formulae:

$$\vec{S}(z) = P(z - z_ref)\vec{S}(z_{ref})$$

$$C(z) = P(z - z_{ref})C(z_{ref})P^T(z - z_{ref})$$

where $P(dz)$ is the matrix that transports the state over a distance dz, i.e.

$$P(dz) = \begin{pmatrix} 1 & 0 & dz & 0 \\ 0 & 1 & 0 & dz \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The extension of the region of interest around the centre is taken to be given by the square root of the elements on the diagonal of the covariance matrix at z.

Hits are only picked up in the spatial region given by above formulae, and at every step, the slopes resulting from the combination of hits are checked for consistence with the limits specified.

One can argue that restricting the region of interest to a window of one sigma is too tight a cut, but one can always scale the covariance matrix before passing it to PatSeedingTool to obtain a wider cut.

Based on the region of interest supplied, the algorithm will also skip stages of the reconstruction where it should not find tracks. For example, if the region of interest is entirely in the Outer Tracker, the stage looking for tracks migrating from Inner to Outer Tracker can be skipped.

5 Summary

PatSeeding, a standalone reconstruction algorithm for the LHCb main tracker, was presented. It achieves 94.9% reconstruction efficiency for long tracks and 96.0% for long tracks above 5 GeV on a sample of 10,000 simulated $B_d \longrightarrow J/\psi K_s$ events from the DC06 production. The fraction of wrongly reconstructed tracks (”ghost tracks”) is 6.1% (event-averaged). Moreover, special tunings for the algorithm were shown, for example to reconstruct without magnetic field, with a misaligned detector, and how to reconstruct cosmics. Due to the possibility to restrict the algorithm to a small region of interest, the algorithm can also be used in the software trigger.

6 References

[1] Roger Forty and Matthew Needham: Standalone Track Reconstruction in the T-stations, CERN-LHCb-2007-022, March 2007

[2] Roger Forty and Matthew Needham: Updated Performance of the T-Seeding, CERN-LHCb-2007-023, March 2007

[3] Christopher Jones, Kurt Rinnert, Stephanie Hansmann-Menzemer, Wouter Hulsbergen: Data handling tools for tracking-interfaces & implementation, LHCb tracking workshop, 30-August 2007

[4] I. Abt, D. Emeliyanov, I. Gorbounov, I. Kisel: Cellular automaton and Kalman filter based track search in the HERA-B pattern tracker, Nucl. Instr. and Meth., A490, p. 546-558, 2002

[5] LHCb Twiki: General Misalignment Samples, `https://twiki.cern.ch/twiki/bin/view/LHCb/AlignmentSamples`

[6] Olivier Callot and Stephanie Hansmann-Menzemer, The Forward Tracking Algorithm and Performance Studies, LHCb-2007-015. CERN-LHCb-2007-015, May 2007

A PatSeedingTool options and default values

option name	default value
`reusePatSeeding`	`true`
`UseForward`	`false`
`InputTracksName`	`LHCb::TrackLocation::Forward`
`DriftRadiusRange`	`-0.6 mm to 2.8 mm`
`dRatio`	`-3.2265e-3`
`InitialArrow`	`4.25307e-9`
`MomentumScale`	`44.1416`
`zReference`	`StateParameters::ZMidT`
`MinMomentum`	`500 MeV`
`CurveTol`	`5 mm`
`zMagnet`	`5383.17 mm`
`xMagTol`	`2000 mm`
`TolCollectOT`	`3 mm`
`TolCollectIT`	`0.3 mm`
`MinXPlanes`	`5`
`CloneMaxXDist`	`10 mm`
`CommonXFraction`	`0.7`
`QualityWeights`	`1.0, -0.2`
`zForYMatch`	`9000 mm`
`MaxRangeOT`	`150 mm`
`MaxRangeIT`	`50 mm`
`yCorrection`	`4.73385e15`
`MaxYAtOrigin`	`400 mm`
`MaxYAtOriginLowQual`	`1500 mm`
`TolExtrapolate`	`4 mm`
`MaxChi2HitOT`	`30`
`MaxChi2HitIT`	`10`
`MaxTrackChi2`	`20`
`MaxFinalChi2`	`15`
`MaxFinalTrackChi2`	`10`
`MaxTrackChi2LowMult`	`5`
`MinTotalPlanes`	`10`
`MaxMisses`	`1`
`OTNHitsLowThresh`	`17`
`MaxUsedFractPerRegion`	`0.30`
`MaxUsedFractITOT`	`0.15`
`MaxUsedFractLowQual`	`0.05`
`StateErrorX2`	`4`
`StateErrorY2`	`400`
`StateErrorTX2`	`6e-5`
`StateErrorTY2`	`1e-4`
`FastMomentumToolName`	`"FastMomentumEstimate"`
`ZOutput`	`StateParameters::ZBegT, StateParameters::ZMidT, StateParameters::ZEndT`
`UseForwardTracks`	`false`
`ForwardCloneMergeSeg`	`true`
`ForwardCloneMaxShared`	`0.3`
`ForwardCloneMaxXDist`	`10 mm`
`ForwardCloneMaxYDist`	`50 mm`
`ForwardCloneMaxTXDist`	`0.005`
`MeasureTime`	`false`
`FieldOff`	`false`
`EnforceIsolation`	`false`
`ITIsolation`	`15 mm`
`OTIsolation`	`20 mm`
`Cosmics`	`false`

B Tunings for different field maps

option name	value for DC06 field map	value for new field map
`InitialArrow`	`4.21826e-09`	`4.25307e-09`
`MomentumScale`	`40.3751`	`44.1416`
`zMagnet`	`5372.1 mm`	`5383.17 mm`
`dRatio`	`-3.81831e-04`	`-3.2265e-04`
`yCorrection`	`8.6746e-15`	`4.73385e-15`

The field map reffered to as DC06 field map can be found under

`$FIELDMAPROOT/cdf/field047.cdf`,

the new field map based on measured data can be found under

`$FIELDMAPROOT/cdf/field048.cN.cdf`

starting from v4r8 of the `FieldMap` package (`N` stands for 1, 2, 3, 4 and enumerates the four quadrants in the xy-plane).

C Acknowledgements

The authors would like to thank Johan Blouw who provided the databases for misaligned detector geometries. They would further like to acknowledge useful discussions with Matthew Needham who pointed out the issue with the insensitive area in the Inner Tracker mentioned in 2.3.1, and they would like thank Thomas Ruf for his help with the event visualisation software.