Pattern Recognition in the TRT for the ATLAS B-Physics Trigger

**C. Hinkelbein, A. Kugel, R. Männer,
M. Müller, M. Sessler, H. Simmler, H. Singpiel**
University of Mannheim, Germany

J. Baines
RAL, Chilton, Didcot, United Kingdom

R. Bock
CERN, Switzerland

M. Smizanska
University of Lancaster, United Kingdom

September 21, 1999

Abstract

The current B-physics trigger strategy in LVL2 starts with a scan of the full volume of the TRT to reconstruct all tracks with $p_T > 0.5$ GeV. The detector volume to be analysed is 85 times larger than a typical RoI, and the p_T range for a track search down to 0.5 GeV is 12 times larger than for a track search down to 6 GeV (RoI-guided). At low luminosity ($10^{33} cm^{-2} s^{-1}$), the full scan will be performed as part of the B-physics trigger with a frequency of 9 kHz [1]. This gives an additional factor of three in required processing power in comparison with the RoI-guided TRT feature extraction algorithm, which is executed with a frequency of 3 kHz. The TRT straw occupancy at low luminosity is lower by a factor of six in comparison with high luminosity.

Taking into account all these factors, the full scan at low luminosity will require 100 - 1000 times more computing power than the RoI-guided scan

at design luminosity. It is the most challenging of all LVL2 algorithms in terms of computing power and bandwidth requirements. A very fast and therefore simple algorithm is thus essential, independent of the hardware realisation.

This paper presents a TRT track reconstruction algorithm which is based on a Hough Transform using a look-up table (LUT). The pattern recognition is ideally suited for an FPGA implementation, whereas the track fit is more suited for implementation on general-purpose processors. The use of a general-purpose processor with FPGA co-processor allows an implementation which best matches the characteristics of the algorithmic parts to the strengths of both hardware components. In this case the execution time for the entire process, pattern recognition plus fit, is reduced by a factor of 20. All stages of the algorithm are implemented in C++. In addition the pattern recognition steps, apart from the fit, are partially implemented in VHDL (standardised Hardware Description Language) for FPGAs (Field Programmable Gate Arrays). For the algorithm development and quality studies, the C++ version was used. The FPGA implementation [2] was compared with the C++ version. Identical behaviour and an improvement in speed was demonstrated.

Contents

1 Introduction

The purpose of this paper is to present an algorithm which can be used for a fast track reconstruction in the whole TRT, and to show the results obtained with that algorithm. Section 2 explains the strategy of the B-physics trigger at low luminosity and the resulting requirements for track reconstruction in the TRT. To meet these requirements, an algorithm has been developed. The detector geometry defines natural boundaries between detector parts, which are treated independently for the pattern recognition[1]. These are two barrel halves and two end-cap parts for the TRT. Tracks are reconstructed separately in the different sub-detector parts. Tracks that cross from one part to another must be combined in a subsequent step. The algorithms for the barrel and the end-caps are described separately in sections 4 and 5. Section 6 is devoted to the barrel-to-end-cap transition region. An object-oriented design for the implementation in C++ is shown in section 7. The implementation for the FPGA co-processor is presented in section 8. The geometry of the TRT detector is briefly described in section 3 and the data samples used are described in section 9.

The reference algorithm xKalman is described briefly in section 2. The values of the execution times obtained are presented in section 10. Finally, the performance of the algorithm is presented in section 11 for single tracks and in section 12 for B-physics events.

2 B-Physics Trigger Strategy

The LVL2 B-physics trigger starts from events containing at least one muon candidate with transverse momentum $p_T > 6\,\mathrm{GeV}$ and within $|\eta| < 2.4$. The LVL1 muon trigger is an area of continuing development, the current prediction for the LVL1 trigger rate is 23 kHz. The first LVL2 action is to confirm the muon by a recalculation of the muon p_T (including the Inner Detector) which allows the application of a tighter threshold cut. The frequency for the surviving events is 9 kHz. They are assumed in this note to be pre-dominantly B-events. B-physics triggers are based on identifying charged products of the B-meson decay. For this task, no locality information (Region of Interest) of the LVL1 trigger is available to guide the processing. This forces the B-physics trigger to process the complete data volume of the TRT - an enormous demand in terms of computing power.

Separate B-physics trigger selections are applied to specific decay channels. This leads to the requirement of high reconstruction efficiency, since B-physics

[1]The term pattern recognition is used to describe the identification of track candidates, but not to include the fit

triggers rely on the identification of two or more particles. Different p_T thresholds are applied in the different selections. The most stringent requirement for the TRT algorithm arises from the need to reconstruct electrons with initial $p_T \geq 1\,\mathrm{GeV}$. For these events a scan of the whole TRT volume is required in order to reconstruct all tracks with $p_T^{gen} \geq 1\,\mathrm{GeV}$[2]. In order to have high efficiency for electrons with an initial $p_T \geq 1\,\mathrm{GeV}$, a p_T cut of 0.5 GeV is used, which allows for energy loss due to bremsstrahlung. On average, 90 particles per event satisfy a 0.5 GeV p_T cut at low luminosity. The electron identification capability of the TRT which makes use of transition radiation [3] is used to select all electron candidates with $p_T^{rec} > 0.5$ GeV and all other track candidates with $p_T^{rec} > 1$ GeV. The parameters of all track candidates are then passed to the SCT and pixels Feature extraction (Fex) to refine the track reconstruction parameters. Both the $1/p_T$ resolution and the ϕ resolution influence the computing requirements, since they determine size of the RoIs for track searches to be performed in the SCT/pixel system.

The final decision of the LVL2 trigger is based on a trigger menu utilising all the information from the reconstructed tracks. The following parameters contribute to the trigger decision and to the overall computing requirements:

- Execution time of TRT, SCT and pixels
- ϕ resolution (influences SCT/pixels execution time)
- $1/p_T$ resolution (influences SCT/pixels execution time)
- Track finding efficiency for all tracks with $p_T > 0.5$ GeV
- Rate of multiple reconstruction of a given track
- Rate of reconstruction of fake tracks
- Electron identification (only for $J/\psi(e^+e^-)$)

The values for the execution times of the TRT full scan algorithm are presented in section 10. The ϕ and p_T resolutions are discussed in section 11, and the last four items are reviewed for B-physics events in section 12.

For the algorithm described here, only the central positions of the TRT straws are used, drift-time information is not taken into account. The p_T resolution achieved without drift-time is expected to be sufficient to define the RoIs for further track searches in the SCT/pixel detectors.

[2] p_T^{gen} is used for the generated p_T, which will be the initial p_T in the real experiment, and p_T^{rec} is used for the reconstructed p_T.

The track finding efficiency, the rate of fake tracks and the electron identification of the algorithm presented in this note are compared in the text with the off-line pattern recognition package xKalman [4], which was chosen as the reference algorithm. In xKalman an initial track search in the TRT produces seeds for subsequent searches in the SCT/pixel system using a Kalman filter-smoother algorithm. The final step is a global track fit to TRT, SCT and pixels. Since xKalman was optimised for the final performance, it does not stress the reduction of multiple reconstructed tracks after the initial search in the TRT. Furthermore, it is not designed to give an intermediate track output after the TRT pattern recognition.

The LUT-based algorithm described here is equivalent in functionality to the initial search of the TRT in xKalman. It is these algorithms which are compared for benchmarking. xKalman makes use of the search in the SCT and pixels in addition to TRT information to reduce the number of track candidates. A comparison with the LUT TRT Fex has been made after the complete xKalman as this represents the best performance which could be achieved.

3 TRT Detector

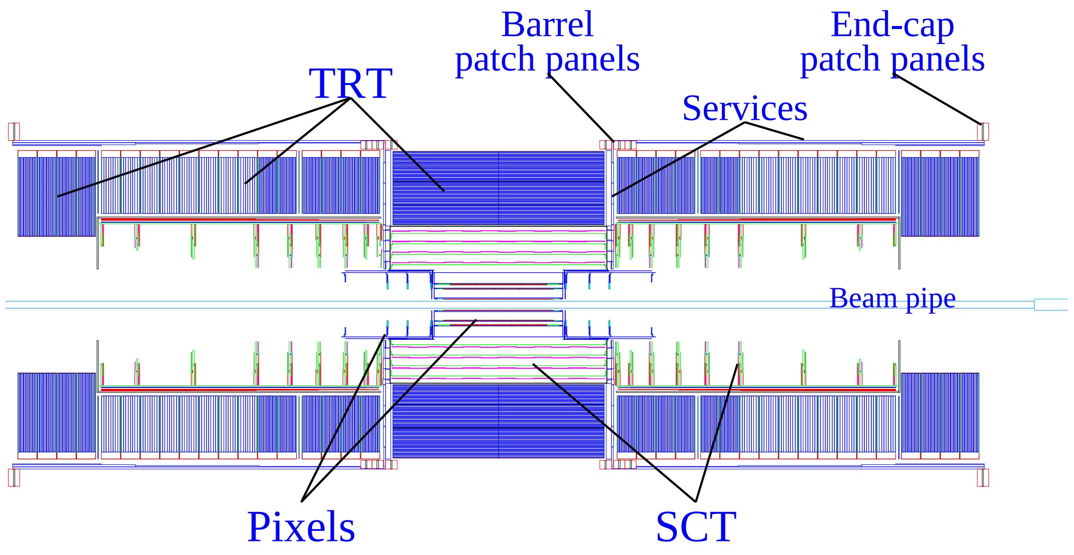


Figure 1: **Inner Detector.** *R-Z projection of the Inner Detector. Figure taken from Inner Detector TDR.*

The TRT is based on the use of straw detectors. Electron identification capability is provided by employing xenon gas to enable the detection of transition-radiation photons created in a radiator between the straws.

The TRT consists of a barrel with a half-length of 74 cm and end-caps in the region 83 cm $< |z| <$ 335 cm. The sensitive element is a straw of internal diameter 4 mm with a single sense wire running down the centre. In the barrel the straws run in a direction parallel to the beam pipe and are 0.68 cm apart. In the end-caps the straws are orientated radially with 576 to 768 straws per layer, giving a straw spacing of 1.1 cm at the outer radius. The barrel has an inner radius of 56 cm and outer radius of 107 cm. However, for the inner layers up to a radius of 62 cm, the active part of the straw is limited to $|z| >$ 40 cm. Outside this radius the straws have an electrical break at z = 0. Each half of the barrel is read out separately.

The transition from barrel to end-cap geometry occurs in the TRT in the region $0.66 < |\eta| < 1.08$. Each end-cap consists of 14 short wheels (64 cm $<$ R $<$ 103 cm) in the region 83 cm $< |z| <$ 278 cm and, at the highest $|z|$, four long wheels with the same outer radius but an inner radius of 48 cm.

The TRT is constructed to to make typically 36 measurements on every track in the whole covered η range. In addition to the presence or absence of a hit, the TRT provides drift-time information.

The geometry used in this study to describe the TRT is from DICE 97_6.

4 Barrel Algorithm

The search for track candidates is performed using the same basic method applied to both the barrel and the end-caps. This consists of an initial search using a histogramming method followed by a fit. The implementations of the C++ and VHDL versions of the algorithm differ, since the hardware architecture is quite different. Sections 4 and 5 describe the algorithms and the implementation in C++. Section 8 describes the implementation into an FPGA-processor, where the method deviates from the C++ implementation.

The barrel algorithm consists of the following steps:

- **Initial track finding:** utilises a LUT-based Hough Transform to find potential track candidates.

- **Local maximum finding:** selects potential track candidates and eliminates multiply reconstructed tracks.

- **Track splitting:** removes hits incorrectly assigned to a track and splits tracks that have been erroneously merged.

- **Final selection:** selects final track candidates.

- **Track fitting:** performs a fit in the R-ϕ plane using a third order polynomial to improve ϕ and p_T reconstruction. The algorithm uses only the straw position (i.e. the drift-time information is not used).

4.1 Initial Track Finding

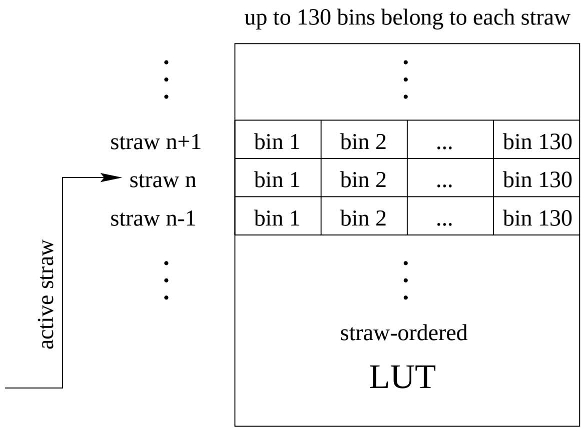


Figure 2: **Structure of LUT for Hough Transform.** *The LUT stores for all possible straws the corresponding histogram bins, for which the counters have to be incremented. All active straws (=hits) of an event are put into this LUT, which returns the bin numbers (each bin number codes one $\phi, 1/p_T$ combination) which have to be incremented. For the end-caps, the symmetrical detector geometry is used to reduce the size of the LUT (section 5.1). The barrel symmetry is not yet used.*

The initial track finding applies a **look-up table (LUT) based Hough Transform** [5]. The Hough Transform is a standard tool in image analysis that allows recognition of global patterns in an image space by recognition of local patterns (ideally a point) in a transformed parameter space. The basic idea of this technique is to find curves that can be parameterised in a suitable parameter space. In the barrel, the Hough Transform performed is from (R, ϕ) space to ($\phi, 1/p_T$) space. The LUT consists of 96 000 ($\phi \times 1/p_T = 1200 \times 80$) pre-defined roads. All pre-defined roads point to the origin. The assumption of straight lines in the R-ϕ projection is not sufficiently accurate for low-p_T tracks. Therefore pre-defined overlapping roads are computed as **exact circles** in the x-y projection, with an increasing road width for increasing R. The equation of the centre of the road is:

$$qC_TR = sin(\phi - \phi_0)$$

where $C_T = 0.3/p_T$, ϕ_0 is the initial azimuthal angle of the particle and q is the particle charge. This equation is solved numerically for all (R, $1/p_T$) pairs where $1/p_T$ is defined by the road and R is the radius of the straw. The results are stored in a table.

The road width increases linearly from 4.5 mm at layer 1 (numbering from the innermost layer outwards) to 6.8 mm at layer 42 and is then constant and equal to 6.8 mm from layer 42 to layer 73. With this definition, $\approx$ 65 straws are assigned to each road.

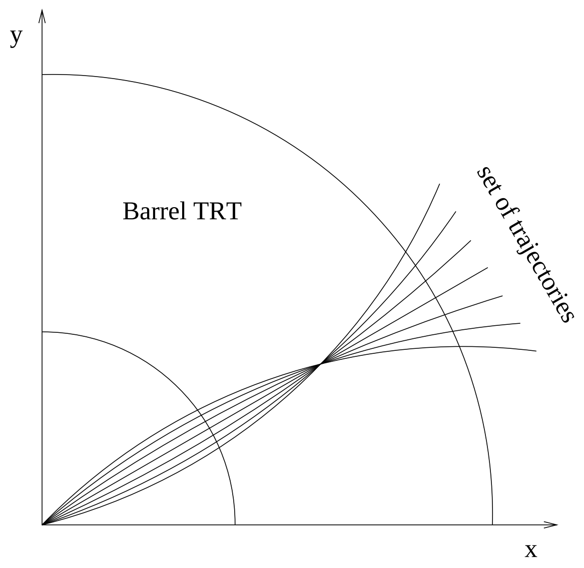


Figure 3: **Set of road trajectories.** *A set containing 80 road trajectories with two common points is computed and then rotated in the barrel and the straws belonging to the roads are entered in the LUT.*

The road trajectories are calculated in sets (bundles) and straws assigned to each road. A bundle of road trajectories is shown in Figure 3. It is a collection of roads with $1/p_T$ values spanning the range from -2 GeV^{-1} to +2 GeV^{-1} in 80 steps of equal width $\Delta 1/p_T$. The roads in a bundle have two common points in the x-y plane, the origin, and a point on the circle R = 0.8252 m. The roads in a bundle therefore correspond to a set of idealised trajectories passing through these points for particles of both charge signs with p_T between 0.5 GeV and infinity.

The radius chosen as the common point for the bundles is located mid-way through the TRT barrel and at a point between two layers to avoid an unequal distribution of straws in the LUT. The ϕ calculated in the Hough Transform space $(\phi, 1/p_T)$ is the ϕ at this radius. Tracks that are close in p_T and ϕ at this radius are close in space. Using this definition of ϕ simplifies the selection of correct track candidates (section 4.2).

One bundle (35 kBytes) is stored on disk and, during the initialisation phase, the full detector LUT is calculated by rotation of the LUT for the bundle. The initialisation takes a few seconds to produce 1200 track bundles shifted by a constant $\Delta\phi = 2\pi/1200$.

The pre-defined roads overlap by 30 % - 50 % in $1/p_T$ and ϕ. This overlapping of the roads prevents the loss of hits from a track with a trajectory which could otherwise pass between two pre-defined roads, but can lead to multiply reconstructed tracks, which have to be eliminated in subsequent steps. Each straw is assigned to ~120 roads (max. 130), as shown in Figure 2.

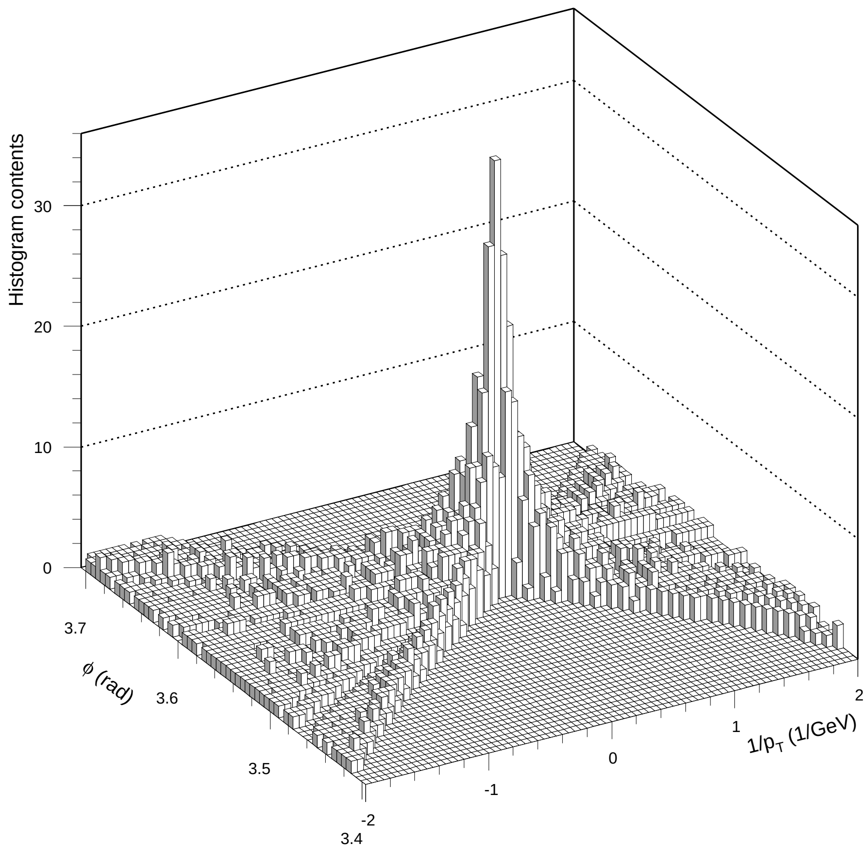
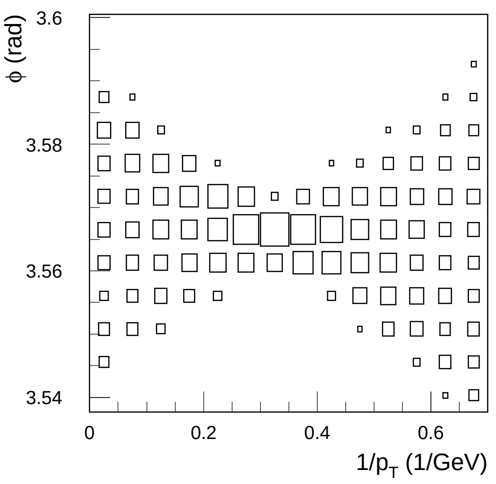


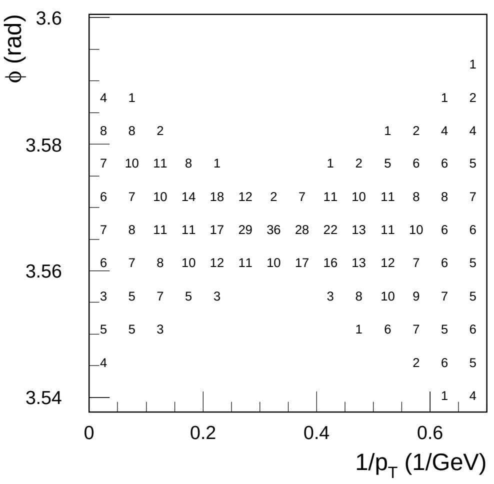
Figure 4: **Histogram due to single muon with $p_T = 3\,\mathrm{GeV}$ in the barrel TRT.** *Each point (hit straw) on the track in (R, ϕ) space is transformed into a quantised curve in Hough space $(\phi, 1/p_T)$. The intersection point of these curves is located in the bin with the highest number of hit straws.*

The Hough Transform proceeds in the following way: for each hit straw in the event, counters are incremented for all roads containing that straw. Each counter corresponds to a bin in a histogram in $(\phi, 1/p_T)$ space. Bins having $> N_{thr}$ hits, where N_{thr} (14) is the threshold, are identified as potential track candidates. The result of the initial track finding is a histogram. An example is shown in Figures 4 and 5 for a single muon with $p_T = 3$ GeV. One can see the steps in ϕ and slope, which corresponds to $1/p_T$. Each bin corresponds to a road in $(\phi, 1/p_T)$ space and the content of the bin is the number of active straws (=hits) in this road.

4.2 Local Maximum Finding



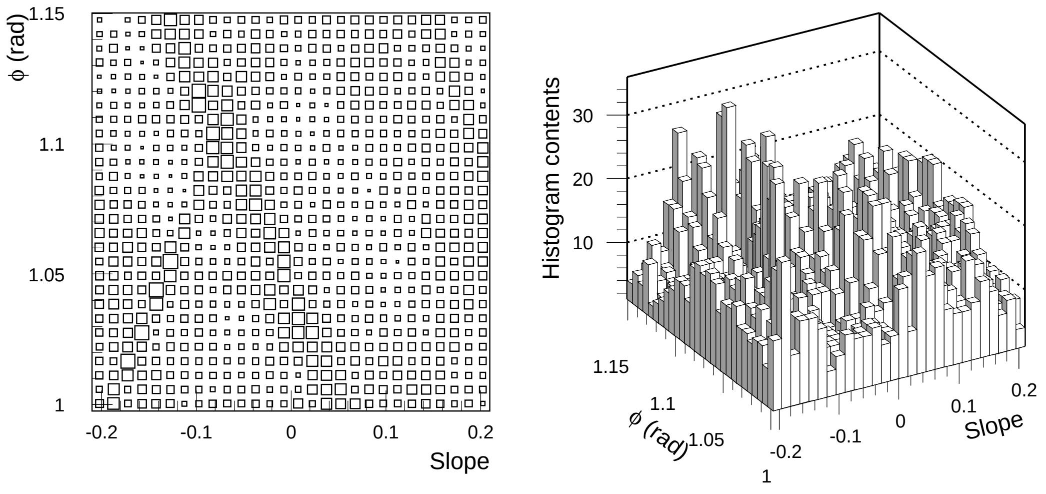
(a) **Box view.**



(b) **Value view.**

Figure 5: **Detail of histogram due to single muon track with $p_T = 3$ GeV in the barrel TRT.** *Resulting from the initial track finding. Slope corresponds to $1/p_T$. Bins having > 14 hits are identified as potential track candidates. The maximum finder chooses the local maximum with respect to the 8 neighbouring bins in ϕ and $1/p_T$.*

The histogram for a single track consists of a "bow-tie" shaped region of bins with entries with a peak at the centre of the region. An example histogram is shown in Figure 4 for a single muon. The bin at the peak of the histogram will, in the ideal case, contain all the hits from the track. The roads corresponding to the other filled bins share straws with the peak bin, and so contain sub-sets of the hits from the track. The fact that the roads overlap in both ϕ and $1/p_T$ increases the number of bins with entries from a given track.



(a) **Box view.** (b) **Column view.**

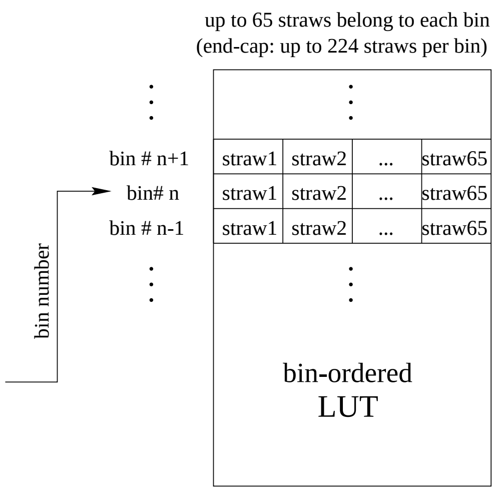
Figure 6: **Detail of typical histogram of a B-physics event.** *There are only two tracks in this slice of histogram, most of the bins are filled with active straws from tracks lying outside of this slice.*

The histogram for a more complex event consists of a superposition of the entries from the individual tracks, see Figure 6. The bins containing the complete set of points from each track can be identified as local maxima in the histogram. Tracks with p_T below 0.5 GeV do not give a peak within the $1/p_T$ range of the histogram, but contribute to the bin occupancy.

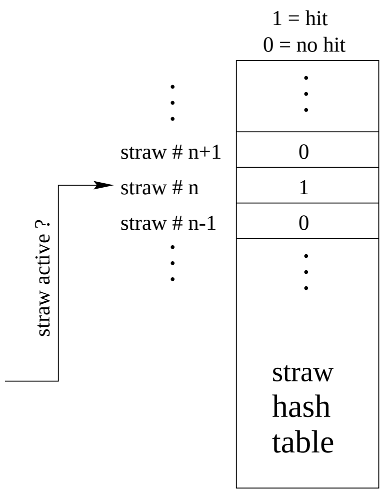
A cut on the number of hits is first applied to reduce the number of bins to be considered by the maximum finder and to eliminate small peaks due to bins with entries from sub-sets of the hits from more than one track. Histogram bins having more than N_{thr} (14) hits are identified to be considered by the maximum finder. In the example for a single muon shown in Figures 4 and 5, eight bins are selected within a small range of ϕ and slope. The maximum finder selects as track candidates bins which have more entries than the immediately neighbouring bins. If two neighbouring bins share the same number of hits, only one bin is chosen as a track candidate.

The maximum finder gives a large reduction in the number of track candidates compared with a simple threshold cut. For B-physics events with pile-up corresponding to low luminosity running, the local maximum finder gives a factor of 10 reduction in the number of candidates.

4.3 Track Splitting



(a) **Bin-ordered LUT.** *Each bin corresponds to a pre-defined road, and the LUT gives all straws corresponding to this road. To select only the straws with hits, the straw hash table is used (see (b)).*



(b) **Straw hash table.** *This hash table, which is filled once per event, provides the possibility to tell quickly whether a certain straw has a hit or not. This is needed by the bin-ordered LUT to extract only the active straws of a road.*

Figure 7: **LUT and hash table for association of track candidate and corresponding straws.**

In this step, the pattern of hits associated to a track candidate is analysed. In order to achieve this in a time-efficient way, a second "bin-ordered" LUT is constructed at the initialisation phase. It differs from the "straw-ordered" LUT described in section 4.1 in that it uses the bin number rather than the straw number as the index, see Figure 7(a). Each bin corresponds to a road. The list of straws lying within the road is stored in the LUT. Furthermore, to speed up the retrieval of the information on which straws in the road have a hit, a hash table[3] is filled once per event with the pattern of 0's and 1's corresponding to straws without and with hits. This is illustrated in Figure 7(b).

The track splitting step applies the following criterion: if potential track candidates contain ≥ 9 consecutive layers without a hit, the track is split into two separate candidates either side of the gap. If one of the candidates contains more

[3]A hash table is a method for directly referencing records in a table by performing arithmetic transformations on keys into table addresses. Here, any search is executed with only one memory access by simply using the key to address the table entry.

than N_{thr} (14) hits, it is retained. If both candidates pass this threshold, the track segment which starts at the lowest radius is retained.

By rejecting fake candidates composed of hits from several low-p_T tracks, the track splitting step results in a overall reduction by a factor of $\sim$2 in the number of track candidates. In addition, for roads containing a good track candidate, it identifies and rejects any additional hits from one or more other tracks. This is particularly important for tracks which traverse from the barrel to the end-cap and for tracks in the central region of the barrel, $|z| < 40$ cm, where layer 10 is the first active layer. The result of the track splitting step is a candidate that consists of a sub-set of the straws in a road. It will have a first and last layer, one or both of which will differ from the end-points of the road. The start of the segment produces rough η information about the reconstructed track in that it can be used to identify candidates entering the barrel in the region $|z| < 40$cm. The η information is used for the final selection step described in the next section.

4.4 Track Selection and Track Merging

In this stage of selection, track candidates are classified according to the layer number (counting from the inside) of the innermost straw with a hit. This classification makes use of the fact that in the region of the barrel at $|z| < 40$ cm, layer 10 is the first active layer, as shown in Figure 8. The layer number of the first hit, therefore, provides some information on the position of the track candidate in z. The selection is as follows:

- If the first hit is in layers 1 to $Layer_{etacut}$ (9) the track is considered to be traversing the barrel/end-cap transition region. All such track candidates are accepted.
- If the first hit is in layers 10 to $Layer_{firsthit}$ (50), the track is assumed to be traversing the region of the barrel at $|z| < 40$ cm. Such a candidate must exceed a threshold of $N_{highthr}$ (16) active straws to be accepted.
- If the first hit is in a layer $> Layer_{firsthit}$ (50), the track is rejected. This cut rejects track candidates which are mainly background or decays, but, in a small fraction of cases, are real tracks on a trajectory which does not point at the origin.

After all described steps are applied, there is on average still more than one reconstructed track segment per generated track remaining, see section 12.2. This is due to the fact that the LUT definition assumes that all tracks come from the interaction region, within a precision of a few millimetres in the x-y plane.

However, due to physics processes in the SCT/pixels (bremsstrahlung), tracks in the TRT may not point to the origin. These tracks are reconstructed as several track segments, none of which contain all active straws belonging to that track. These track segments have to be merged in a subsequent step. This track merging step is still being investigated and results are not yet shown. The track merging will use the fact that several reconstructed tracks from one generated track are in most cases neighbours in the reconstructed track list. In addition, they share over 50% of the hits contributing to the track.

This is the last step of the track finding. The next step is to perform a fit to the track candidates to obtain the reconstructed track parameters p_T, ϕ and η. The track selection step reduces the number of track candidates by a factor of 1.2. The track merging should reduce the number of track candidates by a further factor of 1.2, but this has yet to be demonstrated.

4.5 Track Fitting and Final Selection

For candidates passing the final selection, a fit is performed in the (R, ϕ) plane using a third-order polynomial. The third-order correction is needed for low-p_T tracks, since for them a straight line approximation in the (R, ϕ) plane is not valid anymore. For the fit, the track is assumed to come from the origin. To increase the speed of the track fit, in both the barrel and end-caps, the drift-time information is neglected. If required, the p_T resolution could be improved by utilising the drift-time information. With drift-time, the single hit resolution is improved from 1.2 mm to $\sim$200 μm. However, for low-p_T ($\sim$1 GeV) tracks the ultimate p_T resolution is limited by multiple scattering. The fit using drift-time is more complex as there are two possible positions (left/right ambiguity) for each hit.

After the fit, the threshold, $p_T^{rec} > 0.5$ GeV, is re-applied.

5 End-cap Algorithm

The end-cap algorithm consists of the following steps :

- **Initial track search:** An initial track search using a histogramming method is applied after a Hough transform from (z, ϕ) to $(\phi, 1/p_L)$ space.

- **Threshold cut and 2-D maximum finder:** A threshold cut on the number of hits is applied. Track candidates are identified as bins passing the threshold cut and with a number of entries that is a local maximum.

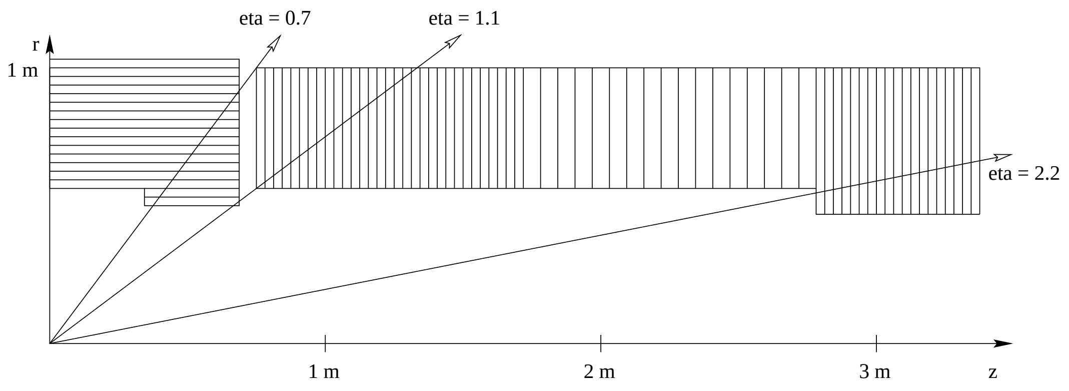


Figure 8: **Schematic cross-section of the TRT.** *The end-cap measures* $1/p_L \sim C_L = \Delta\phi/\Delta z$. *Track splitting produces end-points of the track. This can be correlated to an* r_{min} *and* r_{max} *assumption, and* $\Delta r/\Delta z = tan(\theta)$ *gives* $p_T = 0.3 * tan(\theta)/C_L$.

- **Track splitting:** A track splitting step is used to identify cases where a road contains two (or more) tracks at different η. Roads contain straws from the entire end-cap. However a track from the origin only traverses a sub-set of straw planes in a limited range of $|z|$. The z of the first and last plane with a hit provides an η measurement for the track trajectory.

- **3-D maximum finder:** Track candidates are selected with a number of entries in the histogram that is local maximum in 3 dimensions, making use of the η measurement derived from the track splitting step.

- **Track merging:** Tracks which are multiply reconstructed with slightly different combinations of hits are merged into one track.

- **Track fitting and final selection:** After a straight line fit in z-ϕ, the threshold on the number of hits is raised for track candidates in the central part of the end-caps, where there should be a higher number of hits.

The end-caps of the TRT do not directly measure p_T but rather p_L. A measurement of η, illustrated in Figure 8, allows the calculation of p_T. For this calculation, knowledge of the point of entry and exit to/from the TRT (end-points) are needed. This could be achieved by splitting the end-caps into several overlapping regions and thus producing a rough hypothesis of the end-points of the track.

A different solution was chosen for the algorithm described in this note. A single LUT is defined for the entire end-cap. A separate step (track splitting) is used to determine the start and end planes of the track segment. This step is described in more detail in section 5.3. This solution does not introduce additional overlapping zones. This method has the following advantages over the use of overlapping regions:

- For each overlapping region either more computation is needed, because hit straws have to be considered twice, or there is a loss of signal efficiency or background rejection.
- Multiple reconstructed tracks have to be eliminated in subsequent steps.

The end-cap algorithm is optimised in terms of execution time and track reconstruction quality for a **full scan** at **low luminosity**. However, the method described here is not restricted to low luminosity. For a track search at high luminosity, a different parameter set (defining thresholds etc.) is required.

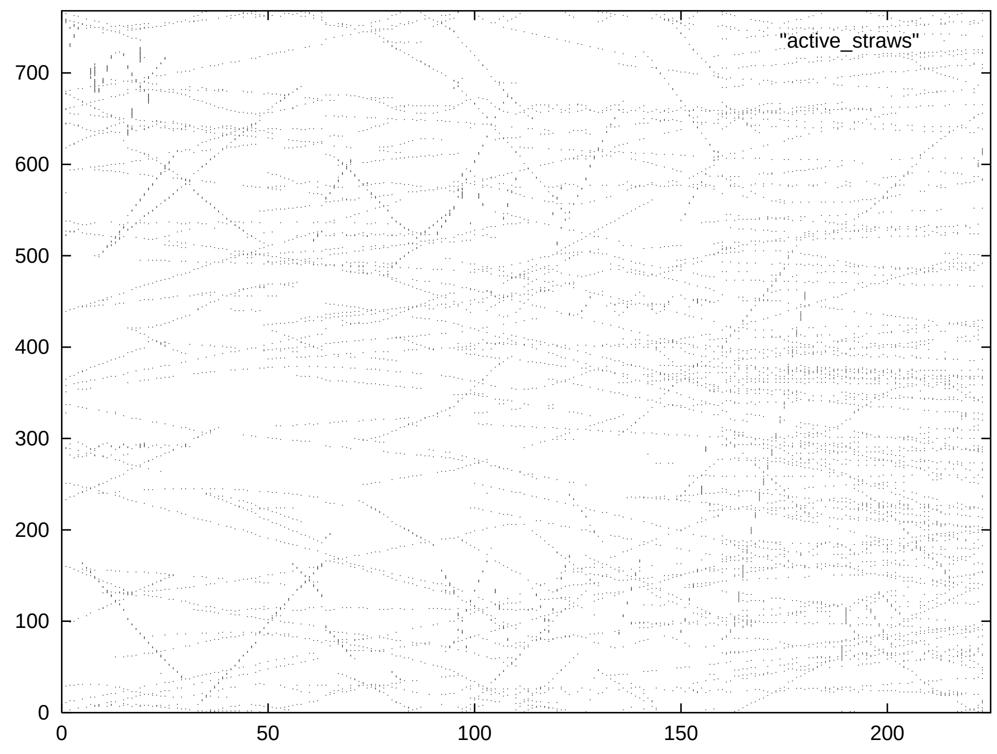


Figure 9: **Typical B-physics event at low luminosity.** *Projection in z-ϕ. Plane numbers in z and "straw in plane" numbers in ϕ. From plane 160 upwards the "straw in plane" numbers are multiplied with 4/3 to compensate for the fewer number of straws in that planes. It can be seen that tracks can start at any plane number.*

5.1 Initial Track Finding

In the end-caps, the Hough Transform is performed from (z, ϕ) space to (ϕ, $1/p_L$) space. The track finding step makes use of the assumption that the particles are produced at the origin. The TRT end-cap has an 192 fold geometrical symmetry[4]. This is exploited to reduce the size of the LUT[5]. The resulting LUT dimension using this symmetry is $\phi \times 1/p_L = 6 \times 80$ instead of 1192×80. How this is achieved is explained in more detail below. The roads are overlapping, the road width (in ϕ) for each plane is 2 π / (number of straws in this plane).

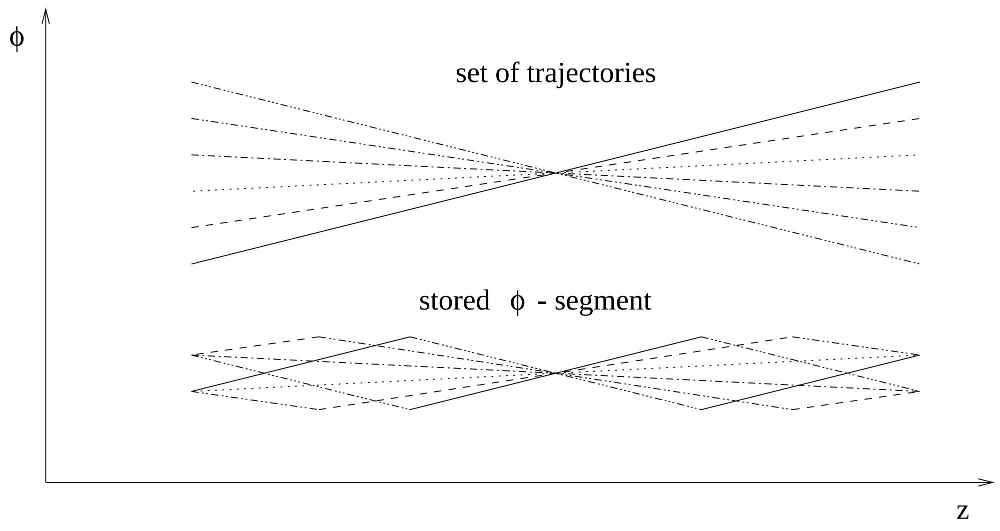


Figure 10: **Set of road trajectories and stored trajectory segment.** *A set containing 80 road trajectories is computed and only a fraction in ϕ of the end-cap is stored.*

The LUT stores one segment of 1/192 of 2π. The number of straws in ϕ is different for short and long end-cap straws (see Figure 8). A segment contains:

- 4 straws per plane ($768 / 192 = 4$) in the planes with $|z| < 280$ cm
- 3 straws per plane ($576 / 192 = 3$) in the planes with $|z| > 280$ cm
- 6×80 pre-defined roads ($1152 / 192 = 6$) in the bin-ordered LUT

[4]1 / 192 is the biggest common denominator of both types of planes with a symmetry of 1 / 576 respectively 1 / 768.

[5]The barrel TRT will also have a symmetry which can be exploited to reduce the size of the LUT. However, the detector description used to produce the simulated data used for this study did not have this symmetry.

In addition to this "straw-ordered" LUT there is a "bin-ordered" LUT for the further algorithms steps, like in the barrel. Using the symmetry, the bin-ordered LUT stores 6 bundles with different ϕ at z = 223.35 cm and 80 pre-defined $1/p_L$.

A symmetric LUT does not necessarily introduce overlaps. The LUT for the end-caps is implemented so as not to introduce overlaps. The use of symmetry for the LUT is fully transparent for the algorithm in this end-cap implementation. This means, that although the stored LUT is much smaller in $\Delta\phi$ than the range a typical low-momentum track traverses, no part of a track is lost. For each active straw, the ϕ offset relative to the stored LUT segment and the corresponding histogram counter number offset is computed. The histogram counter numbers corresponding to the straw in the stored LUT segment are read out, the constant counter number offset is added and the correct histogram counters are incremented. The values of the histogram counters are only interpreted after all hits from the whole end-cap are entered into the histogram, thus avoiding the need for overlapping segments[6].

After the histogramming process, bins with contents exceeding a threshold value of N_{thr} (14) hits are retained as potential track candidates. Since only one LUT is used for the whole end-cap volume, track trajectories with different $1/p_T$ are mixed in the same $1/p_L$ bin. As in the barrel, a bundle of pre-defined roads for different slopes has a common point in the TRT. This point, where a bundle of $1/p_L$ slopes intersect in the end-cap, is chosen to be at z = 223.35 cm. This value does not correspond exactly to the geometrical centre of the end-cap, instead it is chosen towards a higher value in z to reduce the number of multiply reconstructed tracks at high η.

Since tracks in the end-caps coming from the interaction point do not traverse the whole z-range of the end-cap, there are different scenarios for local maximum finding depending on η of the track. However the η of the track is unknown at this stage. For this reason, only a loose maximum finding algorithm can be applied after the initial track finding.

The $1/p_L$ range of the histogram is chosen to cover the $|p_T|$ range down to 0.5 GeV. Since a reduced $1/p_L$ range is required at high $|\eta|$, the number of bins and $1/p_L$ range is reduced in two steps as a function of plane number, see Figure 12. This reduces the number of histogram counters that have to be incremented compared to using a constant $1/p_L$ range and hence reduces execution time. Another benefit is reduced hit occupancy of the roads corresponding to low p_T, since the number of planes contributing to a road is reduced (for bins with a $1/p_L$

[6]A subsequent step of the algorithm, the maximum finder, compares neighbouring histogram counters. Having the full histogram available at once enables the comparison of all neighbours without the introduction of overlaps.

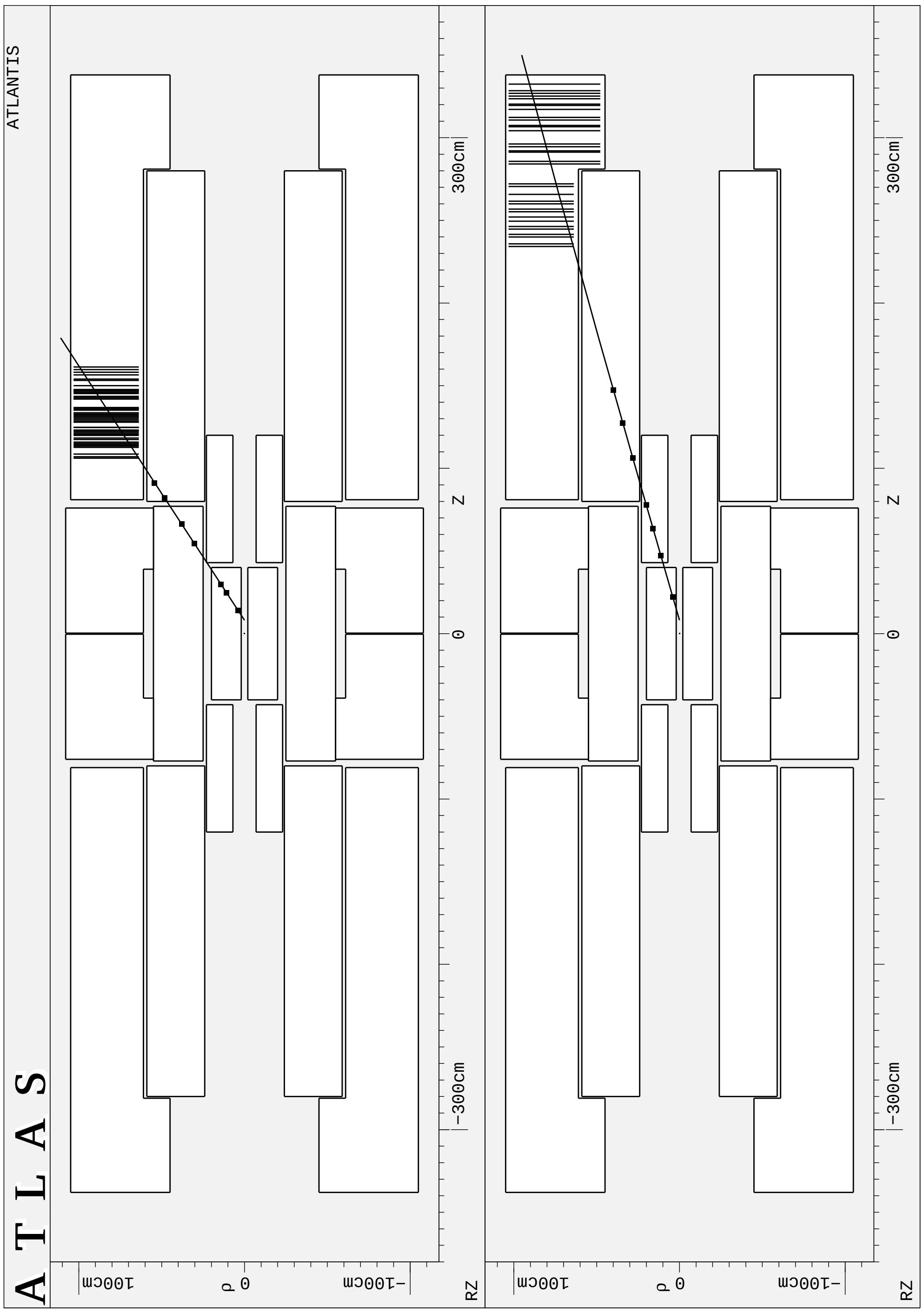


Figure 11: **Two different End-cap tracks.** *TRT straws are shown as lines, SCT/pixel space-points are shown as dots.*

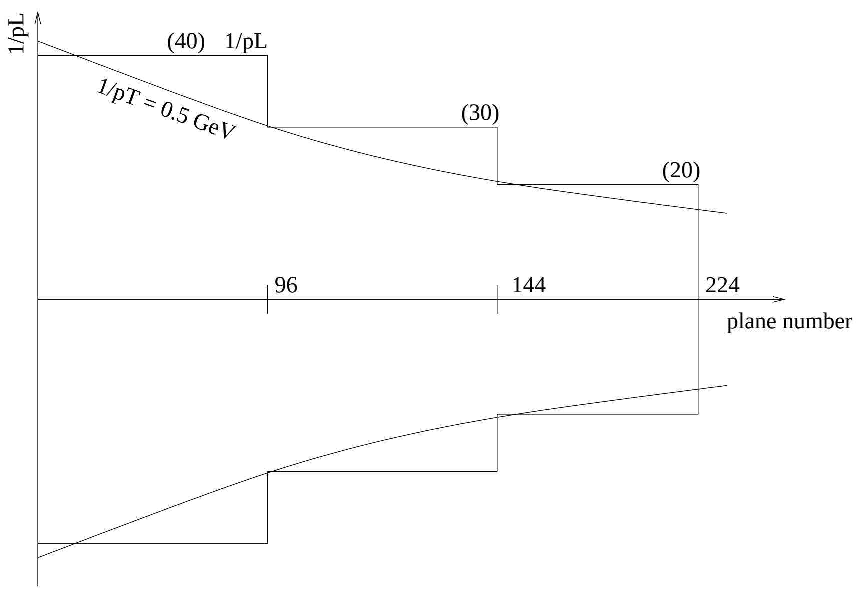


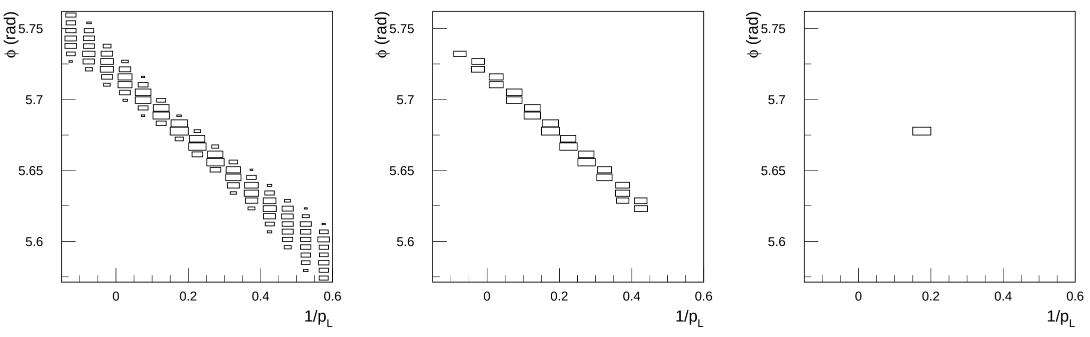
Figure 12: p_L - p_T **relationship.** *The smooth curve shows the value of* $1/p_L$ *corresponding to a value of* $1/p_T$ *of 0.5 GeV as a function of plane number (increasing with z). This defines the range of* $1/p_L$ *of the histogram corresponding to a* $1/p_T$ *threshold of 0.5 GeV and hence the number of bins (indicated on the figure in brackets) required for a fixed bin width in* $1/p_L$*. The number of bins is reduced in two steps with increasing plane number.*

value above the stepped curve in Figure 12). In addition the size of the LUT is reduced. The steps in the $1/p_T$ threshold have an impact on the reconstruction efficiency for low momentum tracks, this will be discussed in section 11.1.

5.2 Threshold and 2-D Maximum Finding

After the initial track finding step a threshold is applied. All bins with more than N_{thr} (14) hits are selected. An example of the effect of the threshold cut is shown in Figure 13(b) for a single muon track. This shows that a single track results in several bins above threshold. The task of the 2-D maximum finder is to determine which bin contains the most hits from the track. It is important to reduce the number of bins to be considered by subsequent stages in order to minimise the execution time for the algorithm as a whole.

The 2-D maximum finder works by comparing bins in an "H" shaped region of the histogram, as illustrated in Figure 14. To be selected, the bin at the centre of the region must contain more entries than any other bins in the region. Figure 15 shows examples of histograms produced by muons at $\eta = 1.1$ (left) and $\eta = 2.5$



(a) **Histogram after the initial track finding.**

(b) **Histogram after the threshold cut.** *There are 22 potential track candidates (histogram bins above threshold). They are ordered in a shape to which the 2-D maximum finder is adapted.*

(c) **Histogram after the 2-D maximum finding.** *The shape of the 2-D maximum finding chooses only the one best track candidate with the highest number of hits.*

Figure 13: **The histogram for a muon at $\eta = 2.5$** *after the initial track finding, after the threshold cut and after the 2-D maximum finding.*

(right). The reason for the choice of the shape of the region considered by the maximum finder is apparent from the shape of the histogram peaks in the two cases. The shape of the region of filled bins is independent of charge and not strongly dependent on p_T. The η dependence is due to the common intersection point for road bundles approximately in the middle of the end-cap. Figure 13 shows the same event as Figure 15(b), after the application of the threshold cut (middle) and after the maximum finding algorithm has been applied (right). Only one bin survives the selection.

The effect of pile-up is to increase the number of bins per track that pass the selection. The number of candidates could be reduced by increasing the size of the region considered by the maximum finder. This would reduce the probability that two bins are selected within the same histogram peak. However it would adversely affect the ability to resolve two tracks close in ϕ and p_L. In events with many tracks, peaks above threshold can occur in bins with entries from more than one track. These spurious isolated peaks are likely to be accepted by the maximum finder regardless of the size of the region considered by the algorithm described here. Instead, additional algorithmic steps are used to reduce further

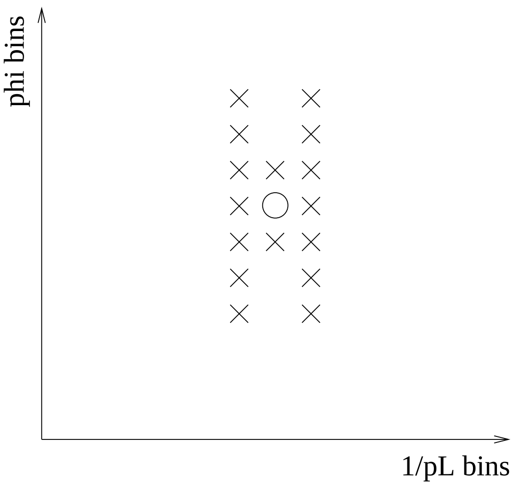
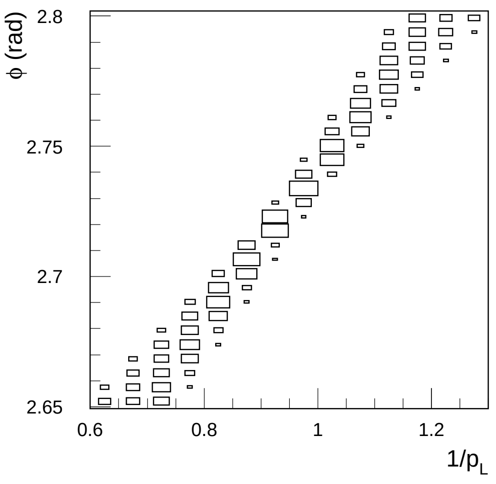


Figure 14: **2-D Maximum Finding.** *Bins exceeding the threshold (circle) must be a local maximum with respect to neighbouring bins (crosses) in both dimensions ϕ and $1/p_L$.*

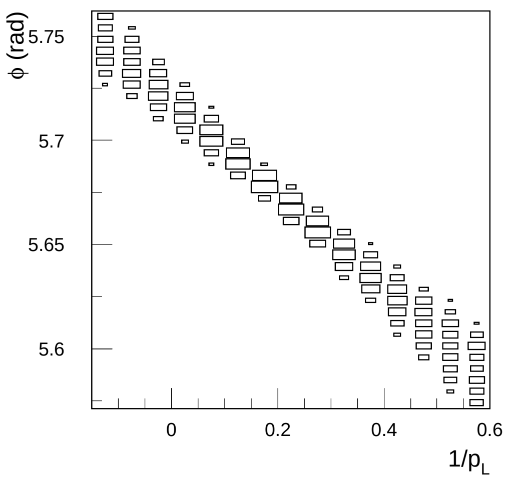
the number of track candidates. One of these steps is a 3-D maximum finder which uses, in addition to ϕ and $1/p_L$, η information derived from the track splitting step described in the following section.

5.3 Track Splitting

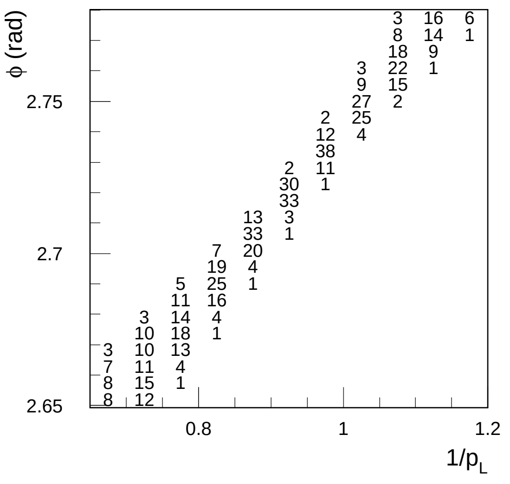
The next stage is a track splitting step similar to that described for the barrel. A road contains straws from the entire end-cap. However a track from the interaction point will traverse only part of the end-cap in a limited range of z. Tracks at different η will populate different parts of the road with hits. A road can, therefore, contain the complete set of hits from more than one track with the same p_L. However, the predominant effect is that there may be some hits from tracks with a different p_L from that of the road. The purpose of the track splitting stage is to identify gaps between sequences of hits from different tracks and to remove spurious additional hits. The algorithm divides a track into two if the number of consecutive planes with no hits is greater or equal to $N_{isolation}$ (8). Any resulting track candidates with more than N_{thr} (14) hits are retained. As a result of the track splitting algorithm, the end-points of a sequence of hits from a track are identified. This gives a measurement of the z coordinate at which the track entered and exited the end-cap, from which the η of the track can be calculated. The quality of the η measurement is a function of η and will depend on detector occupancy, in-efficiency and depends critically on the performance of the track splitting algorithm.



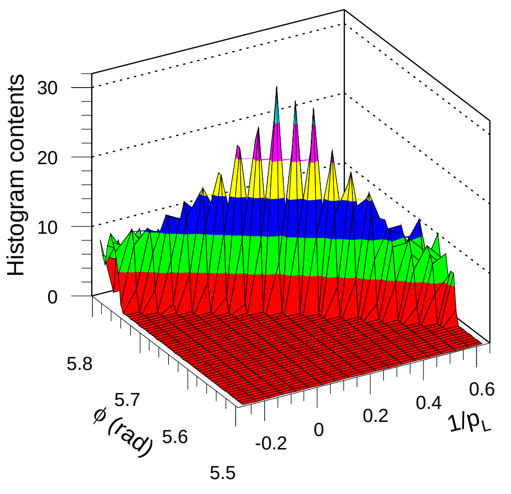
(a) **Muon at $\eta = 1.1$.** *Same as (c).*



(b) **Muon at $\eta = 2.5$.** *Same as (d).*



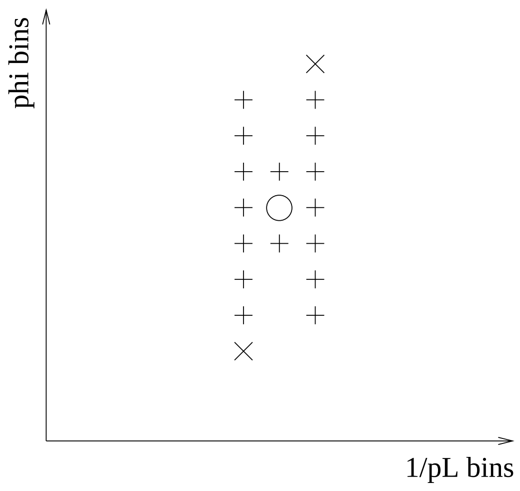
(c) **Muon at $\eta = 1.1$.** *There are several local maxima over the threshold. The 2-D maximum finder selects the one track candidate with the highest number of hits.*



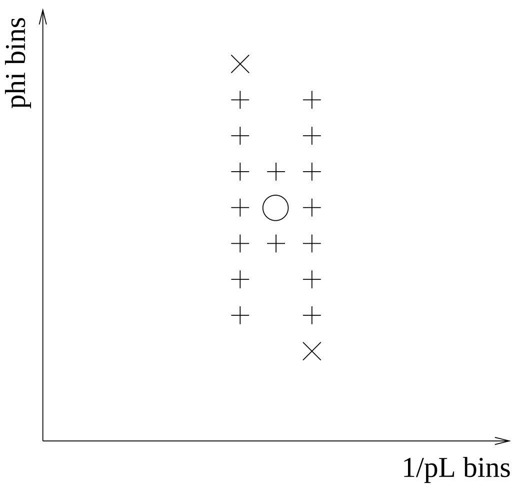
(d) **Muon at $\eta = 2.5$.** *There are several local maxima over the threshold. The 2-D maximum finder selects the one track candidate with the highest number of hits.*

Figure 15: **Histograms are shown for single muons at $\eta = 1.1$ (a) and (c) and $\eta = 2.5$ (b) and (d).** *The bins with entries form a diagonal band in the histogram plane, the orientation of which is determined by η.*

5.4 3-D Maximum Finding



(a) **3-D maximum finding.** *Low η.*



(b) **3-D maximum finding.** *High η.*

Figure 16: **3-D maximum finding shape for low-η and high-η tracks in the end-cap.** *To be retained, in addition to the 2-D maximum finder (+), tracks starting at plane 1-80 (a) respectively plane 151-224 (b) have to exceed the number of hits of the marked bins (x).*

As shown in Figure 15, the shape of the region of the histogram populated by a single track depends on η. For the next step in the selection, the η information determined from the track splitting stage is used to extend the region considered by the 2-D maximum finder in an asymmetric way, depending on the η of the track. A knowledge of η can be used to define different shaped regions to be considered depending on the z plane number of the first hit on the track. The shapes used are shown in Figure 16 for tracks starting at planes 1-80 (Figure 16(a)) and tracks starting at planes 151-224 (Figure 16(b)). No additional selection is applied for tracks starting at planes 81-150. For these tracks, the plane at which ϕ is measured (the common point of the bundle) is roughly at the mid-point. The resulting histogram is therefore the same as the "bow-tie" shape seen for the barrel, for which the 2-D maximum finder alone works well. The 3-D maximum finder results in a small reduction in the number of candidates per track.

5.5 Track Merging

After all the selection steps described so far have been applied, the number of candidates per track is still significantly greater than 1. The results of measurements will be shown in section 12.2. Furthermore, in many cases multiply reconstructed tracks do not contain all hits coming from the generated particle.

Therefore, a step merging the different track segments is needed. In the end-cap this is particularly important, because the end-points of the track influence directly the η and p_T measurements. Missing hits can seriously degrade the reconstructed track parameters.

In most cases multiple candidates are neighbours in the output track list and the track candidates share more than 50 % of hits. In these cases a final track candidate could be generated by merging the hits from the two track segments. This is an area under study. An algorithm is being developed.

5.6 Track Fitting and Final Selection

The next step is to perform a fit to the hit positions for each track candidate to determine the reconstructed track parameters, p_T, ϕ and η. A straight line fit is performed in the z-ϕ plane. For increased speed, the drift-time information is neglected. The inclusion of drift-time would lead to two possible positions per hit, either side of the sense wire. This would necessitate a further stage of selection to choose one of the two possible positions for each hit, see Figure 17. The omission of the drift-time information leads to some degradation of the p_T resolution for high-p_T tracks. However, for low-p_T tracks, the p_T resolution is limited by multiple scattering. The measured $1/p_T$ resolution will be shown in section 11.3. After the fit, the p_T threshold cut of $p_T > 0.5$ GeV is re-applied.

Initially a low threshold on the number of hits is applied in the end-caps so as to maintain high efficiency in the barrel-to-end-cap transition region. In the final selection stage, the threshold is raised to $N_{highthr}$ (16) for tracks outside the transition region, i.e. tracks ending at plane number > 72. This results in a small reduction in the number of candidates. The lower threshold in the transition region results in an increased number of fake track candidates composed of hits from several low-p_T tracks.

Another consequence of the lower initial threshold is an increase in execution time due to an increased number of bins that must be considered in the selection steps. An increase in speed could be obtained by performing the histogramming process in two steps. Firstly the hits from straws in the transition region would be entered into the histogram and the low threshold applied to produce a list of track candidates in the transition region. Then the hits from the remaining

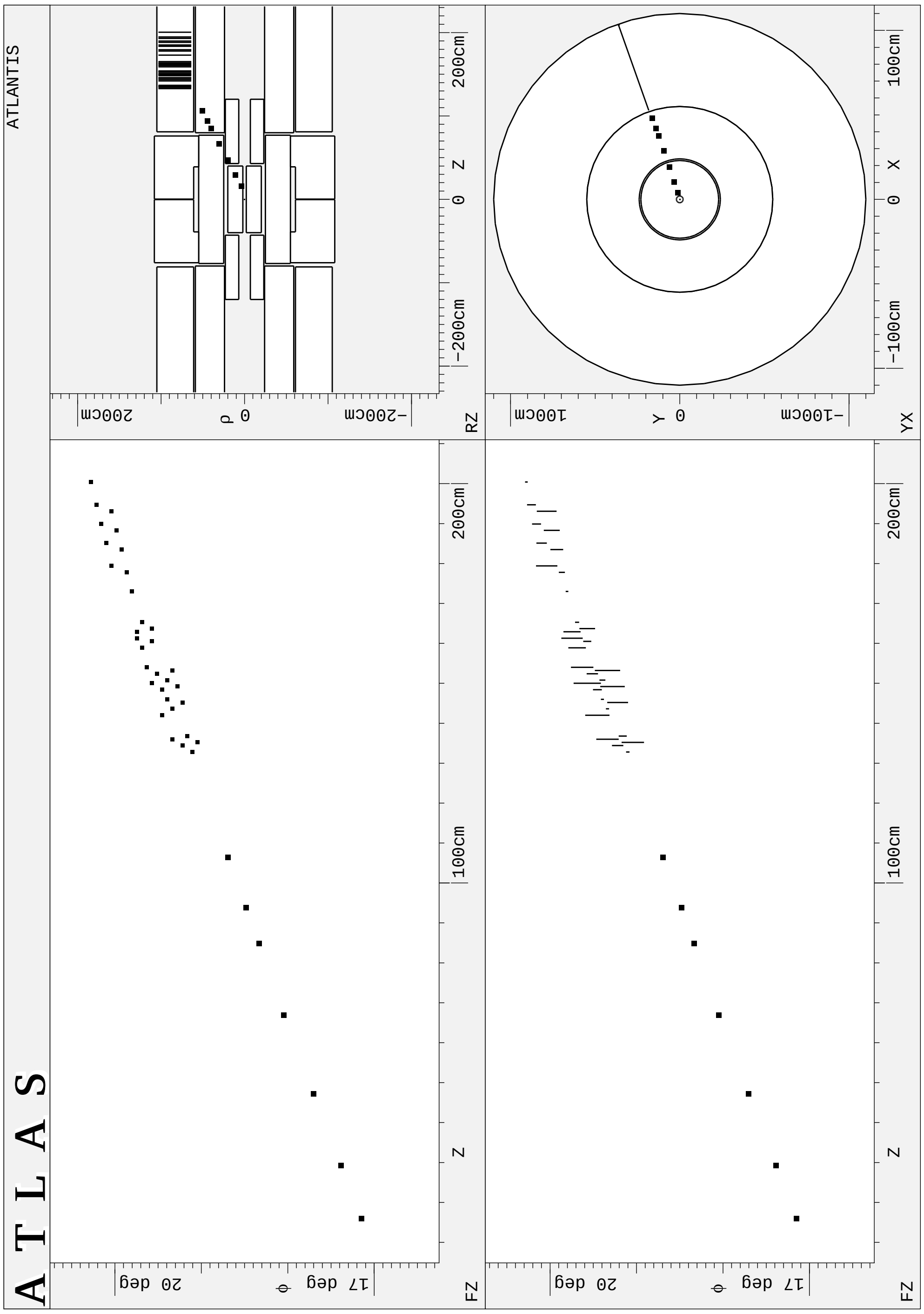


Figure 17: **The benefit of drift-time in the End-cap.** *SCT/pixel space-points are shown as squares, TRT straw central positions are shown as dots (top). TRT straw drift-time information is indicated as a line which joins the two possible positions for the hit (left/right ambiguity) (bottom).*

straws would be added and the higher threshold applied to give a list of track candidates for the remaining part of the detector.

6 Barrel-to-End-cap Transition Region

The most critical part of the TRT for the pattern recognition is the barrel-to-end-cap transition region. In the worst case, tracks have 50% of their hits in the barrel and 50% in the end-cap. Since the precise measurements are done both in barrel and in end-caps in two dimensions, but in a different space (r,ϕ versus z,ϕ), it is not straight-forward to combine those measurements, see Figure 18. In principal, there is the possibility to define a 3-D LUT for the whole TRT instead of two 2-D LUTs for barrel and end-caps. This possibility has been studied and rejected, since the gain of this method is outweighed by the loss in execution time. The method adopted here, is a separate track search in barrel and end-caps. This leads to a lower efficiency in the transition region or to more track segments found if a lower threshold is used. In the barrel, this lower threshold has to be used for all barrel hits, since only after the pattern reconstruction a hint of the position of the track can be obtained. In the end-caps, it is a priori known which hits can contribute to tracks from the transition region. This knowledge can be used to raise the efficiency in the transition region without increasing the execution time.

After the separate track finding in barrel and end-caps, barrel and end-cap track segments from the transition region have to be merged. This step will reduce the track reconstruction multiplicity in the transition region. Merging of the reconstructed tracks of the different detector parts is an area of ongoing study.

7 OO Design

The algorithm was implemented in C++. It was designed to run within the LVL2 testbed reference software [6] framework. This implies conformance to the class definitions given in [7] and shown in Figure 19.

The `TRT_Algorithm` receives as arguments a `LVL1Id`, a `Region` defining the direction and the size of the RoI (in this case the whole TRT), and a `TRT_Data` object. The `TRT_Data` object itself is an aggregation of `TRT_Hit`, implemented as 4 vectors of `TRT_Hit` for the different sub-detector parts (left/right barrel/end-caps). A `TRT_Hit` is a generalisation of `TRT_Straw`, implemented through inheritance. The `TRT_Straw` uses a `TRT_Geometry` Singleton[7] to be able to provide

[7]A Singleton ensures that a class only has one instance, and provides a global point of access

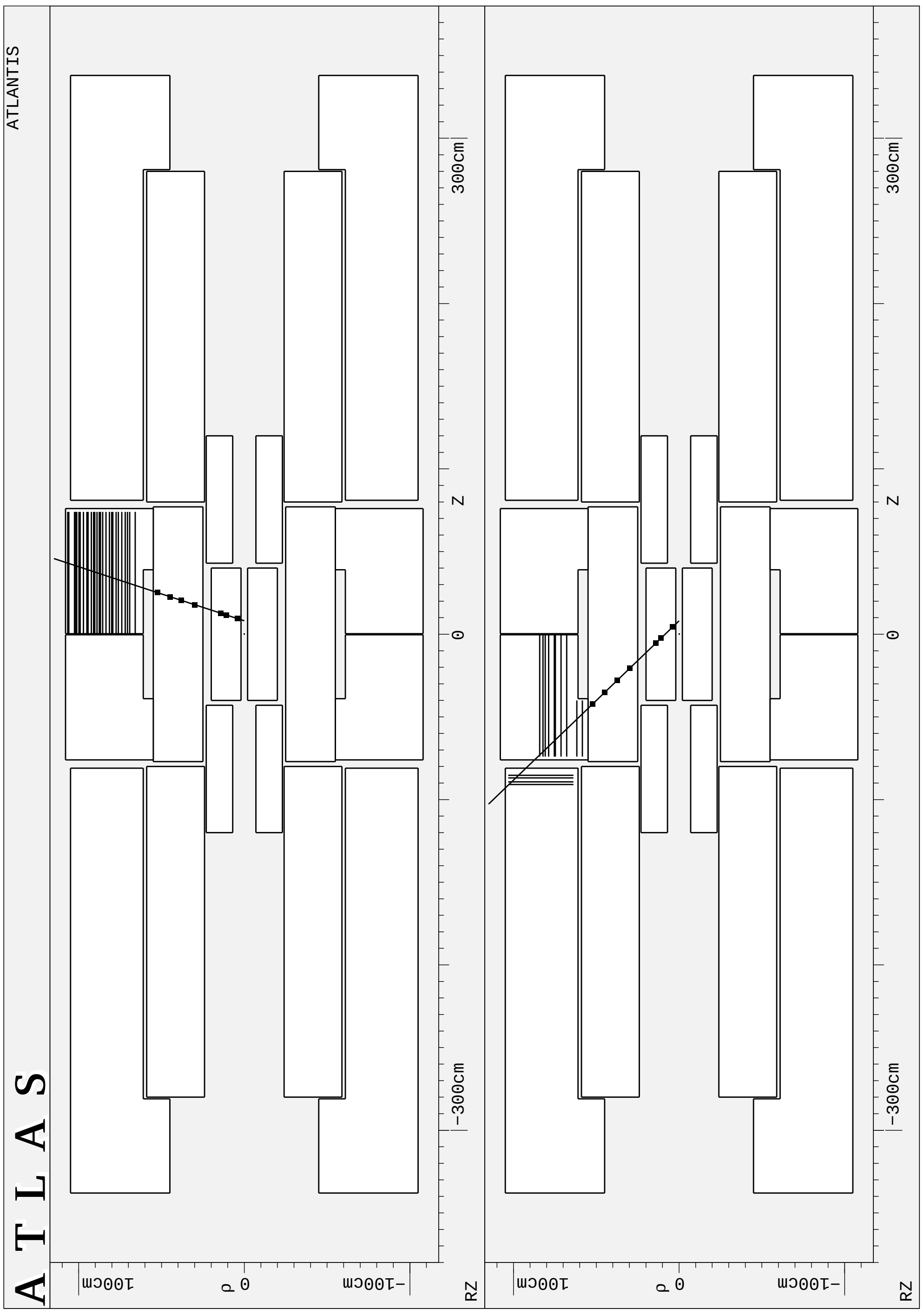


Figure 18: **Barrel and Barrel-to-End-cap transition track.** *TRT straws are shown as lines, SCT/pixel space-points are shown as dots.*

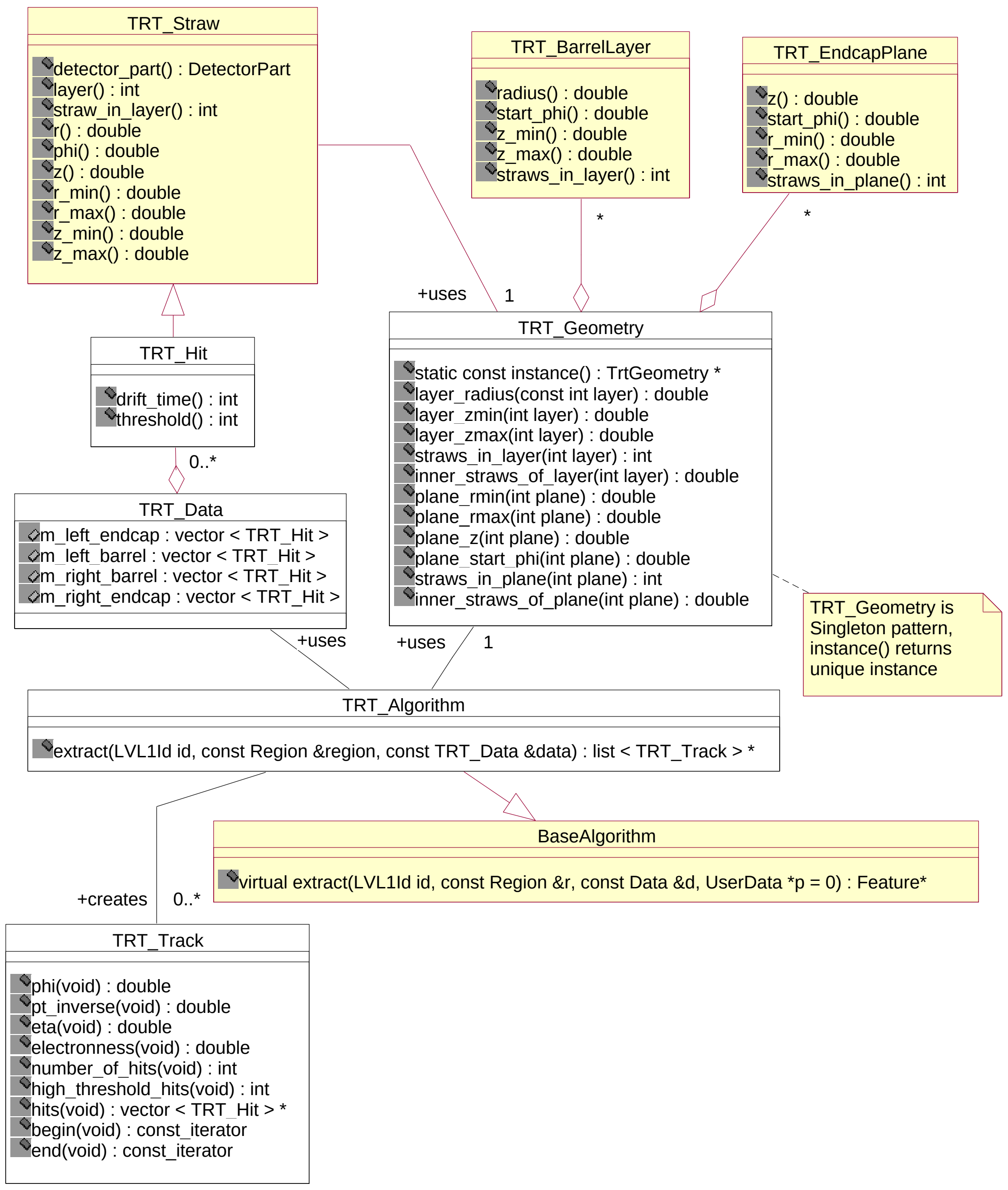

TRT_Straw
detector_part() : DetectorPart
layer() : int
straw_in_layer() : int
r() : double
phi() : double
z() : double
r_min() : double
r_max() : double
z_min() : double
z_max() : double
TRT_BarrelLayer
radius() : double
start_phi() : double
z_min() : double
z_max() : double
straws_in_layer() : int
TRT_EndcapPlane
z() : double
start_phi() : double
r_min() : double
r_max() : double
straws_in_plane() : int
*
*
+uses
1
TRT_Geometry
static const instance() : TrtGeometry *
layer_radius(const int layer) : double
layer_zmin(int layer) : double
layer_zmax(int layer) : double
straws_in_layer(int layer) : int
inner_straws_of_layer(int layer) : double
plane_rmin(int plane) : double
plane_rmax(int plane) : double
plane_z(int plane) : double
plane_start_phi(int plane) : double
straws_in_plane(int plane) : int
inner_straws_of_plane(int plane) : double
TRT_Geometry is Singleton pattern, instance() returns unique instance
TRT_Hit
drift_time() : int
threshold() : int
0..*
TRT_Data
m_left_endcap : vector < TRT_Hit >
m_left_barrel : vector < TRT_Hit >
m_right_barrel : vector < TRT_Hit >
m_right_endcap : vector < TRT_Hit >
+uses
+uses
1
TRT_Algorithm
extract(LVL1Id id, const Region &region, const TRT_Data &data) : list < TRT_Track > *
BaseAlgorithm
virtual extract(LVL1Id id, const Region &r, const Data &d, UserData *p = 0) : Feature*
+creates
0..*
TRT_Track
phi(void) : double
pt_inverse(void) : double
eta(void) : double
electronness(void) : double
number_of_hits(void) : int
high_threshold_hits(void) : int
hits(void) : vector < TRT_Hit > *
begin(void) : const_iterator
end(void) : const_iterator

Figure 19: **Class relations in the TRT Fex System.**

services satisfying requests for physical (3-D straw coordinates) and logical identifiers. To reduce the amount of data to be stored, the `TRT_Straw` object only stores the logical identifiers internally. The `TRT_Geometry` object provides services describing the physical detector geometry.

The result of the Feature extraction process is a pointer to a list of `TRT_Track` objects. `TRT_Track` describes a complete track with all contributing `TRT_Hit` objects, no matter whether it is in the barrel, the end-cap, or in the barrel-to-end-cap transition region.

8 FPGA Implementation

Past experience has shown that the performance of FPGA-based implementations is limited by the extent to which the algorithm can be parallelised. Since the full track reconstruction has both inherently parallel steps and parts requiring floating-point arithmetic, a hybrid CPU/FPGA hardware architecture might be the best solution. The algorithm is split into a parallel part executed in the FPGA and making use of look-up tables stored in SRAM, and a sequential part using floating-point arithmetic executed in the CPU.

All pattern recognition steps apart from the fit have been transformed to make use of instruction level parallelism and instruction pipelining on the FPGA. The SRAM with a large word length allows a parallel execution of LUT-based instructions. Furthermore, SRAM allows a fast random access, which is needed for some steps of the algorithm. The fit has not been transformed to an FPGA implementation, since floating-point operations are not very efficient on FPGAs. The benchmarking results show that there is no increase in the overall execution time resulting from executing the fit on a general-purpose processor.

The assumed hardware architecture for the ATLAS trigger is a distributed processor farm with a big central switch, connecting all computing nodes to all detector Read-Out Buffers (ROB). The number of required ports is reduced by the use of ROB-to-Switch Interfaces (RSI). For B-physics, one PCI-based accelerator card (FPGA co-processor) per computing node is added. This accelerator card, as shown in Figure 20, contains a PCI chip, an FPGA and SRAM. The SRAM is organised in 20 bit address and 320 bit word length. For LUT-based operations the large word length between FPGA and SRAM is important. The required bandwidth between the FPGA co-processor and the computing node CPU is well below the current PCI limits. The transformation of the track finding step into the FPGA accelerator is described in section 8.1. Section 8.2 describes

to it.

the subsequent pattern recognition steps in FPGAs and Table 3 summarises the reasons for the faster execution of the different steps of the algorithm.

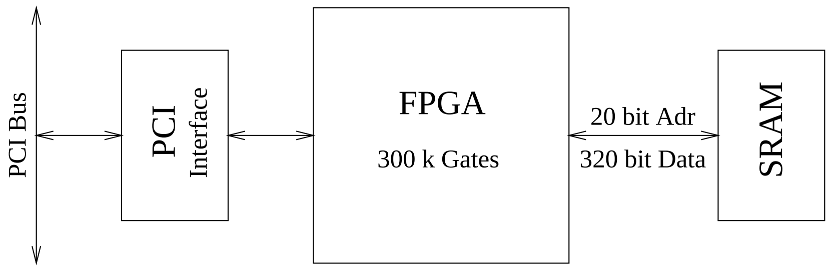


Figure 20: **FPGA accelerator board.** *Schematic view of the FPGA accelerator, which should exist in every computing node once.*

8.1 Initial Track Finding

Identical results are obtained for the initial track finding step for both the CPU and the FPGA implementations. However, the details of the implementation are quite different and are therefore described here. The CPU works sequentially and increments on average 130 histogram counters per hit, as shown in Figure 21(b). The LUT itself stores the addresses of the histogram counters.

In contrast, in a brute force implementation for FPGAs the LUT stores for each straw a bit pattern for all pre-defined roads. The bit pattern stores 1, if the straw belongs to that road, and 0 if the straw does not belong to that road. This would translate into a LUT with a 19 bit address and 96 000 bits of data[8]. This implementation would need over 50 GBit RAM and, furthermore, would be very slow.

Therefore, several optimisations are done:

1. **Symmetry** is used to reduce the size of the LUT. A 16-fold symmetry can be used in the barrel and a 192-fold symmetry can be used in the end-caps. This reduces the required address space of the LUT. However, this does not improve the execution speed, since the execution speed is connected with the word length of the SRAM and the corresponding number of histogram counters in the FPGA, see Figure 21.

2. The fact that a hit can only be part of a road which is spatially close to the hit can be used to reduce the execution time. This **geometrical consideration** is used with the concept of a “pseudo RoI”. A pseudo RoI is defined

[8]96 000 bits of data for barrel straws and 92 160 bits for end-cap straws.

as a sector of the TRT containing all the straws that must be considered when searching for tracks ($p_T > 0.5\,\mathrm{GeV}$) traversing a smaller search region. The definition of the pseudo RoI is illustrated diagrammatically in Figure 22. The relation of the size of the pseudo RoI and the search range is given by the requirement that for tracks with maximum curvature (0.5 GeV tracks) there is no loss of hits. This requirements guarantees that there are no track segment losses[9]. The size of the needed pseudo RoI defines the number of contributing roads per straw, which is 10 000.

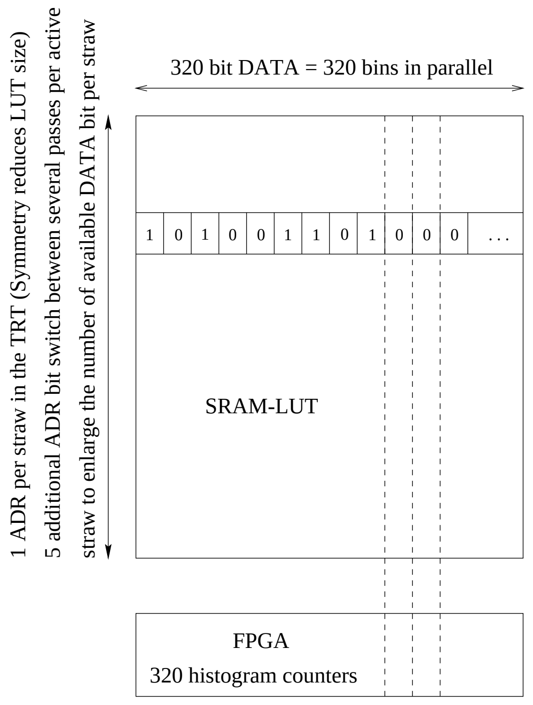
3. Only 320 out of the required 10 000 histogram counters (one per road) are in the FPGA and can be incremented per clock cycle. Therefore **multiple passes** per active straw are used. A 5 bit pass counter ($2^5 = 32$) in the address is incremented at each pass, see Figure 21(a). Therefore $320 \times 32 = 10\,240$ roads are available to each straw. Only after filling the histogram with all hits from the pseudo RoI (see Figure 22) can the histogram counters be read out and the next pass executed.

4. A further, optional optimisation step is the introduction of **small track segment losses** for low-p_T tracks. Allowing for an efficiency loss of a few percent for 0.5 GeV to 1.0 GeV tracks, the number of histogram counters to be possibly incremented shrinks from 10 000 to 5 000. Therefore 16 passes instead of 32 are sufficient, which results in an execution speed-up of factor 2. Usually this optimisation would be chosen for FPGAs (small losses in efficiency for some low-p_T tracks), but achieving exactly the same efficiency is also possible with 32 passes.

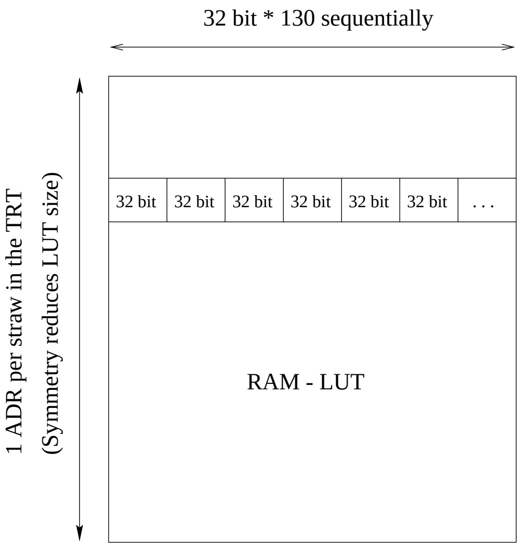
The increased speed of the FPGA implementation is due to several factors, as follows. Firstly, instead of sequentially incrementing 130 histogram counters per hit, all counters are incremented in only 16 passes. In other words, on average 8 histogram counters are incremented in parallel. Furthermore, random access to SRAM for the FPGA is faster than random access to SDRAM for the processor. The cache sizes currently available are not sufficient to allow effective use of the cache for the required LUT sizes. However exploitation of detector symmetry in the barrel will help.

In the FPGA case the execution time scales linearly with the number of predefined roads, independent of the road size. This means, doubling the number of

[9]Track segment losses occur when the required pseudo RoI for a given search region is reduced on purpose. The effect on the algorithm quality is a loss of short track segments for low-p_T tracks close to the search range borders. The effect on the algorithm execution time are a few percent reduction for a CPU implementation, but a huge reduction for an FPGA implementation.


320 bit DATA = 320 bins in parallel
1 ADR per straw in the TRT (Symmetry reduces LUT size)
5 additional ADR bit switch between several passes per active straw to enlarge the number of available DATA bit per straw
1 0 1 0 0 1 1 0 1 0 0 0 ...
SRAM-LUT
FPGA
320 histogram counters

(a) **LUT for FPGAs.** *For each pass per active straw potentially 320 histogram counters, which are directly connected with the SRAM data output, can be incremented.*


32 bit * 130 sequentially
1 ADR per straw in the TRT (Symmetry reduces LUT size)
32 bit 32 bit 32 bit 32 bit 32 bit 32 bit ...
RAM - LUT

(b) **LUT for general-purpose processor.** *For each active straw the corresponding histogram counters are incremented sequentially.*

Figure 21: **FPGA implementation versus C++ implementation.**

pre-defined p_T and η roads gives an increase by a factor of four in the execution time. For a general-purpose processor, however, the execution time depends on the number of predefined roads and the road size. An increase in the number of pre-defined p_T and η roads is usually accompanied by a corresponding reduction of the road size. This effect is used in implementations on general-purpose processors to give a better execution time scaling with the number of pre-defined roads. This fact makes a direct comparison problematic, since the general-purpose processor can still work efficiently with a fine-grained histogram, whereas the FPGA implementation works most efficiently with a coarse-grained histogram.

FPGAs can perform very fast a coarse-grained track search. Due to the different implementation of the track finding step in FPGAs, the speed-up in comparison to a CPU is the higher, the less road trajectories are searched for. Furthermore, the FPGA speed-up increases for a search for high-p_T tracks, because in this case the search range and the pseudo RoI can have similar size.

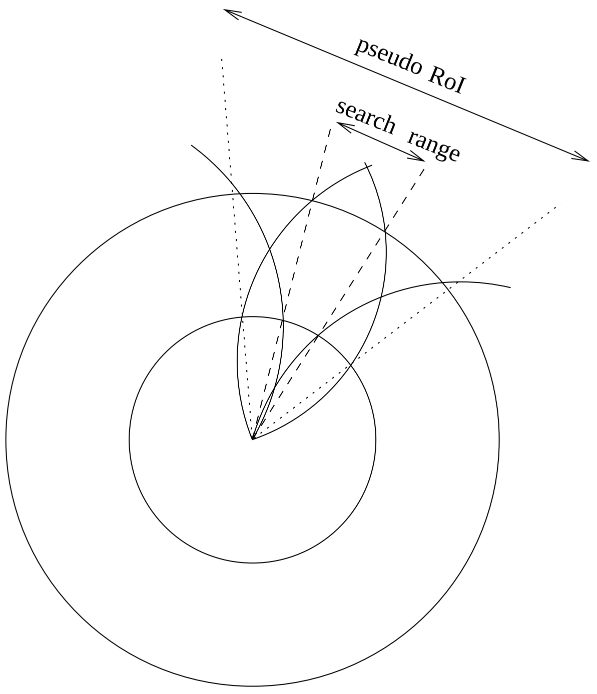


Figure 22: **Pseudo RoIs.** *To fill the histogram bins in the search range, all active straws from the pseudo RoI have to be considered, thus no track segment losses occur. Shown are 0.5 GeV tracks, which define the pseudo RoI for a given search range.*

8.2 Subsequent Pattern Recognition Steps

After the initial track finding, which is done in blocks of 320 histogram counters, the threshold is applied. This is performed in parallel for each block of 320 histogram counters, since all counters are available inside the FPGA.

The next step is the 2-D maximum finding. This is also done in parallel for all histogram counters inside the FPGA. Since there is always only a fraction of the entire histogram in the FPGA, the track reduction of the maximum finder is a bit worse than in the processor implementation. This affects the timing rather than the quality, since more tracks have to be eliminated at a later stage.

The track splitting step is quite similar to the processor implementation. It also utilises the two LUTs shown in Figure 7. One small difference is that only one track candidate can be considered per histogram counter. Having more than one track candidate in one histogram counter can only occur in the end-caps and is very unlikely, therefore this difference is negligible. The track splitting step results in the output of the hits belonging to the track candidates.

Finally, the 3-D maximum finder is applied for the histograms which lead to the track candidates. It is applied in a similar way to the 2-D maximum finder. The numbers of the histogram counters which are eliminated by this step are output, and they are removed from the list of track candidates before the transmission to the CPU for the fit.

9 Data Samples

For the evaluation of track reconstruction performance and of the electron identification capability, fully simulated $bB_d^0X \rightarrow \mu J/\psi(e^+e^-)K_s^0X'$, ($\mu$: $p_T > 6\,\mathrm{GeV}$, $|\eta| < 2.4$, e: $p_T > 1$ GeV) events with minimum bias pile-up (3.2 pile-up events on average) from tape Y00347 [8] have been used. The spatial distribution of the interaction point in both the longitudinal and transverse directions was simulated.

For the measurements of the execution time, track reconstruction efficiency and for studies of multiply reconstructed tracks and the number of reconstructed tracks, fully simulated $b\bar{b} \rightarrow \mu X$, ($\mu$: $p_T > 6\,\mathrm{GeV}$, $|\eta| < 2.4$) events with minimum bias pile-up at low luminosity from tape Y00347 have been used.

For the measurement of single track efficiencies, ϕ resolution, $1/p_T$ resolution and η resolution, samples of single μ^-, π^- and e^- with fixed p_T (1 GeV, 5 GeV, 20 GeV) were used. The samples were without pile-up.

The physics simulations are based on PYTHIA. The detector simulation uses the GEANT based program DICE.

10 Benchmarking and Algorithm Parameters

10.1 Algorithm Parameters

The C++ code contains parameters, which can be changed to allow the tuning of the algorithm for specific tasks. For example, an adaption of the algorithm for a RoI-guided TRT scan for high-p_T tracks can be achieved simply by loading a different set of parameters. The parameters shown in Table 1 are the default parameters which have been used to obtain all results in this paper.

10.2 Benchmarking

All the algorithm steps presented in this note have been implemented in C++ and can be run on general purpose processors. All steps apart from the final fit have also partially been implemented in VHDL to run on the FPGA processor Enable++ [9] or on a FPGA co-processor as described in section 8. Execution times are shown in Table 2. Times are given for implementations on a 300 MHz PC, a 300 MHz PC with a FPGA co-processor board and a 50 MHz FPGA system,

Parameter	**Barrel**	**End-caps**
threshold (=N_{thr})	14	14
high_threshold (=$N_{highthr}$)	16	16
isolation (=$N_{isolation}$)	9	8
LUT_phi	1200	1152
LUT_pT (LUT_pL)	80	80
first_hit (=$Layer_{firsthit}$)	50	-
etacut (=$Layer_{etacut}$)	9	-
road_min	4.5	-
road_max	8.5	-

Table 1: **Employed algorithm parameters for all results obtained in this paper.**

Enable++ [10]. The times measured for Enable++ are for the configuration described in [11], i.e. using two FPGA boards in the computing core, both containing 16 FPGAs. A successor of Enable++, ATLANTIS, which is being developed in Mannheim, needs only eight FPGAs to achieve similar performance as Enable++ with 32 FPGAs.

The measurements shown in Table 2 are for the algorithm steps up to, but excluding, the final fit. For comparison, the execution time for the initial TRT track search in the off-line pattern recognition program xKalman is 1100 ms. The xKalman measurement was made using a profiling utility.

The implementation of the algorithm on a processor with a FPGA co-processor gives a factor of 20 reduction in execution time compared with the PC implementation. The FPGA co-processor board executes the pattern recognition steps and the general-purpose processor executes the track fit. An execution time breakdown is shown in Table 3. The required bandwidth between co-processor board and CPU of 10 MByte/s is well below the current PCI limit. A further factor of five increase in speed is obtained with the ENABLE++ implementation. In both cases where FPGAs are added to the system, the LUT is stored in the SRAM of the FPGA board and the CPU memory is free for other use.

The execution times for all algorithm steps, including the final fit, are given separately for barrel and end-caps for the C++ implementation in Table 4.

The distribution of execution times for the algorithm running on a 300 MHz

Algorithm	**xKalman**	**LUT-based**	**LUT-based**	**LUT-based**
Processor	Pentium-II	Pentium-II	Pentium-II + FPGA-co-processor	FPGA-processor
Execution Time	1100 ms	214 ms	10.36 ms	1.62 ms

Table 2: **Benchmarking of different hardware and software.** *TRT full scan for $b\bar{b} \rightarrow \mu X$ events including pile-up at low luminosity. All measurements shown exclude the fit. Pentium-II processor with 300 MHz, FPGA-processor and FPGA co-processor with 50 MHz clock frequency.*

Pentium-II processor with a Linux operating system is shown in Figure 23(b) for B-physics events at low luminosity. The dependence of the execution time on the TRT straw occupancy is shown in Figure 23(a). To a good approximation the execution time scales linearly with occupancy, with a non-zero intercept value. The later is due mainly to the time taken for the application of the threshold cut to all histogram bins. At occupancies above about 10 %, the execution time is somewhat lower than would be predicted by a linear extrapolation. This can be explained as an effect of the caching of the LUT. At higher occupancies there is an increased probability that the required portion of the LUT will already have been loaded into the cache when processing a previous hit.

11 Single Track Performance

In this section, the reconstruction performance for single tracks is presented. The results obtained represent an idealisation of what can be expected under normal LHC operating conditions.

Muons have been used to study track parameter resolutions due to pattern recognition, multiple scattering and ionisation loss effects. Hadrons suffer from secondary interactions in addition. The effect of these interactions can be to stop the incident track, so that it may prove impossible to reconstruct the entire track. However, in the absence of secondary interactions, the distributions of its reconstructed parameters are similar to those of a muon. Electrons, finally, may lose a significant fraction of their energy through bremsstrahlung emission in the material in the Inner Detector.

Particles were generated at a range of discrete values of transverse momentum

Steps of the Algorithm	**Execution Time** Pentium-II	**Execution Time** FPGA co-processor	**Speed-up due to**
Initial Track Finding	131 ms	6.42 ms	instruction and data parallelism
Threshold Application	16 ms	0.02 ms	instruction level parallelism
2-D Maximum Finding	4 ms	0.16 ms	instruction level parallelism
Track Splitting	55 ms	3.62 ms	random access to huge, fast SRAM and instruction pipelining
3-D Maximum Finding	8 ms	0.14 ms	instruction level parallelism
Fit and Final Selection	7 ms	–	–
Full Algorithm	221 ms	10.36 ms	speed-up: **20**

Table 3: **Execution time comparison of the C++ implementation and the mixed FPGA co-processor/general-purpose processor implementation.** *TRT full scan for $b\bar{b} \rightarrow \mu X$ events including pile-up at low luminosity. Pentium-II processor with 300 MHz, FPGA co-processor with 50 MHz clock frequency. Latency: 10.4 ms FPGA processing + 7 ms CPU processing + 0.4 ms PCI transmission = 17.8 ms.*

p_T (1 GeV, 5 GeV and 20 GeV), in order to investigate the dependence of performance on p_T. Pions and electrons are key components in the B-physics trigger. In the current menus [12] electrons from 1 GeV and pions from 1.5 GeV have to be reconstructed efficiently. The 1 GeV and 5 GeV samples are used to illustrate the performance over roughly the range of p_T in B-physics events. The 20 GeV samples illustrate the performance for “straight” tracks, at a p_T above the range relevant to the B-physics trigger.

11.1 Track Reconstruction Efficiency

The reconstruction efficiencies for single $p_T = 1$ GeV, 5 GeV and 20 GeV muons, pions and electrons are shown in Figure 24.

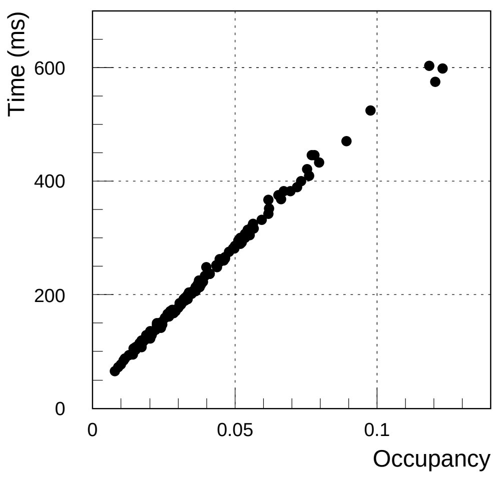
Steps of the Algorithm	**Barrel** 300 MHz PC	**End-caps** 300 MHz PC
Initial Track Finding	35 ms	96 ms
Threshold and 2-D Maximum Finding	7 ms	13 ms
Track Splitting	3 ms	52 ms
Threshold and 3-D Maximum Finding	–	8 ms
Final Selection and Fit	2 ms	5 ms
Full Algorithm	47 ms	174 ms

Table 4: **Execution time breakdown for the C++ implementation of the TRT full scan, for B-physics events including pile-up at low luminosity.**

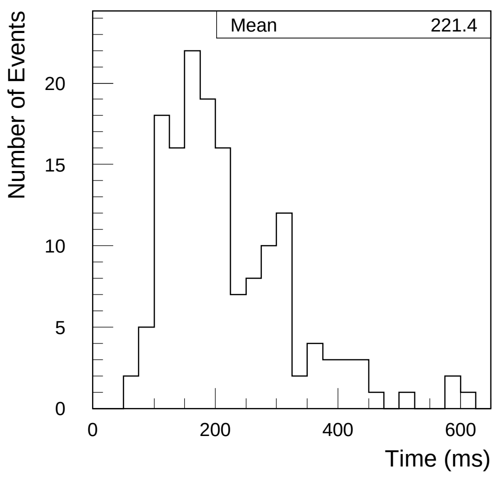
The efficiency for muons is close to 100 % except in the regions $\eta = 0$ and $\eta = 0.85$. The loss of efficiency near $\eta = 0$ is due to the fact that the signal readout from the TRT straws is divided at z = 0. The track search is performed separately in the two halves of the barrel. Since the interaction point has a spread in the z direction of σz = 5.6 cm, a small proportion of tracks will cross from one half of the barrel to the other. No η information is available for tracks reconstructed in this part and so the threshold on the number of hits cannot be lowered in this region. The threshold value chosen is a compromise between a low value for efficiency in this region and a higher value to reduce the number of fake and multiply reconstructed tracks and so increase algorithm speed.

There is also a loss of efficiency in the transition from barrel to end-cap near η = 0.85. Here, all tracks transit from barrel to end-cap and in the worst case, 50 % of the hits are in the barrel and 50 % of the hits are in the end-cap. However, after the reconstruction of the track segments, it is known whether a track segment is in the barrel-to-end-cap transition region, as the first hit in the barrel is in a layer < 10. A lower threshold on the number of hits is applied to tracks in the transition region. This recovers many tracks traversing from barrel to end-cap, but on the other hand the number of multiple tracks and fake tracks rises. This can be seen for pions in full events at low luminosity in Figure 31(b).

For 1 GeV pions, in addition to the loss in the transition regions, there is an



(a) **B-physics.** *Execution time dependency on straw occupancy. The average occupancy at low luminosity is around 4 %.*



(b) **B-physics.** *Distribution of execution time per event. The average execution time is 221 ms.*

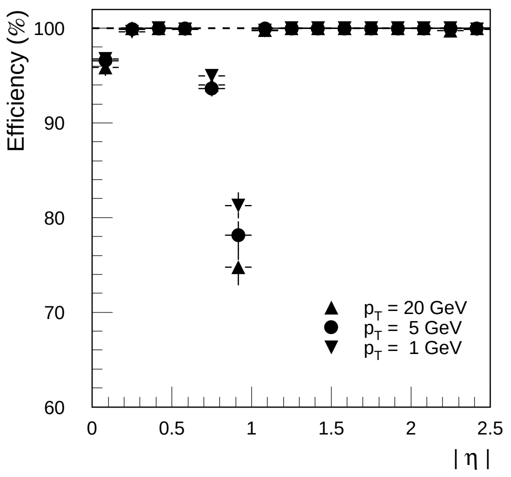
Figure 23: **Execution time in C++ for B-physics events at low luminosity on a 300 MHz Pentium-II processor under Linux.**

overall loss of efficiency due to interactions, especially around $\eta = 1.7$. This loss of efficiency corresponds to the distribution of material in the Inner Detector, as shown in Figure 25(a) [3]. The consequences for the TRT of this material are:

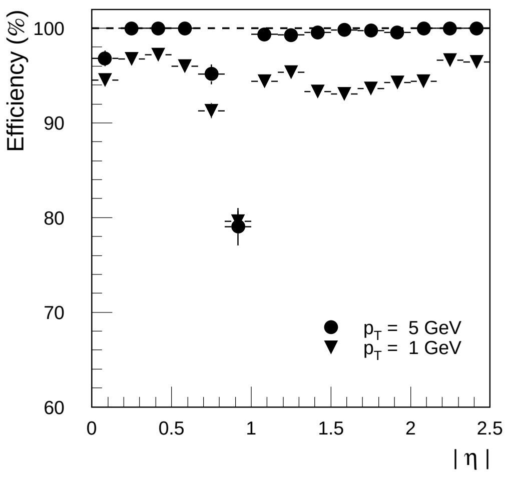
- Low momentum tracks undergo significant multiple scattering
- There is an increase in multiplicity from secondary particles
- Electrons suffer significant bremsstrahlung energy loss
- Photons have a significant conversion probability
- Absorption of hadrons, causing tracks to be lost

Figure 25(b) shows the pion interaction probability as a function of $|\eta|$. This rises from 8 % in the barrel to a peak of greater than 20 % around $|\eta| = 1.7$ and reproduces the shape of the distribution of the number of radiation lengths in Figure 25(a).

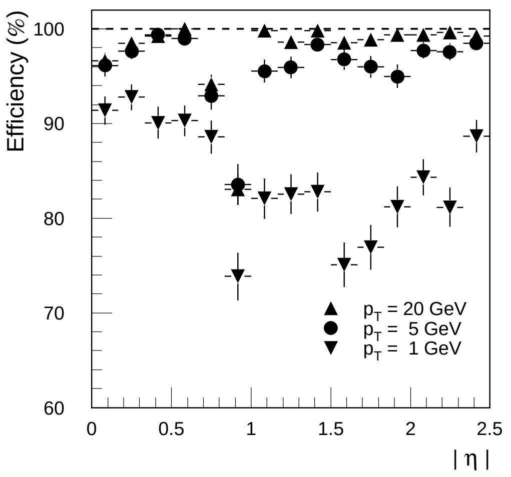
For 1 GeV electrons there is a big loss of efficiency at $|\eta| \approx 1.6$. This is due to a combination of the effects of energy loss due to bremsstrahlung and



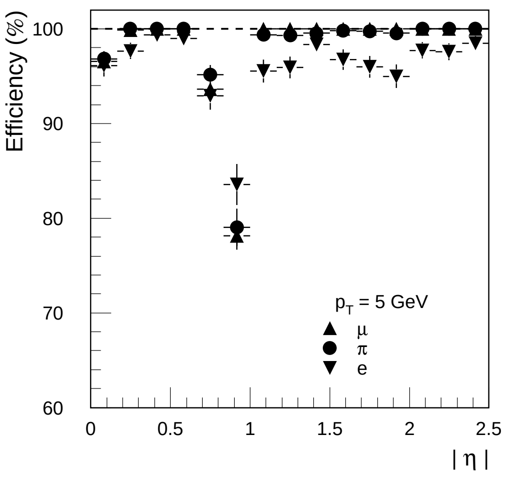
(a) **Muons.**



(b) **Pions.**



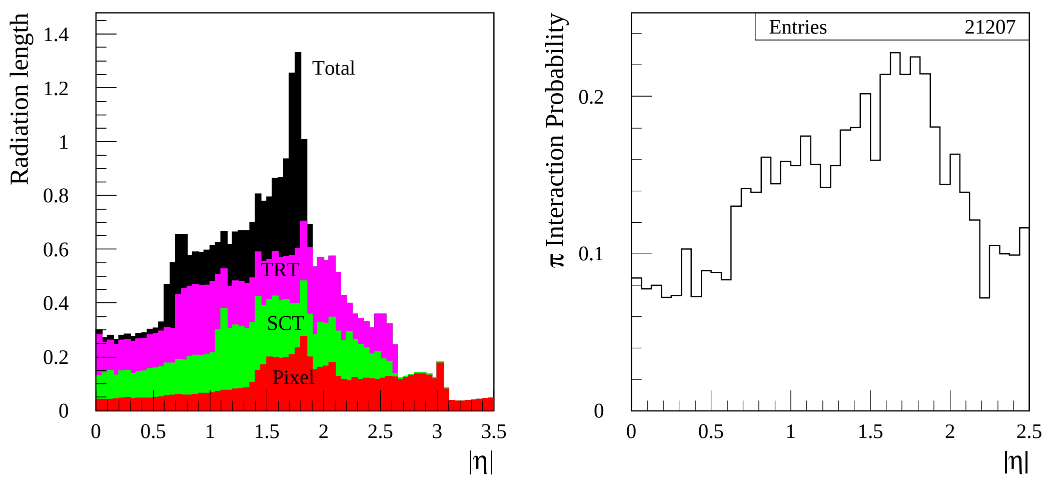
(c) **Electrons.**



(d) **5 GeV comparison.**

Figure 24: **Reconstruction efficiency for single particles with various momenta as a function of $|\eta|$.**

the definition of the LUT. As discussed in section 5.1, the LUT is constructed with a p_L threshold that is decreased in two steps with $|z|$. As a consequence the effective p_T threshold for the initial track finding jumps from 0.35 GeV to 0.5 GeV for tracks at $\eta \approx 1.6$, and from 0.4 GeV to 0.5 GeV at $\eta \approx 1.2$. Figure 24(c) demonstrates that the efficiency in these regions is much higher for 5 GeV electrons.



(a) **Radiation lengths.** *Cumulative distribution for number of radiation lengths for (a) pixels, (b) SCT, (c) TRT and (d) external services and patch-panels. Figure taken from [3].*

(b) **Pion interaction probability.** *Pion interaction probability as a function of $|\eta|$. Figure taken from [3].*

Figure 25: **Material in the Inner Detector.**

The efficiencies for reconstructing single tracks are shown in Figure 26 as a function of the p_T threshold applied. For muons there is a sharp threshold rise. For 1 GeV muons an overall efficiency of 99 % can be obtained for $p_T^{rec} > 0.7$ GeV. For 1 GeV pions an overall efficiency of 92 % can be obtained for $p_T^{rec} > 0.7$ GeV. The threshold for pions is less sharp due to hadronic interactions. The threshold is least sharp for 1 GeV electrons, due to the effect of bremsstrahlung energy loss. In order to obtain an efficiency of 75 % a 0.65 GeV threshold must be applied. This efficiency rises to 87 % if the threshold is reduced to 0.5 GeV.

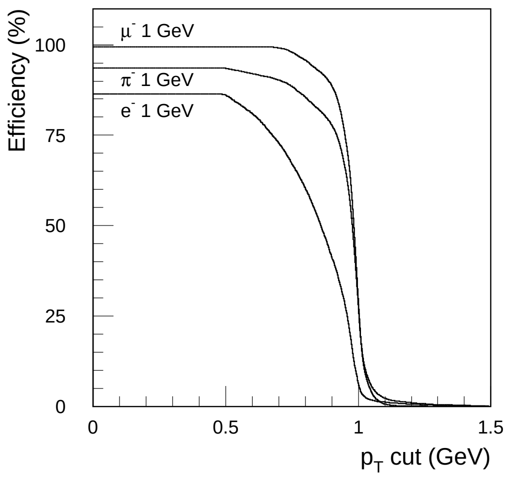


Figure 26: **Track reconstruction efficiency.** *Reconstruction efficiency for single tracks in events without pile-up as a function of the p_T threshold applied (after the fit). The minimum p_T for the LUT road definition is always 0.5 GeV, therefore the curves are flat from 0 GeV to 0.5 GeV.*

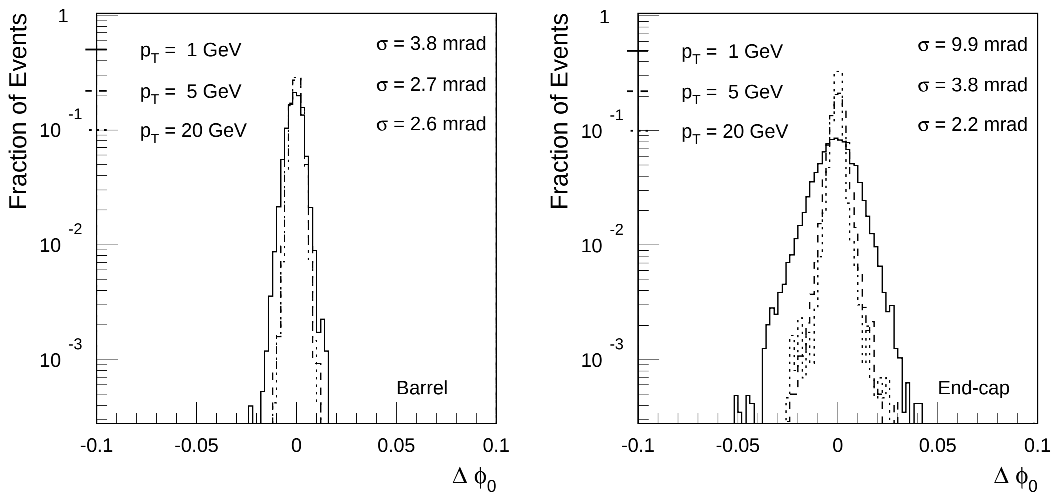
11.2 ϕ Resolution

As mentioned in section 2, the track candidates from the TRT define search regions for the SCT and pixels to refine the track measurement. Therefore, the ϕ resolution obtained from the TRT will be directly correlated to the CPU time required for the SCT and pixels. The probability of finding the same track in SCT and pixels is also reduced if the TRT ϕ resolution is degraded.

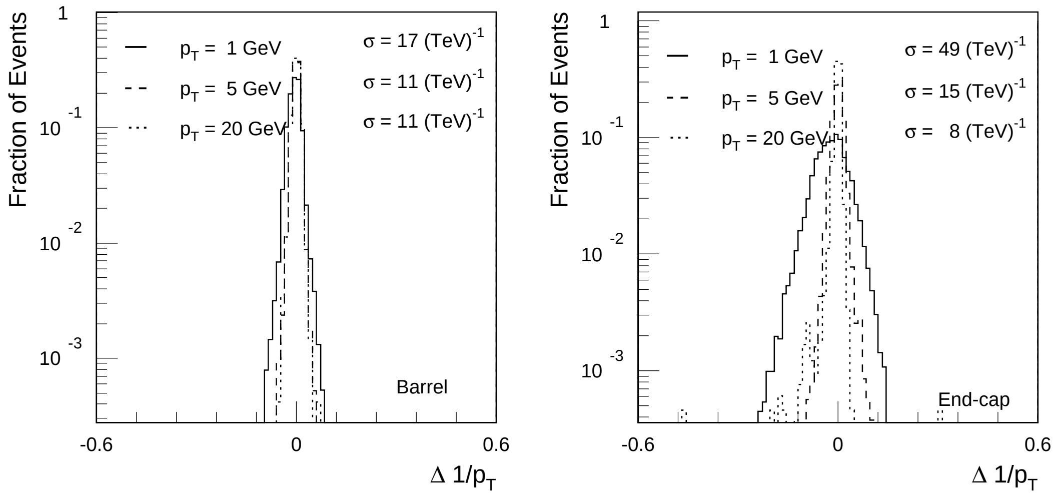
To obtain the ϕ resolution, the azimuthal angle of the generated track, ϕ_0^{gen} was compared to the azimuthal angle reconstructed by the TRT, ϕ_0^{rec} with the assumption that tracks come from the origin. Both ϕ_0^{gen} and ϕ_0^{rec} are measured at the origin.

The ϕ resolution is limited by physics processes, the detector measurement accuracy and pattern recognition errors. For muons, as shown in Figure 27(a), the ϕ resolution is dominated by different effects, depending on the particle momentum. For 1.0 GeV muons, multiple scattering and detector resolution contribute roughly equally to ϕ resolution. For 5 GeV and 20 GeV muons, the detector accuracy limits the ϕ resolution. In this case the resolution can be improved making use of the drift-time information. However, the p_T spectrum of tracks from B is peaked at low values.

For electrons, as shown in Figure 28(a), an asymmetric tail due to bremsstrahlung is observed with 20 % of electrons having $\Delta\phi > 0.02$. The direc-

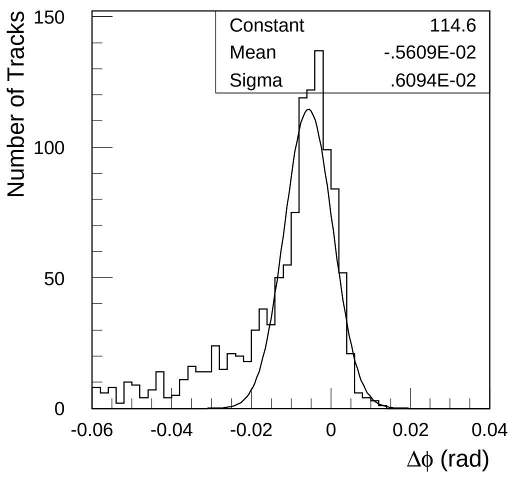


(a) **Single muon ϕ reconstruction.** *ϕ resolution, $\Delta\phi = \phi_0^{rec} - \phi_0^{gen}$, for muons in the region $|\eta| < 0.7$ (left) and $0.7 < |\eta| < 2.5$ (right).*

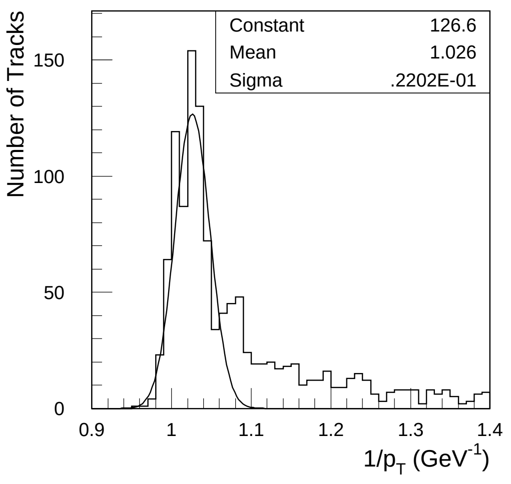


(b) **Single muon $1/p_T$ reconstruction.** *$1/p_T$ resolution, for muons in the region $|\eta| < 0.7$ (left) and $0.7 < |\eta| < 2.1$ (right).*

Figure 27: **Single muon ϕ and $1/p_T$ reconstruction.**



(a) ϕ **reconstruction.** *ϕ resolution, $\Delta\phi = \phi_0^{rec} - \phi_0^{gen}$, for 1.0 GeV electrons in the region $|\eta| < 0.7$.*



(b) $1/p_T$ **reconstruction.** *$1/p_T$ reconstruction for 1.0 GeV electrons in the region $|\eta| < 0.7$.*

Figure 28: **Single electron reconstruction.**

tion of this one-sided tail depends on the charge of the electron / positron and results from the assumption in the initial track finding that the tracks point to the vertex in the transverse plane. This results in a strong correlation between the ϕ and the $1/p_T$ reconstruction, as shown in Figure 28.

11.3 $1/p_T$ Resolution

The $1/p_T$ distributions for muons tracks are shown in Figure 27(b) for the barrel region and for the part of the end-cap region up to $|\eta| < 2.1$. In the end-caps, p_T is calculated from a knowledge of p_L and η. However no η measurement is possible for tracks at $|\eta| > 2.1$. In this region an η value of 2.5 is assigned, causing a degradation of the p_T resolution in this region.

As is the case for the ϕ_0 resolution, multiple scattering and detector resolution contribute approximately equally to the p_T resolution at $p_T = 1\,\text{GeV}$. At higher p_T, detector resolution dominates. The p_T resolution is important as it affects the value of the p_T cut that must be applied in order to achieve a given efficiency. Better resolution allows a higher cut to be applied which reduces the number of track candidates to be extrapolated to the SCT and pixel detectors. If necessary, the p_T resolution could be improved by performing a fit including drift-time information. However the benefits would have to be weighed against the cost of additional CPU time for the fit. For electrons, the effect of energy loss due to bremsstrahlung dominates, see Figure 28(b).

11.4 η Resolution

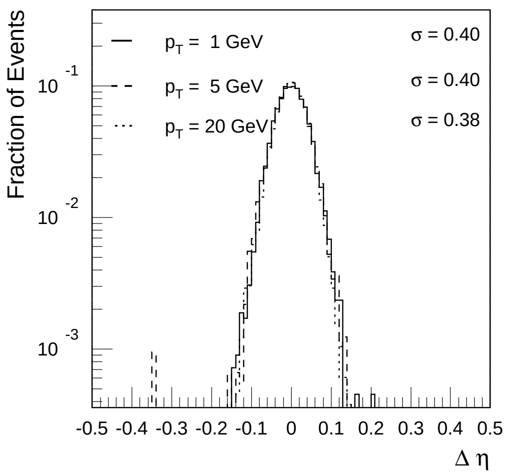


Figure 29: **Single muon η reconstruction.** *η resolution, for muons in the region 0.7 < $|\eta|$ < 2.1.*

The η resolution is shown in Figure 29 for the region of the TRT end-caps up to $|\eta| < 2.1$. The TRT makes no η measurement in the barrel[10] or in the region $|\eta| > 2.1$. The η resolution is important as it directly influences the calculation of p_T for the TRT end-caps.

12 B-Physics Performance

For a comparison of the LUT-based algorithm (from now on simply called LUT) with the reference algorithm xKalman, identical data sets were used for both algorithms. The data samples are described in section 9. The Figures of sections 12.1, 12.2 and 12.3 below show the LUT results, and the comparison with xKalman is made in the corresponding text. The cuts applied in this study are:

- $|\eta| < 2.5$ [11].
- only the particles with a point of origin within a 1 cm transverse distance from the interaction point were considered. This is due to the fact that the

[10]Some information can be obtained as described in section 4.4.
[11]In all Figures η stands for the η of the initial particle direction.

majority of B-hadrons will decay at radius of $< 1\,\text{cm}$ and that there is no trigger on neutral strange secondary hadrons.

12.1 Track Reconstruction Efficiency

The reconstructed track candidate is associated with the generated particle which contributed the largest number of hits. Multiply reconstructed tracks do not affect the track reconstruction efficiency. As a consequence, the efficiency is defined to be at maximum 1. Figure 30 shows the track reconstruction efficiency for pions and electrons in B-physics events as a function of p_T and η. For pions with $p_T^{gen} > 1$ GeV an efficiency of 90 % is reached in the barrel, whereas the efficiency in the end-cap is around 82%. The reconstruction efficiency for electrons from $J/\psi(e^+e^-)$ in events without pile-up is around 95 % in the barrel and 90 % in the end-caps[12]. These results compare favourably with xKalman using TRT and SCT/pixel.

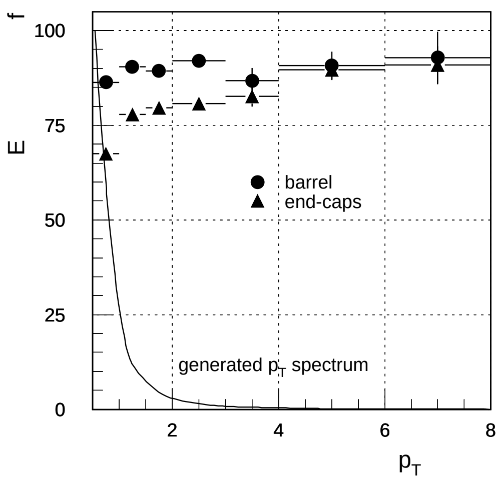
12.2 Multiple Reconstructed Tracks and Fake Tracks

Steps of the Algorithm	**Barrel**	**End-caps**
2-D Maximum Finding	10	5
Track Splitting	2	2
3-D Maximum Finding	–	1.2
Track Merging and Selection	1.4	1.4

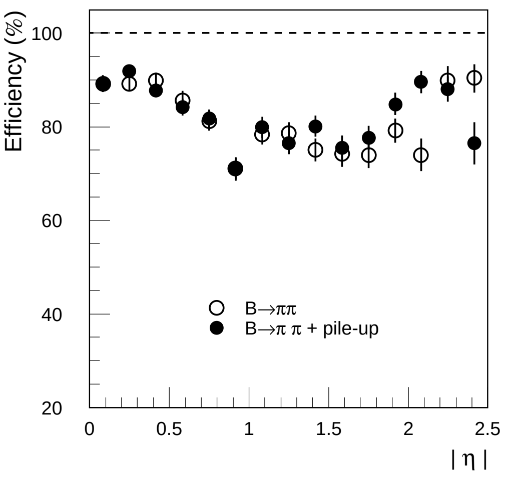
Table 5: **Reduction factors on the number of track candidates obtained by the different steps of the algorithm.**

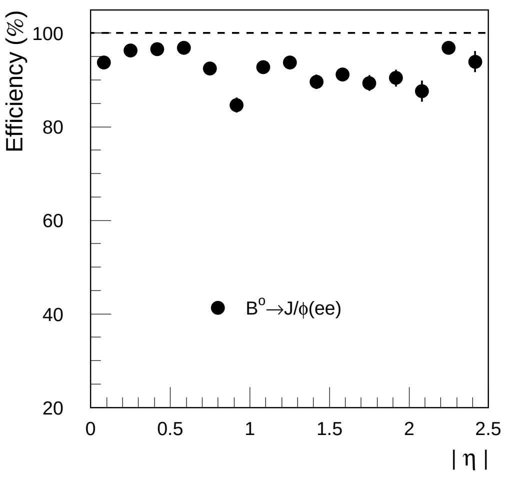
After the histogramming step and the application of the threshold on the number of hits in a bin, there are typically several hundred and up to several thousand initial track candidates in typical B-physics events including pile-up. This is several times the number of charged particles in the event, due to multiply reconstructed tracks. Therefore, the steps described above have been introduced

[12]For electrons a challenge is the matching of the corresponding track segments between the TRT and the SCT/pixels.



(a) **Track reconstruction efficiency.** *Efficiency for pions in the region $|\eta| < 2.5$ versus p_T^{gen} for $b\bar{b} \to \mu X$ events with pile-up. The inlay shows the p_T spectrum of all generated tracks in this η range.*





(b) **Track reconstruction efficiency.** *Efficiency for pions with $p_T^{gen} > 1.0$ GeV versus $|\eta|$.*

(c) **Track reconstruction efficiency.** *Efficiency for electrons with $p_T^{gen} > 1.0$ GeV versus $|\eta|$, without pile-up.*

Figure 30: **B-physics track reconstruction efficiency.**

to reduce this number to, ideally, one track candidate per generated track. The number of track candidates obtained per reconstructed pion with $p_T^{gen} > 1\,\text{GeV}$ is shown in Figure 31. This means, only the generated tracks which have been reconstructed at least once, are included in the Figure. Table 5 shows the power of the different steps of the algorithm in terms of rejecting multiple reconstructed tracks and fake tracks.

After the entire track reconstruction the number of track candidates with $p_T^{rec} \geq 0.5$ GeV is around a factor of two larger than the number of generated tracks with $p_T^{gen} \geq 0.5$ GeV. This effect has two causes:

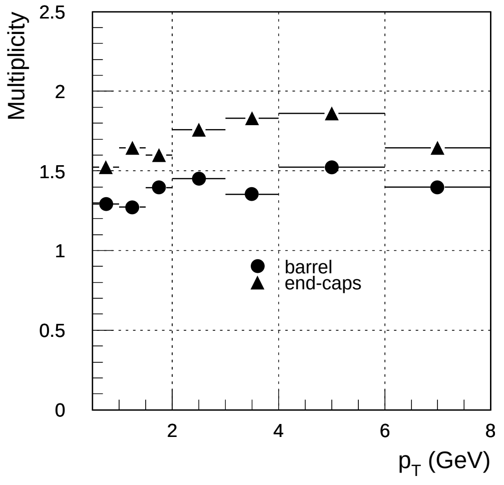
- **Multiply reconstructed tracks:** A fraction of the tracks are multiply reconstructed with slightly different hit combinations. The multiplicity of reconstructed tracks per generated pion with $p_T > 1.0$ GeV, that have been reconstructed at least once, is shown as a function of p_T and η in Figure 31. By definition, this track reconstruction multiplicity is ≥ 1. The fraction > 1 shows the average number of double tracks in this p_T range. For pions the average track reconstruction multiplicity is 1.5.

- **Fake tracks:** A 30 % contribution arises from particles with $p_T^{gen} < 0.5$ GeV, as shown in Figure 32(a). These particles are very numerous in events with pile-up, as shown in the inlay of Figure 30(a).

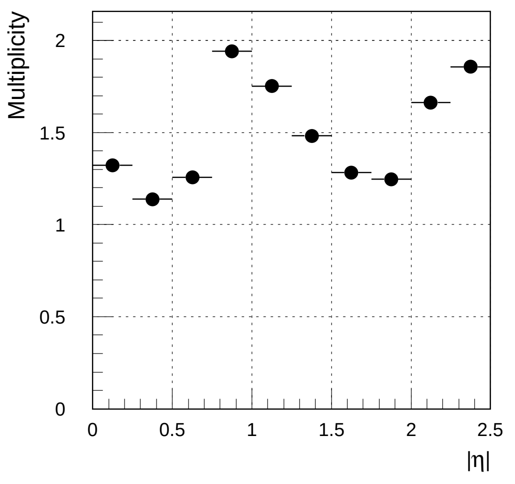
The classification into good, multiply-reconstructed, or fake tracks, is done by comparison of the reconstructed tracks with the simulated tracks at the hit level. A reconstructed track with hit contributions from different simulated tracks is assigned to the simulated track which contributed the most hits. If the reconstructed track is assigned to a simulated track with $p_T < 0.5$ GeV, it is labelled as a **fake track**. If it is assigned to a track with $p_T \geq 0.5$ GeV which is already reconstructed, it is a **multiply reconstructed track**.

To compare this result with xKalman, a completely analogous analysis was performed. Figures 31 and 32(a) show that the LUT algorithm (only using the TRT) reconstructs a factor of 2 more tracks than simulated. This value can be compared to that obtained with xKalman, which reconstructs a factor of 3 more tracks after the TRT step, and 20 % more tracks after using SCT/pixels and TRT.

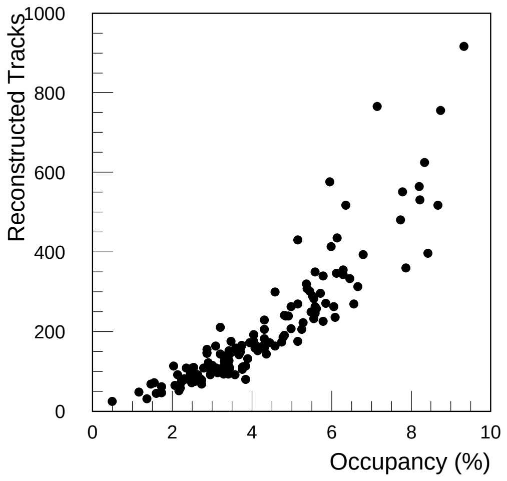
To reduce the total execution time of the Inner Detector (TRT and SCT/pixels) pattern recognition, the track reconstruction multiplicity of the described TRT algorithm can be lowered. This could be done with a track merging step, which is under study. Figure 31(b) shows that the track reconstruction multiplicity is especially high at low and high $|\eta|$ values in the end-caps. This is due to the definition of the LUT, which is made of sets of roads with one common



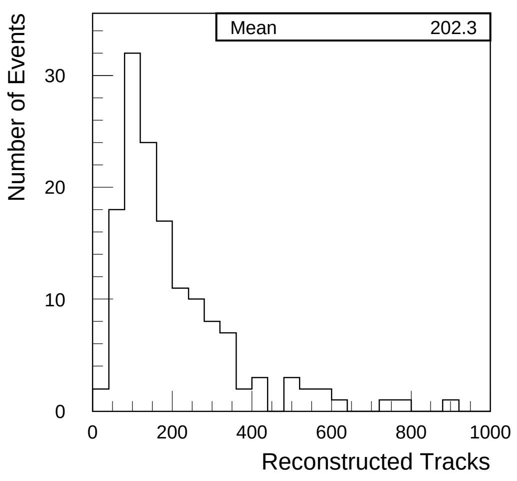
(a) **Multiple reconstructed tracks.** *Multiplicity of reconstructed tracks for pions in the region $|\eta| < 2.5$ versus p_T^{gen}.*



(b) **Multiple reconstructed tracks.** *Multiplicity of reconstructed tracks for pions with p_T^{gen} > 1.0 GeV versus $|\eta|$.*



(c) **All reconstructed tracks.** *The number of reconstructed tracks versus the TRT straw occupancy for B-physics events at low luminosity.*

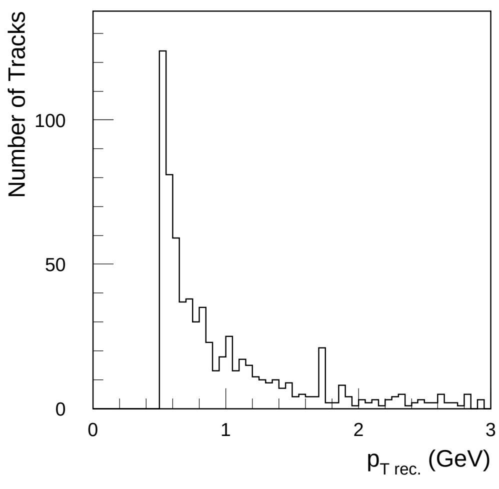


(d) **All reconstructed tracks.** *The distribution of the number of reconstructed tracks for B-physics events at low luminosity.*

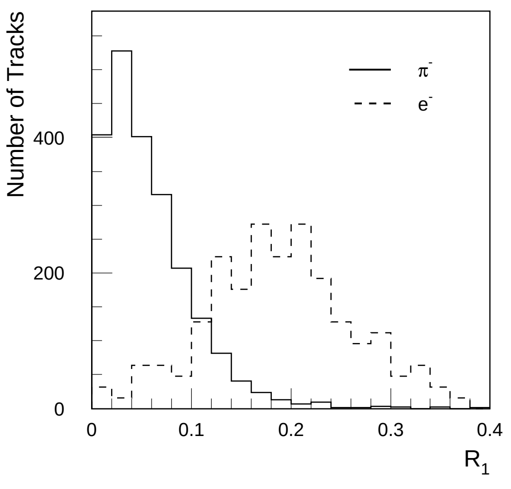
Figure 31: **Track reconstruction multiplicity and total number of reconstructed tracks.** For $b\bar{b} \to \mu X$ events with minimum bias pile-up.

intersection point approximately in the middle of the end-caps. The maximum finder works well for reconstructed tracks at $|\eta| = 1.7$, but does not eliminate tracks well at low and high $|\eta|$.

An important parameter for the overall B-physics trigger execution time is the number of tracks which have to be followed into the pixel and SCT detector. Figure 31(c) shows the dependency of the number of all reconstructed tracks over $p_T^{rec} = 0.5$ GeV on the occupancy, and Figure 31(d) shows the distribution of the number of reconstructed tracks, for B-physics events at low luminosity. Typical B-physics events with pile-up at low luminosity have around 90 tracks with $p_T > 0.5$ GeV in the TRT, most of which are pions. Since most of the tracks are identified as non-electrons below 1.5 GeV, they are not followed into the Precision Tracker (SCT and pixels), but still a large number of tracks remain. It is also possible to eliminate fake tracks and multiply reconstructed tracks (often with bad track parameter measurement) by the failure to find a prolongation in the Precision Tracker. However this is not the optimum solution, since it increases the computation needed and the data volume to be transmitted.



(a) **Reconstruction of fake tracks.** *The distribution of reconstructed p_T for tracks ($p_T^{rec} > 0.5\,GeV$) where the majority of hits are from a particle with $p_T < 0.5\,GeV$.*



(b) **Electron identification.** *Histograms of the ratio R_1 of transition radiation hits to TRT hits for reconstructed pions and electrons, integrated over $|\eta| < 0.7$ and $p_T^{gen} > 0.5$ GeV.*

Figure 32: **"Fake" tracks and electron identification capability.**

12.3 Electron Identification

The electron identification capability of the TRT is used to select electron tracks, for example in the trigger for $B \to J/\psi(e^+e^-)$. Typically a track is categorised as an electron if it contains a fraction of transition radiation hits, R_1, $\geq 10\,\%$. It is, therefore, important that hits are correctly assigned to tracks. In particular, an erroneous mixing of electron and pion track segments will degrade the electron identification capability. Figure 32(b) illustrates the electron identification capability of the LUT algorithm. Distributions are shown of the fraction of transition radiation hits on tracks due to electrons and pions. The identification probability for electrons with $R_1 > 0.1$ is $90\,\%$, with a rejection factor against hadrons of 6.7. This result is comparable to the result obtained with xKalman.

13 Conclusions and Outlook

It has been shown that a look-up table based algorithm provides a fast TRT full-scan implementation on general purpose processors. A considerable further increase in speed can be obtained by implementing time-critical steps on FPGAs. The algorithm presented here is well suited to such an implementation.

The track reconstruction efficiency obtained for B-physics events, with and without pile-up, is comparable with that of the initial track search in the off-line reconstruction program xKalman.

In the B-physics trigger, the tracks reconstructed in the TRT are extrapolated inwards to define search regions in the SCT and pixels detectors. The number of extrapolations to be made can be minimised by merging TRT track segments that have been split, e.g. in the barrel/end-cap transition region. This is an area of work which is on-going. Studies are under-way to determine the overall B-trigger performance of this algorithm in conjunction with various SCT and pixel reconstruction algorithms. The results of these studies will be reported in a separate ATLAS note.

References

[1] *ATLAS Trigger Performance Status Report*, ATLAS Trigger Performance Community, CERN/LHCC 98-15, 30 June 1998.

[2] *ATLAS Level-2 Trigger Demonstrator A Activity Report Part 1: Overview and Summary*, A. Kugel et al, ATLAS DAQ-note-085, 26 March 1998.

[3] *ATLAS Inner Detector Technical Design Report*, ATLAS Inner Detector Community, CERN/LHCC 97-16, 30 April 1997.

[4] *Description of Global Pattern Recognition Program (xKalman)*, I. Gavrilenko, ATLAS INDET-note-165, April 1996.

[5] *The Data Analysis BriefBook*, R. K. Bock, W. Krischer, ISBN 3-540-64119-X, Springer Verlag, 1998

[6] `http://www.cern.ch/Atlas/project/LVL2testbed/www/`

[7] *Object-Oriented Design of the Feature Extraction System for LVL2.* André dos Anjos, Reiner Hauser, Matthias Sessler and Stefan Tapprogge, Testbed Technical Note 20, 1998

[8] `http://wwwinfo.cern.ch/ ~ msmizans/production/tapelist.html`

[9] *ATLAS Level-2 Trigger Demonstrator A Activity Report Part 2: Demonstrator Results*, A. Kugel et al, ATLAS DAQ-note-084, 26 March 1998.

[10] *A Hybrid Approach for the ATLAS Level-2 Trigger*, A. Kugel et al, Proceedings of the Third Workshop on Electronics for LHC Experiments, London, CERN/LHCC/97-60, 21 October 1997.

[11] *ATLAS Level-2 Trigger Demonstrator A Activity Report Part 3: Paper Model*, A. Kugel et al, ATLAS DAQ-note-101, 2 June 1998.

[12] *ATLAS Trigger Menus*, J. Baines et al, ATLAS DAQ-note-121, 26 June 1998.