A Linear Algorithm to Filter and Group 3D Hits of the Discrete Tracking Layers

H.Drevermann[1], D.Kuhn[2], and B.S.Nilsson[3]

Abstract

For 3D hits from the pixel detector and the silicon strip detector a method is proposed to discriminate hits of the physics event from background hits. The accepted hits can be grouped, each group containing practically all hits of one or a few single tracks. The method leads to a code with a short execution time, which inreases only linearily with the total number of hits.

[1]CERN, CH-1211 Geneva 23
[2]Inst. für Experimentalphysik der Universität Innsbruck, A-6020 Innsbruck,
supported by grant of Austrian Federal Ministry of Science and Transport
[3]Niels Bohr Institute, DK-2100 Copenhagen

1 Introduction

When ATLAS will run with full luminosity a typical LHC event will contain the hits of the physics event, which triggered the recording and the hits of a few tens of minimum bias events and random noise. The events produce also delta electrons etc.. Here we are concerned with 3 dimensional hits sampled in the two innermost parts of the ATLAS detector, namely the pixel detector and the silicon strip detector, called silicon detectors in the following. In the case of the silicon strip detector (SCT) there will be furthermore ghost hits, if several charged particle pass through one module.

For visual analysis of these events it is of interest to separate the "good" hits belonging to sufficiently high p_t tracks of the triggering event from the "bad" hits, which consist of hits from low p_t tracks of the triggering event, of ghost hits, of all hits coming from the minimum bias events, of delta rays and noise etc.. We will propose in the following a method to solve this task despite the very high number of bad hits in the silicon detectors.

2 Filtering Method

A method of separation must be based on the difference between the features of the good and bad hits. Most of the good hits lie on tracks of rather high p_t pointing to one primary vertex. Most of the bad hits lie on tracks of rather low p_t pointing to several primary vertices, which differ mainly by their Z coordinate. Noise and ghost hits etc. do not lie on tracks.

These different features can be expressed more formally: a good hit must be contained in at least one set of N hits falling into a restricted interval of the polar angles ϕ and θ, which are calculated in a coordinate system with the primary vertex of the physics event at its origin. All hits not fulfilling this criterion are assumed to be bad.

The interval size $d\phi * d\theta$ depends on the measuring accuracy and in the case of ϕ also on the minimum p_t value to be accepted. The minimum number N of hits required in such a θ, ϕ cone depends on the detection efficiency of the silicon detectors.

The most simple solution to separate good from bad hits is to exclude a hit if its minimum θ, ϕ distance to all other hits is above a certain value. For M hits, this leads to M*(M-1)/2 combinations to be checked, i. e. to a computing time consumption which increases quadratically with the total number of hits.

Here we present another method, for which computing time increases only linearily with the number of hits. This linear behaviour is achieved by generating a ϕ/θ histogram of all hits in one loop only. A hit is then accepted as good, if it falls into a histogram bin with at least N entries.

In order to avoid problems due to rounding errors, each hit is counted in all eight neighbouring bins as well. Bins containing at least one genuine hit will be called

"direct bins", neighbouring bins without a genuine hit will be called "indirect bins".

For illustration a small section of such a histogram is shown in figure 1a presenting the hits and the resulting bin contents. Direct bins are coloured white, indirect bins black.

For filtering the following parameters were used:

$d\phi = 2$ deg

$d\theta = 0.18$ deg

minimum number of layers required: L=4

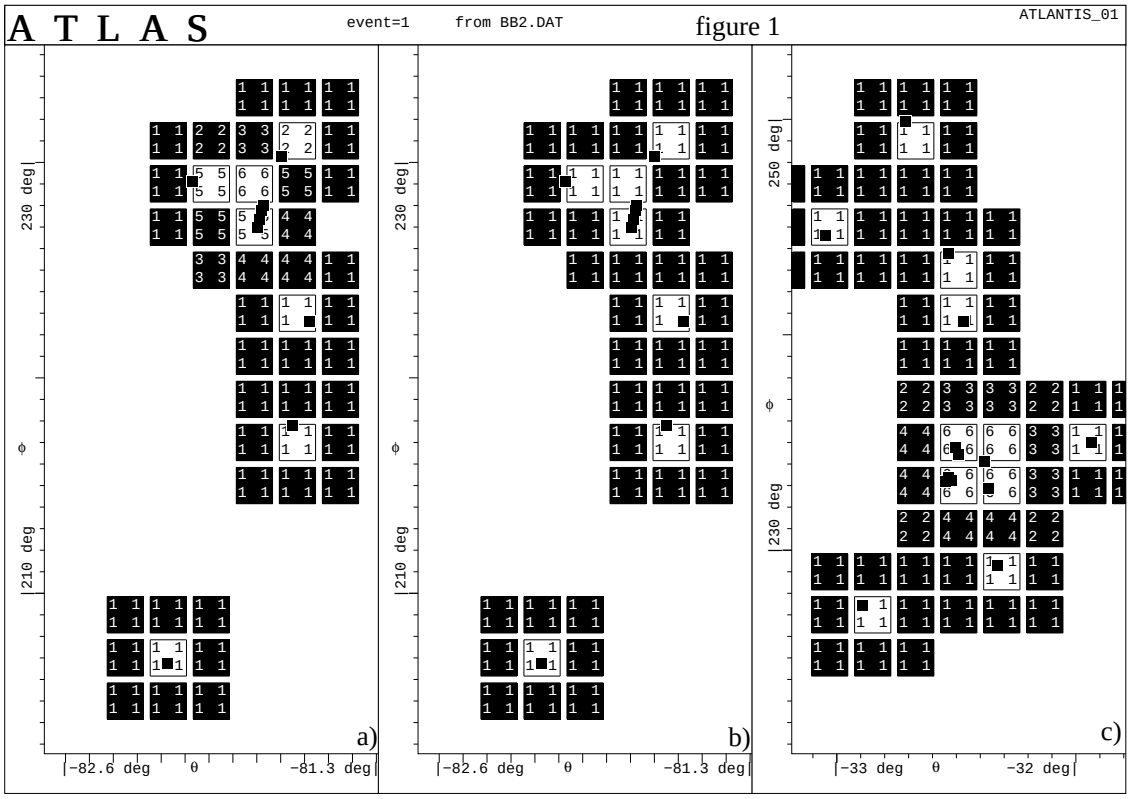

Figure 1: **Standard Projections.** (a) Small sections of a ϕ/θ histogram and (b), (c) of ϕ/θ layer histograms of a simulated LHC event. Direct bins are white, indirect bins are black. Superimposed are the hits (small black squares) by which the histogram section was filled.

As a further refinement of the method we use the fact that the hits of a genuine high p_t track should not only lie in a given θ, ϕ interval but originate from at least L different layers. Otherwise it may happen, that a track is faked by hits which lie on only few layers, which is the case of the four close hits in figure 1a.

Therefore, instead of registering the number of hits falling into a bin, we register the number of different activated layers. This is shown in the "layer histogram" in figure 1b, where the four close hits lie on the same layer and therefore are to be discarded.

Figure 1c gives another section of the layer histogram which contains a group of seven close hits on 6 different layers. Six of these hits belong to a simulated track.

All seven hits will be accepted as good, whereas the other hits on the picture will be flagged as bad.

This method, which was applied to high p_t tracks, can easily be modified to work also in other momentum intervals.

3 Grouping of Hits

Figure 2a shows a third section of the layer histogram. Requiring L=4 for good hits, four clusters of good hits and many bad hits are recognized. The method described so far to filter hits may be further extended by grouping those hits which fall into the same cluster in the histogram.

For this purpose, we associate a unique cluster number to all direct bins which touch each other. This is shown in figure 2b, where the four clusters got the numbers 13, 35, 9 and 7. In this way, groups of hits may be identified and lists be formed, so that for visual analysis one may selectively work with one group.

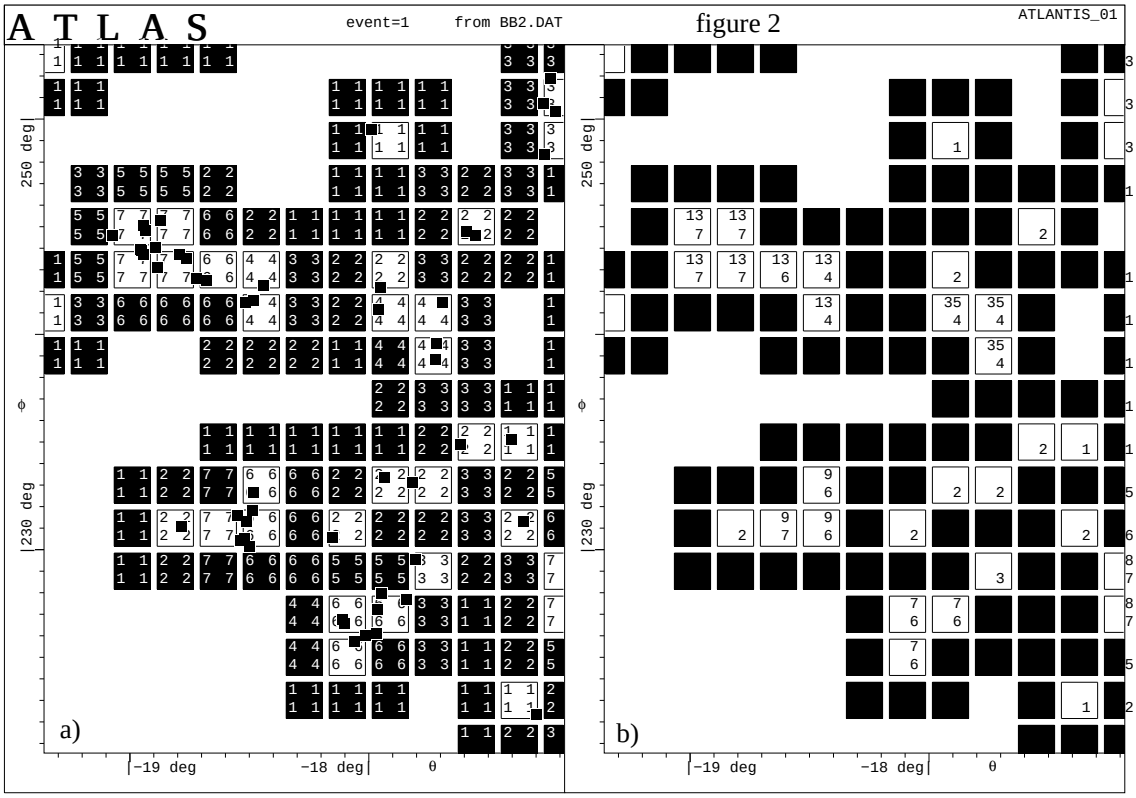

Figure 2: **Methods of Representing 3D Data.** (a) A small section of a layer histogram with background hits and hits of five tracks, (b) the same section where all direct cells with more than 3 entries are clustered. In the clustered bins the cluster number is printed on top of the number of entries.

In order to find and number all histogram clusters of bins with at least L activated layers, two methods are available:

- Either one loops over all histogram bins, which is too time consuming in our case and therefore not used,
- or one finds a starting point of the clustering algorithm by looping over all hits, which means in detail:
  - If a hit falls into a bin not yet treated, it gets a new cluster number and the histogram clustering starts from the corresponding bin in the layer histogram,
  - otherwise it gets the cluster number already allocated to this bin.

4 Event Display

In the following we will demonstrate, how filtering and grouping acts on hits of an simulated LHC event containing a b jet. Figure 3 shows the hits of a section of a simulated LHC event containing a jet in a ρ/Z projection (3a). One sees that many primary vertices at different Z positions exist. No tracks are recognized. Even when showing the filtered hits only, which are seen in a ρ/Z and a Y/X projection in figures 3b and 3c, no tracks are recognized, as in these representations for the human eye grouping of hits into layers dominates over grouping into tracks.

These projections are therefore not suitable to judge the efficiency of filtering. Instead we use the V-plot representation (developped originally for the ALEPH experiment), which improves track recognition considerably.

In the V-plot a 3D hit is represented by two points drawn left and rigth to its ϕ/θ position [1]. Helix segments near the centre are represented by a V pattern. Track assignment of the hits is not required to generate the V-plot. A V-plot of the filtered hits is shown in figure 3d, where in most cases the V-pattern is clearly recognised.

In the simulated LHC event it is know which hits belong to which tracks of the hadronic jet. Figure 4 shows V-plots of the four combinations:

- jet hits passing the filter (4a)
- background hits passing the filter (4b)
- jet hits rejected by the filter (4c)
- background hits rejected by the filter (4d).

Each hit belongs to only one of these combinations. If the filtering delivers exactly the jet hits, the two combinations 4a and 4d should contain all hits while the other two, 4b and 4c, should be empty. If not, the combination 4c should contain

- wrongly rejected jet hits and
- jet hits, which do not lie on a track (e.g.hits from delta electrons).

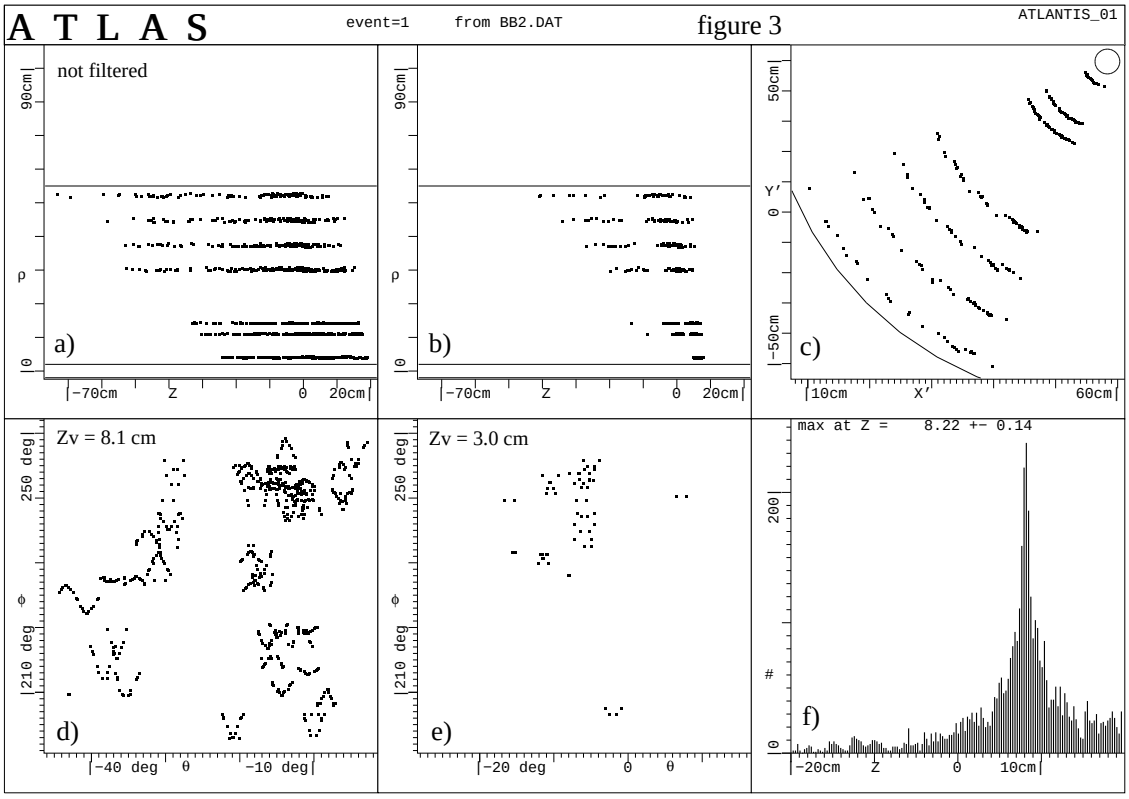

Figure 3: **Check of Tracks with the V-plot.** Region of a simulated LHC event containing a jet: (a) ρ/Z projection (b)ρ/Z projection of filtered hits, (c) Y/X projection of filtered hits, (d) V-plot of filtered hits with vertex position at 8.1 cm, (e) V-plot of filtered hits with vertex position wrongly assigned to 3.0 cm, (f) number of passing hits as a function of assumed vertex position Z.

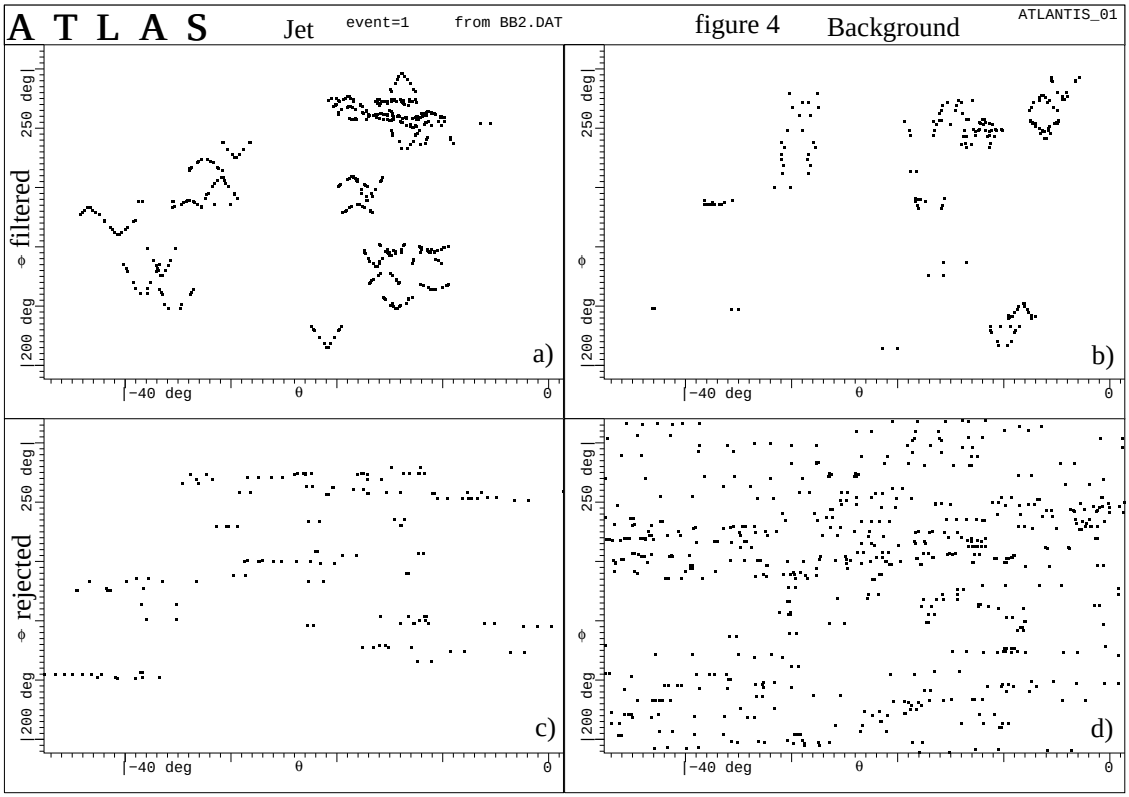

Figure 4: **Simulation of an Event with Very High Multiplicity**
A region of a V-plot for four exclusive combinations of hits, which are flagged as hits belonging to the jet or to the background. (a) Jet hits passing the filter, (b) background hits passing the filter, (c) jet hits rejected by the filter, (d) background hits rejected by the filter

Combination 4b contains

- background hits, which the filtering failed to reject, and
- jet hits, which were not flagged correctly.

(On the right side of figure 4b, five V-patterns are recognized, which means that there are five tracks which can be extrapolated back to the primary vertex. Tracks passing not too close to the vertex yield a deformed V- pattern, which is not the case here. It is therefore highly probable that these five tracks belong to the jet, but their hits were not flagged as jet hits.)

Two examples of hit grouping are demonstrated in figure 5. All hits belonging to a selected track (track 32 in this example), including the hits of delta rays created by it, are shown in figure 5 as V-plot (a), in a ϕ/ρ projection (b) and a ρ/Z projection (d). Only the hits drawn as solid squares passed the filter and were indeed assigned to a single cluster containing no other track in this case. The straight lines represent the corresponding tracks from the MC truth. As seen from the V pattern, these filtered hits are the good hits of the track; the rejected ones (open squares) are probably the hits of a secondary delta electron.

As a second example, a group was selected which contains more than one simulated track. The hits belonging to the group are shown in the same projections as for the first example (see figures 5d, e, g.). Figure 5f presents figure 5e in compressed form, where the tracks are easily recognized even without the suggestive lines. In this case some bad hits passed the filter and the grouping could not separate the three tracks in this group.

5 Finding the Vertex Position along the Beam Line

The filtering method (as well as the V-plot, which is a special form of a ϕ/θ projection) relies heavily on a precise determination of θ, which in turn implies the knowledge of the Z position of the vertex. If filtering and grouping would be applied to ease subsequent track following, a method is required to find the Z position of the vertex before applying the filter and not as the result of track following.

Due to the small beam diameter it is only the Z position of the vertex which has to be found. In the following we will describe a method to find the Z position of the vertex, based on the fact, that at the correct vertex position a maximum number of hits passes the filter.

This may be seen by comparing figure 3d, where the nominal vertex position as known from the simulation was used, to figure 3e, where a different, arbitrarily selected Z position was used. We recognize that only few hits pass the filter in the latter case. By applying the filtering process at subsequent Z positions and counting the number of hits N_h passing the filter, we get for this LHC event the histogram seen in figure 3f, where N_h is shown as a function of Z. It shows a clear peak at the

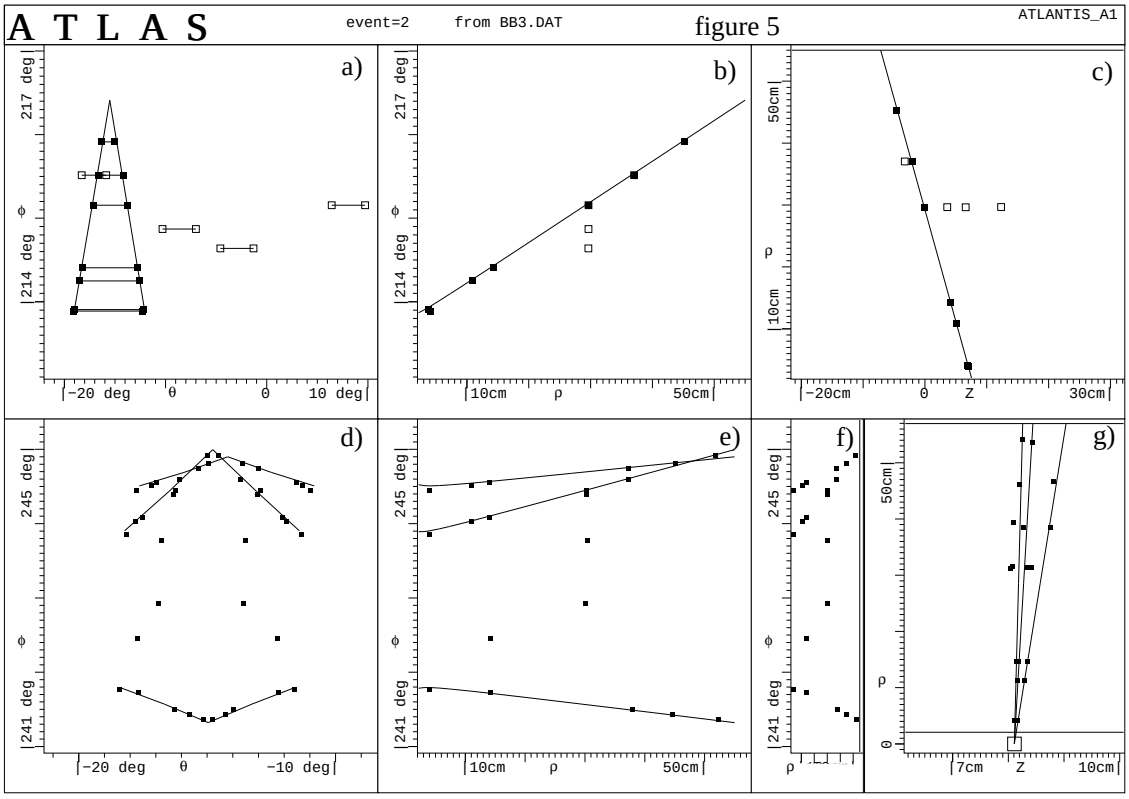

Figure 5: **Hits and tracks from MC truth.**
(a), (b), (c) Example of hits created by a track of the jet. Solid squares: hits passing the filter, open squares: rejected hits, (a) V-plot, (b) ϕ/ρ projection, (c) ρ/Z projection, (c), (e), (f), (g): all hits belonging to the same group, (d) V-plot, (e) ϕ/ρ projection, (f) ϕ/ρ projection compressed without tracks

bin between 8.01 and 8.15 cm, which contains the nominal value of 8.1 cm, known from the simulation.

We propose to investigate if this method is suited to find the Z position of the vertex for other events, which may even have a different topology. The ϕ bin size may be an appropriate tool to adjust the strength of the filter to different event topologies.

6 Time Consumption

The time to filter and group hits by the two dimensional layer histogram depends linearily on the number of hits. It was measured to be 14 msec + 11 msec/1000 hits on an ALPHA workstation DEC 3000 model 300, which corresponds to 15 old CERN units. The constant time of 14 msec was needed to set all bins of the histogram to zero. The corresponding FORTRAN code is found in appendix A. For comparison, on the same workstation a simple quadratic loop in which the number of pairs of hits with $|dx| < D$ and $|dy| < D$ is determined, needs 640 msec for 1000 hits (see appendix 2 for the code).

7 Outlook

It will be interesting to see to what extent these methods, which were tested with one MC event only, can be geneally applied to other events. We believe that they can be applied also to data from the endcaps, as ϕ and θ, which are the basic variables here, are defined for both barrel and endcap.

8 Conclusions

For visual analysis the filtering of hits based on a layer histogram, studied on a single MC event, yields an efficient separation of good and bad hits, so that pictures of events filtered in this way can be more easily interpreted. The hit grouping delivers furthermore groups of hits which are strongly correlated to the good hits of the simulated tracks. This further facilitates visual analysis of the results of simulation and pattern recognition etc.. It might be of interest to investigate the application of these methods to other fields of online and offline analysis, especially track following.

9 Acknowledgements

We aknowledge the help of A. Poppleton, from whom we got the data in an easily readable form.

References

[1] ‘ Event display: can we see what we want to see? ’
H. Drevermann, D. Kuhn and B.S. Nilsson,
Proc. of 1992 CERN School of Computing, L’Aquila, Italy, CERN Yellow Report 93-03.

[2] ‘Skewplot - a Momentum Sensitive Visualisation of Two Dimensional Hits from Very high Multiplicity Events’
H. Drevermann, D. Kuhn and B.S. Nilsson,
Proc. of HEPVIS96 Workshop on Visualization in High Energy Physics, CERN, September 96, CERN Yellow Report 97-01.

APPENDIX 1

C FORTRAN PROGRAM for filtering and grouping.

```
DIMENSION IFI(10000),ITE(10000)   ! store φ,θ bin position
COMMON /HISTOGRAM/ MNL,IGROUP,NLa,NLb,NSD
BYTE NLa(0:179,0:999),NLb(0:179,0:999) ! histogram bins
BYTE NSD(0:127)                   ! sum of binary digits
DATA MNL/4/
DATA NSD(0:127) /
          & 0, 1, 1, 2, 1, 2, 2, 3,       ! 0 - 7
          & 1, 2, 2, 3, 2, 3, 3, 4,       ! 8 - 15
          & 1, 2, 2, 3, 2, 3, 3, 4,       ! 16 - 23
          & 2, 3, 3, 4, 3, 4, 4, 5,       ! 24 - 31
          & 1, 2, 2, 3, 2, 3, 3, 4,       ! 32 - 39
          & 2, 3, 3, 4, 3, 4, 4, 5,       ! 40 - 47
          & 2, 3, 3, 4, 3, 4, 4, 5,       ! 48 - 55
          & 3, 4, 4, 5, 4, 5, 5, 6,       ! 56 - 63
          & 1, 2, 2, 3, 2, 3, 3, 4,       ! 64 - 71
          & 2, 3, 3, 4, 3, 4, 4, 5,       ! 72 - 79
          & 2, 3, 3, 4, 3, 4, 4, 5,       ! 80 - 87
          & 3, 4, 4, 5, 4, 5, 5, 6,       ! 88 - 95
          & 2, 3, 3, 4, 3, 4, 4, 5,       ! 96 - 103
          & 3, 4, 4, 5, 4, 5, 5, 6,       ! 104 - 111
          & 3, 4, 4, 5, 4, 5, 5, 6,       ! 112 - 119
          & 4, 5, 5, 6, 5, 6, 6, 7/       ! 120 - 127
 C
BYTE NPOF2(7)
DATA NPOF2/1,2,4,8,16,32,64/      ! power of 2
C
DIMENSION LIST(50,100),NuH(100) ! NuH = # of hits / group
 CALL VZERO(NLa,45000)            ! set 0
CALL VZERO(NLb,45000)
CALL VZERO(NuH,100)
 PHI_BINSIZE = 360/180            ! 180 bins in φ [degrees]
THETA_BINSIZE = 180/1000          ! 1000 bins in θ [degrees]
C
DO N = 1, Number_of_hits          ! fill layer histogram
KF2= PHI(N) / PHI_BINSIZE         ! PHI(N) = φ of hit N
KT2= THETA(N) / THETA_BINSIZE ! THETA(N) = θ of hit N
IFI(N) = KF2                      ! Store KF2,KT2 for later use
ITE(N) = KT2
KT1=KT2-1                         ! neighbouring bins
```

```
KT3=KT2+1
IF( KF2.EQ.0 ) THEN                ! phi bin size = 2 . Therefore,
KF1=179                            ! neighbour of bin 0 = bin 179 ;
KF3=1
ELSE IF(KF2.EQ.179) THEN
KF1=178
KF3=0                              ! neighbour of bin 179 = bin 0 .
ELSE
KF1=KF2-1
KF3=KF2+1
END IF
 NL = IPOF2(LAYER(N))              ! LAYER(N) = layer (1-7) of hit N
 NLa(KF1,KT1)=NLa(KF1,KT1).OR.NL ! fill direct and indirect bins
NLa(KF2,KT1)=NLa(KF2,KT1).OR.NL
NLa(KF3,KT1)=NLa(KF3,KT1).OR.NL
 NLa(KF1,KT2)=NLa(KF1,KT2).OR.NL
NLa(KF2,KT2)=NLa(KF2,KT2).OR.NL
NLa(KF3,KT2)=NLa(KF3,KT2).OR.NL
 NLa(KF1,KT3)=NLa(KF1,KT3).OR.NL
NLa(KF2,KT3)=NLa(KF2,KT3).OR.NL
NLa(KF3,KT3)=NLa(KF3,KT3).OR.NL
 NLb(I,J)=-1                       ! flag direct bin
END DO
C                                  ! group and list hits
IGROUP=0                           ! # of group
DO N = 1, Number_of_hits           ! loop over hits (not bins!)
KF=IFI(N
KT=ITE(N)
ISD=NSD(NLa(KF,KT))                ! # of layers
IF(ISD.GE.MNL) THEN                ! ¿ Min_number_of_layers = MNL
C                                  ! accept only good hits
IF(NLb.LT.1) THEN                  ! hit was not yet clustered
IGROUP=IGROUP+1
CALL CLUSTER(KF,KT)                ! cluster all touching hits
END IF
NuH(IGROUP)=NuH(IGROUP)+1          ! List hits. Better: use one-
LIST(NuH(IGROUP),IGROUP)=N         ! dimensional list and pointers
END IF
END DO
END
 RECURSIVE SUBROUTINE CLUSTER(KF,KT)
C                                  ! Cluster all neighbours of KF,KT.
C
C                                  ! Recursive: only in FORTRAN 90.
```

```
C                                       ! In the current version of
C                                       ! ATLANTIS, which uses FORTRAN 77
C                                       ! the recursive subroutine CLUSTER
C                                       ! is written in a non recursive
C                                       ! way,which is more difficult to
C                                       ! understand and therefore not
C                                       ! given here.
 COMMON /HISTOGRAM/ MNL,IGROUP,NLa,NLb,NSD
BYTE NLa(0:179,0:999),NLb(0:179,0:999)
BYTE NSD(0:127)
 COMMON /LOOP/ JF1(8),JT1(8)
 DATA JF1/-1, 1, 0,-1, 1, 0,-1, 1/      ! Neighbours
DATA JT1/ 0, 0,-1,-1,-1, 1, 1, 1/ NLb(KF,KT)=IGROUP
DO J=1,8
JF=KF+JF1(J)
JT=KF+JT1(J)
IF (NLb (JF,JT)).LT.0.AND.              ! direct bin
& NHIT(NLa(JF,JT)).GE.MNL)              ! bin contains enough layers
& CALL CLUSTER(JF,JT)                   ! start clustering again
END DO
E ND
```

APPENDIX 2

```
DIMENSION XYZD(4,1000)
N=0
DO K1=1,999
DO K2=K1+1,1000
D1=ABS(XYZD(1,K1)-XYZD(2,K2))
D2=ABS(XYZD(2,K1)-XYZD(2,K2))
IF(D1.LT.DMIN.AND.D2.LT.DMIN) N=N+1
END DO
END DO
```