SVT Event Reconstruction Algorithms

Duncan J. Prindle
University of Washington
Nuclear Physics Laboratory GL-10
Seattle, Washington 98195

March 25, 1993

Abstract

We present two algorithms that are useful in analyzing the STAR SVT data. The first algorithm locates the primary vertex along the beam direction. This algorithm is fast enough that it will not be the bottle neck in offline event reconstruction. It may be fast enough to incorporate into a third level trigger but more work needs to be done to demonstrate this.

The vertex finding algorithm has been tested on small samples of central Au-Au, Si-Si and p-p events and typically finds the primary vertex within 300 μm of the generated vertex.

The second algorithm uses the vertex position to group the SVT hits in ϕ-θ. Each group is a set of points that is topologically consistent with having been created by a single track. This grouping algorithm should be fast enough that it will not be a significant bottle neck in offline event reconstruction. The effort to date has been to demonstrate that this is a useful approach to tracking.

The grouping algorithm has been tested on Monte Carlo central Au-Au events. In these events over 94% of the primary tracks that had three SVT hits are reconstructed correctly. For tracks with $p_\perp > 200$ MeV/c the track finding efficiency is over 97%.

The next steps are to determine track parameters for each group and then merge the SVT tracks with the TPC tracks. With the tracking information one should be able to locate the primary vertex to better than 20 microns in three dimensions.

1 GEANT Geometry conventions

In order to facilitate comparisons of different SVT geometries we have automated the procedure to get SVT hits into an easily understood table. To do this it was easiest to adhere to some conventions when defining the GEANT geometry. These conventions are:

1. Top volume name is `'SVT '`
2. Next volumes are named `'SVTn'` where n runs from 1 to 9. (This makes it easy vary the number of layers. A possible problem is if the tiling is complex enough that there is no simple bounding volume that does not overlap an adjacent layer. In practice this has not been a problem.)
3. Drift region is named `'SVTD'`
4. The connection between the SDD Local Reference Frame and TPC Reference Frame is shown in table 1.

When running GEANT the hits will be stored in the ZEBRA Hits bank. The TAS module mct will convert hits from the ZEBRA format to a native TAS table. The SVT hit row of the TAS table is given in table 2.

2 Vertex Finding

2.1 Basic idea

In tests of the vertex finding algorithm we have started from the GEANT SVT hit table listed in table 2. We should pass this table to the Silicon Resolution Simulator, then pass the resulting table to a module that merges nearby hits. The resulting table is the one we should be using in event reconstruction. **IT IS IMPORTANT THAT THESE STEPS BE INCLUDED IN THE STANDARD EVENT RECONSTRUCTION BEFORE THE July 1993 COLLABORATION MEETING.** (It won't change any results but it will give the perception of progress.)

The vertex finding algorithm relies on the fact that the majority of tracks passing through the SVT have very little curvature over an arc length of 10 to 15 cm. For a particle with $p_\perp = 100$ MeV/c in a 0.5T magnetic field we expect

$$\phi(R = 15cm) - \phi(R = 5cm) = 5.1^\circ \tag{1}$$

If we plot SVT hits in ϕ bins of 6° most of the tracks will have all their SVT hits within one bin. Some of the tracks will be split among adjacent ϕ bins.

The second angle that defines a straight line is θ. Since calculating θ takes some time we note that

$$\frac{Z_i - Z_0}{R_i} = \frac{1}{\tan(\theta)} \tag{2}$$

where Z_i and R_i are the Z coordinate and radius for each hit, Z_0 is the primary vertex position of the event and θ is constant for the track. We can define a variable $\bar{Z}_i$

$$\frac{Z_i - Z0}{R_i} = \frac{\bar{Z}_i - Z0}{R_{ref}} \tag{3}$$

or

$$\bar{Z}_i = \frac{R_{ref}}{R_i} \times Z_i + Z0 \times (1 - \frac{R_{ref}}{R_i}) \tag{4}$$

which is useful because it is a constant for the track and is fairly quick to calculate. We note that $\bar{Z}_i$ depends on Z_0. For the purposes of vertex finding and grouping the reference radius, R_{ref} should be chosen near the geometric mean radius.

The basic vertex finding algorithm is now easy to explain. We make a guess at Z_0, plot the SVT hits in appropriately sized bins of ϕ versus $\bar{Z}$, and count the number of bins containing hits from all three layers. When we choose Z_0 far from the true vertex position we will only have a combinatorial background. The closer Z_0 is to the correct value the more triplets we will find, limited by the number of tracks in the event.

The immediate question is; what is the appropriate bin size? If the ϕ bins are too large we have a large combinatorial background of triplets. If the ϕ bins are too small we have a poor efficiency for finding triplets since they are split among different ϕ bins. For the $\bar{Z}$ bins we have a few more considerations to take into account. If the $\bar{Z}$ bins are large we have a poor constraint on Z_0. If the $\bar{Z}$ bins are small the peak in the number of triplets as a function of Z_0 is narrow and we have to scan many values of Z_0 to be sure of finding the peak.

In the current implementation we have chosen a bin size of 3.6° in ϕ and 1mm in $\bar{Z}$ which makes an array of 100 by 400 for each of the layers. To minimize the problem of tracks being split among bins we treat each hit as being the size of a bin, 'overlay' this bin on the array and increment each array element by the amount of overlap. Thus each hit contributes to four bins.

The number of triplets per 1 mm step in Z_0 for one central Au-Au event is shown in figure 1. The peak near the generated vertex position of 8.334 cm is clearly visible above the background. We also show a close up of the region near the peak.

2.2 Some details about optimizing the speed of the Vertex Finder

Performing 401 scan steps is a little time consuming. We can take advantage of the fact that the peak is at least 10 bins wide by performing a rough scan in steps of 1 cm, then a fine scan about the rough peak. This reduces the number of scan steps to 62. (If we already knew Z_0 to 1 cm we could eliminate another 41 scan steps.) This is still fairly time consuming if we need to calculate ϕ and $\bar{Z}$ for all points on every step. It is clear that ϕ is independent of Z_0, so we need only calculate that once. We have written equation 4 to make it obvious there is a part ($Z_i \times \frac{R_{ref}}{R_i}$) that is independent of Z_0. The second term in equation 4 is Z_0 times a scale factor $1 - \frac{R_{ref}}{R_i}$. We store this scale factor as an array with one entry per ϕ bin (since R_i depends on ϕ) for each SVT layer. Z_0 times this scale factor (or step size) is interpreted as an offset in the $\bar{Z}$ direction into the two dimensional SVT hit array.

A further time saving device is to create an index of occupied bins for one of the layers. This means we don't have to waste time searching through empty bins. In practice we do not fill in the layer three array of ϕ versus $\bar{Z}$ but instead simply calculate and store ϕ and $\bar{Z}$ for each hit.

With these enhancements we can state the scan steps in a little more detail.

```
Loop over Z0
   For each value of Z0 loop over Layer 3 hits
      For each layer 3 hit get phi, Z indices
      Correct Z index by Z0*(Step(phi,layer)-Step(phi,3))
      If array elements (phi, corrected Z) for layers 1 and 2
        are non-zero we increment number of triplets for this Z0
      END Loop over Layer 3 hits
   END Loop over Z0
```

The first scan is done in 1 cm steps from $Z_0 = -20$ cm to $Z_0 = +20$ cm. The Z_0 bin containing the maximum number of triplets is the center of the ensuing 1 mm scan.

A few minor notes:

1. Using layer three hits to calculate the expected ϕ-$\bar{Z}$ bin allows us to bypass range checking for layers one and two if the assumed Z_0 is within a restricted range. In the case presented here all SVT layers are ± 20 cm so if we restrict $|Z_0| \leq 20$ cm we do not need to perform range checking. In practice we expect events in which the primary vertex is outside this range.

2. The amount of time required for the Z_0 scan is nearly proportional to the number of scan steps and the number of layer 3 hits.

3. There has been no attempt made to look for multiple primary vertices within one event. This amounts to examining the triplet array created by the first Z_0 scan. This would seem to be a good thing to do.

4. There has been no test of what happens when the primary vertex is not on the Z-axis. If it is a few mm away there should be no problem.

This section is intended to give the flavor of some of the optimizations that have been attempted and to suggest that further optimizations are not likely to be trivial. It is likely that tuning the algorithm to a specific compute environment will make it run significantly faster.

2.3 Results

The vertex finding algorithm has been tested on small Monte Carlo samples of central Au-Au, Si-Si and p-p events. The algorithm failed on one p-p event which had a single track through the SVT and on one Au-Au event in which the primary vertex was near -23 cm. In all other events the calculated vertex was within 0.5 mm of the generated vertex. It should be emphasized that this algorithm uses reconstructed space points as the input. These points need to be determined considerably better than 1 mm for the purposes of primary vertex reconstruction.

An implementation of the algorithm has been timed on an HP-710 running at 50 MHz with version 8.07 of the operating system and its Fortran compiler. The version of the SVT simulated had three layers at radii near 5, 8 and 11 cm, each layer 40 cm long with no tiling in ϕ. For central Au-Au events the time to find the primary vertex was approximately 0.8 cpu seconds.[1] The scan over Z_0 was approximately 0.5 cpu seconds.

The time used in the scan could be cut in half if we started with a measurement of Z_0 that was accurate to one or two cm. There is likely to be another factor of two to be gained by carefully analyzing the program. This still gives us an execution time of 250 ms. Since all ϕ bins are independent we could gain nearly a factor of 100 if we had that many cpus. (There is a bottle neck when evaluating the results of the rough scan.) It is not completely unrealistic to hope for execution times on the order of 5 ms for finding the Z coordinate of the primary vertex in a central Au-Au event starting from reconstructed space points. One must not under-estimate the amount of manpower required to implement this algorithm in a data acquisition environment. In particular making sure the program is robust and fails in a predictable manner (when it fails) is essential and may be harder than achieving the requisite timing characteristics.

3 Grouping

The so called 'grouping' technique is essentially an extension of the vertex finding method presented in the previous section. The method relies on

[1]Since the working group meeting we have been reminded of compiler switches which make the program run faster at the expense of file and memory size.

tracks being nearly straight lines over the small arc-lengths measured in the SVT allowing one to map reconstructed points onto a $\phi - \bar{Z}$ space where hits from the same track tend to group closer to each other than the typical distance between tracks. The emphasis to date has been to demonstrate that a pattern recognition technique like grouping is useful.

3.1 Display for Grouping Optimization

The grouping algorithm was inspired by attempts to display the SVT data in a meaningful form. We have developed a special purpose display program for the SVT data that has been very helpful in developing topological criteria for groups. This program displays SVT hits as ϕ versus $\bar{Z}$, with ϕ in radians and $\bar{Z}$ in cm. The results of the grouping algorithm are displayed by drawing lines between hits that have been grouped. Hits from different layers are represented with different colors. For 'correct' groups the hits in layer one are green, hits in layer two are light blue and hits in layer three are dark blue. Hits that were not used in any groups are white for layer one, gray for layer two and black for layer three. Groups that are 'incorrect' have hits colored red. It is possible to query the program to find out what tracks a given set of hits belong to.

Central Au-Au events have so many SVT hits that it is not possible to display all of them at once and still see correlations between hits. The display program allows one to zoom in on a region of interest. A typical region in a central Au-Au event is shown in figure 2. This is about 4% of the area covered by the SVT. The hits have been drawn much larger than the expected resolution in order to be easily seen. In this figure most of the hits are correctly grouped. There are many cases which are not resolved at this scale.

Figure 3 is another region of the same event at higher magnification. In this particular case all the hits were correctly grouped, with three hits (correctly) being left over. One thing that should be noted is that there are clear cases where a point does not belong with the closest group. It is not obvious in this figure but there are at least three pairs of points (due to tiling) that have been correctly 'merged'. Hits are still drawn larger than the expected resolution of 50μm.

Figure 4 is a view of a region in which the current grouping algorithm failed. In this area there are actually two tracks and we get an extra pair of hits in layer one because of tiling. It should be possible to handle this case correctly.

3.2 The Topological Criteria

Grouping is performed in a number of passes. The first pass selects hits that are very close together in $\phi - \bar{Z}$ space. The characteristic distance scale for the first pass is 0.025 cm in the $\bar{Z}$ direction and 0.625° in the ϕ direction. Each of the three subsequent passes is a factor of two larger. With these distances the first pass should have good efficiency down to $p_{\perp} = 400$ MeV/c and the fourth pass should go down to 50 MeV/c.

Once we find a set of SVT hits that are close enough together we require at least one hit in every layer. If there are more than three hits we take all pairs of hits in layers one and three and require there is a hit in layer two near the midpoint. If there is no layer two hit close enough we reject that pair. If there is more than one hit we take the closest. If a hit is included in more than one triplet we only take the best triplet. If two hits in a layer are close enough to each other we assume they are due to tiling and add them both to the group.

These steps may not be the ones one would think of at first, but they are close to what one does when looking at events. When trying to decide on grouping criteria there is no substitute for looking at examples.

3.3 Gruesome details of the grouping algorithm implementation that should be skipped over by everyone who is not planning to re-write the code

The current implementation of the algorithm is based on a 'bit-map' approach. There is an array that is 576 bins wide in ϕ by 1600 bins in $\bar{Z}$ by 10 bins deep. (Clearing this 35 MByte array is the single biggest usage of cpu time for the algorithm.) Each SVT hit has its identification number placed into the appropriate place in this array. For the first pass we use

$$IZ = \bar{Z} \times \frac{1600}{40} + 800 \tag{5}$$

$$I\phi = \phi \times \frac{576}{360} \tag{6}$$

On the second pass we use 800 instead of 1600 and 288 instead of 576. Likewise for the third and fourth passes we divide by two again.

To find a candidate group we search along ϕ, then along $\bar{Z}$ until we find an occupied bin. We then recursively examine all eight nearest neighbors to see if they are occupied. Thus a candidate group consists of all SVT hits that are in bins that are connected at least by corners.

(It may be faster and will certainly use much less memory to use a sorted list of SVT hits instead of this 'bit-map'. I would be happy to discuss how to implement this with anyone who would like to give it a shot.)

The distance between points is defined to be

$$d^2 = 0.01 \times (\phi_1 - \phi_3)^2 + (\bar{Z}_1 - \bar{Z}_3)^2 \tag{7}$$

where ϕ is in degrees and 0.01 is a conversion factor. It should also be noted that if $|\phi_1 - \phi_3| > 180°$ we make the appropriate correction.

The center point of a pair of hits is simply $\phi_c = (\phi_1 + \phi_3)/2$ and $\bar{Z}_c = (\bar{Z}_1 + \bar{Z}_3)/2$. We require that

$$d^2(layer2 - center) < 0.25^2 \times d^2(layer1 - layer3) + multiple\ scattering\ correction \tag{8}$$

The multiple scattering correction depends on $p_\perp$. We are using 0.00025, 0.0009, 0.0036 and 0.014 for the four passes. If the distance between two hits in the same layer is less than 0.01 we assume they are from the same track but on different wafers so we place the hits in the same group.

3.4 Results

We have optimized the grouping algorithm on an un-tiled event for the first two passes only. With the display program it is clear that the grouper is quite good at finding tracks and does not find a lot of extraneous tracks.

To quantify this we need to define an efficiency. We define the possible tracks as all tracks from the primary vertex that left at least one hit in each of the SVT layers. The found tracks are a subset. For a track to be found we require the group contain at least one hit from the track in each of the layers. The efficiency is the number of found tracks divided by the number of possible tracks. In figure 5 we plot these numbers as a function of $p_\perp$ for two central Au-Au events. The overall track finding efficiency is over 94%. For $p_\perp > 200$ MeV/c the efficiency is over 97%. There is a sharp drop in efficiency below $p_\perp = 120$ MeV/c. We can probably increase the efficiency at low $p_\perp$ with modifications in the algorithm.

Using a tiled geometry the track finding efficiency drops by about 2 or 3%. This is probably due to tiled hits on adjacent ladders not being merged properly. We can probably increase the efficiency to the level of the un-tiled geometry by handling this properly.

The track finding efficiency is high enough that this approach should be pursued.

Table 1: Orientation of `SDD` in TPC reference frame.

`SDD` Frame	TPC Frame
X	Phi direction
Y	Radial
Z	Z

Table 2: Elements contained in mct SVT hit table.

`id`	Unique SVT hit number.
`idet`	Should identify SDD. Currently is $Layer \times 2^{27}$ (I have not figured out how to get Ladder within the Layer and SDD within the Ladder. These should be encoded into bit fields also.)
`itrack`	GEANT Track number associated with hit.
`de`	Energy deposited in Silicon
`pr(3)`	px/ptot,py/ptot,pz/ptot of track at hit position.
`ptot`	ptot of track at hit position.
`sleng`	path length of track up to hit position
`tof`	time of flight of track up to hit position
`x(3)`	hit position. (Always at center of SDD)
`xp(3)`	hit position in local SDD reference frame

Figure 1: Number of triplets per 1 mm primary vertex scan step in Z_0. The peak near the generated vertex position of 8.334 cm is clear.

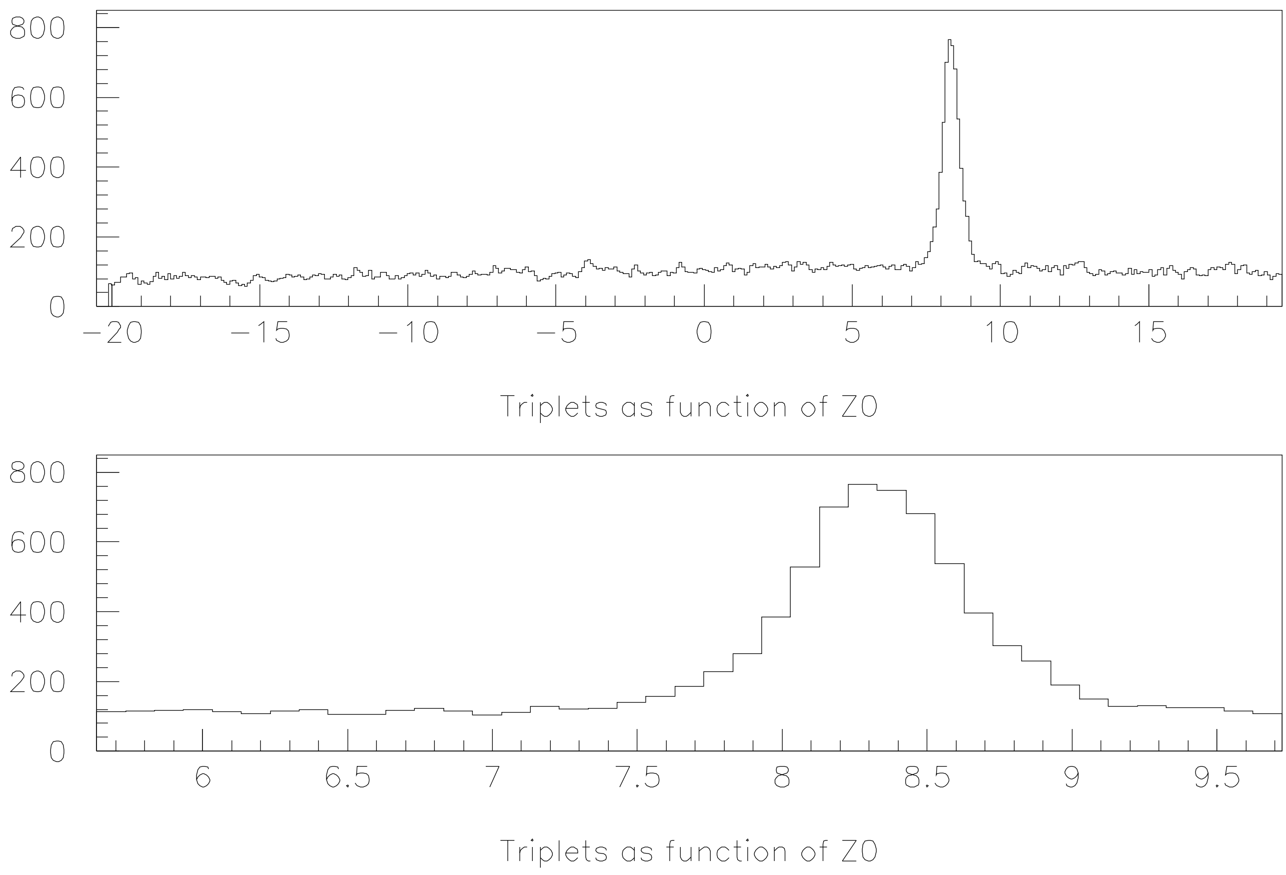

Figure 2: Part of SVT ϕ versus $\bar{Z}$ coverage for a central Au-Au event with a tiled geometry. ϕ (radians) is vertical and $\bar{Z}$ (cm) is horizontal. This is approximately 4% of the area covered by the SVT. (A color display helps for these plots.) The hits are drawn larger than the expected resolution of 50μm. Most of the groups are found correctly. Many groups are not resolved at this scale.

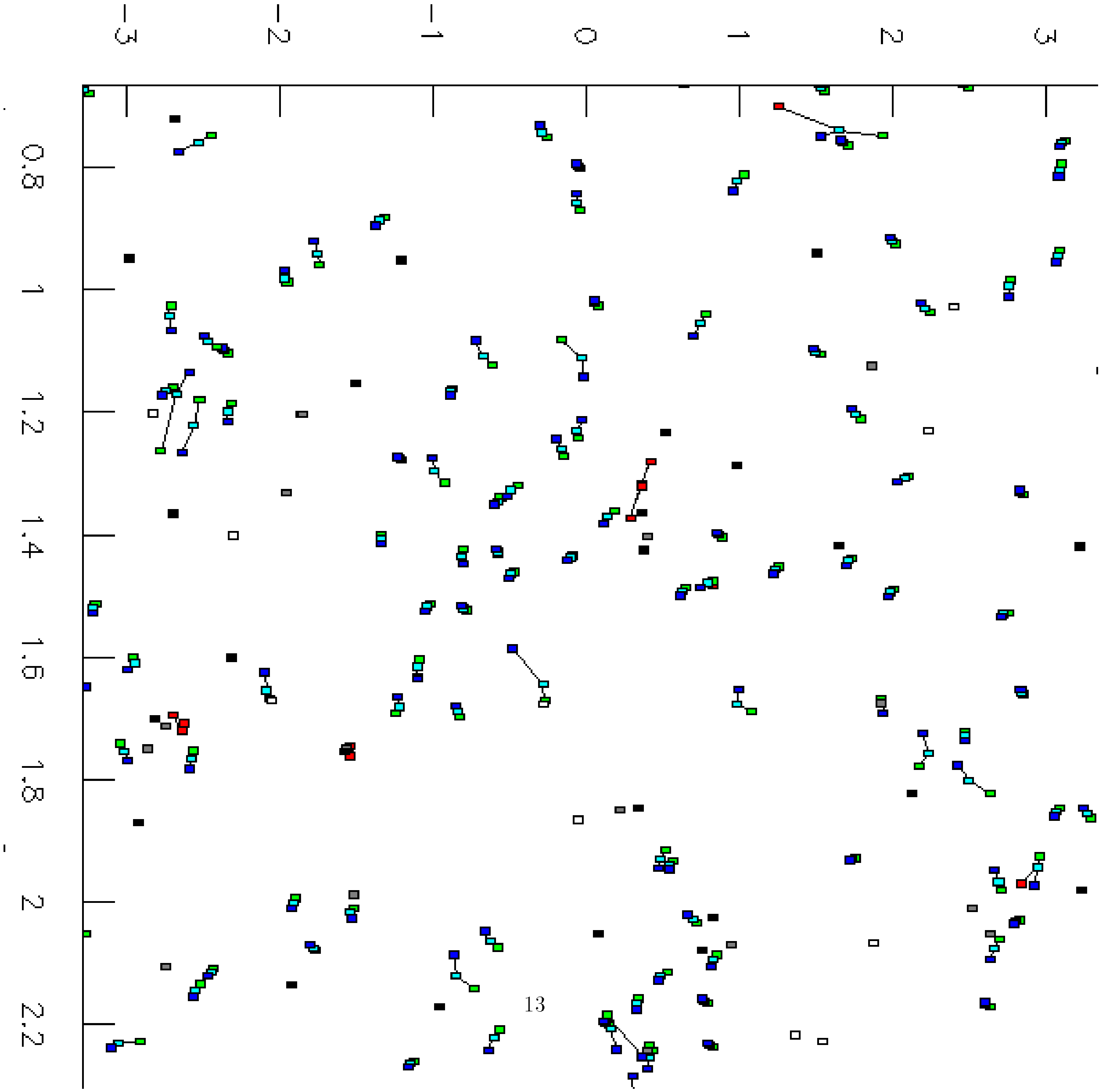

Figure 3: Part of SVT ϕ versus $\bar{Z}$ coverage for a central Au-Au event with a tiled geometry. ϕ is vertical and $\bar{Z}$ is horizontal. The current implementation of the grouping algorithm did everything correctly in this region. Three hits are unused and at least three hits due to tiling have been grouped with their partners.

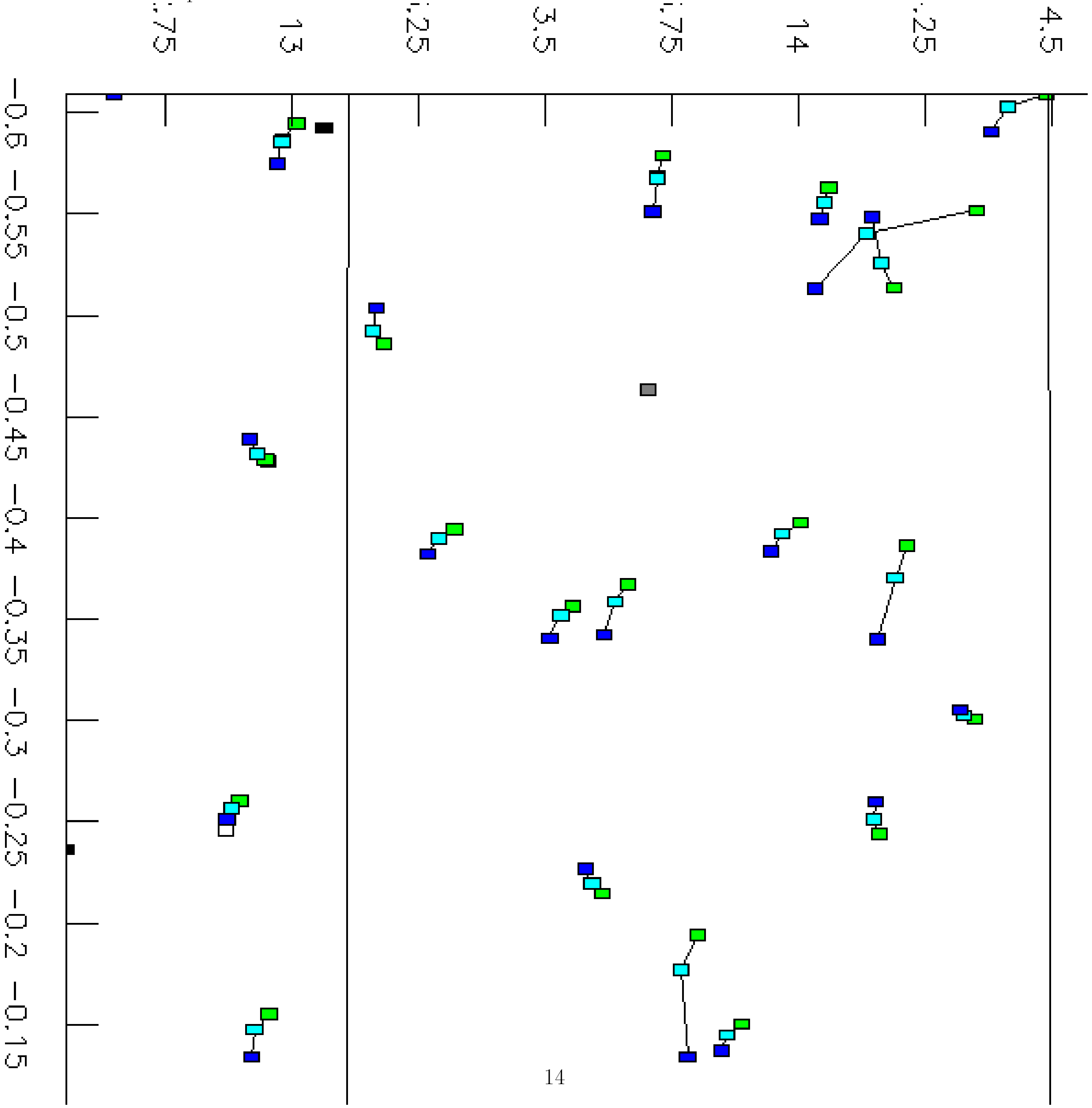

Figure 4: Part of SVT ϕ versus $\bar{Z}$ coverage for a central Au-Au event with a tiled geometry. ϕ is vertical and $\bar{Z}$ is horizontal. For these hits the current implementation of the grouping algorithm failed. With some training it is possible to discern the two tracks by eye.

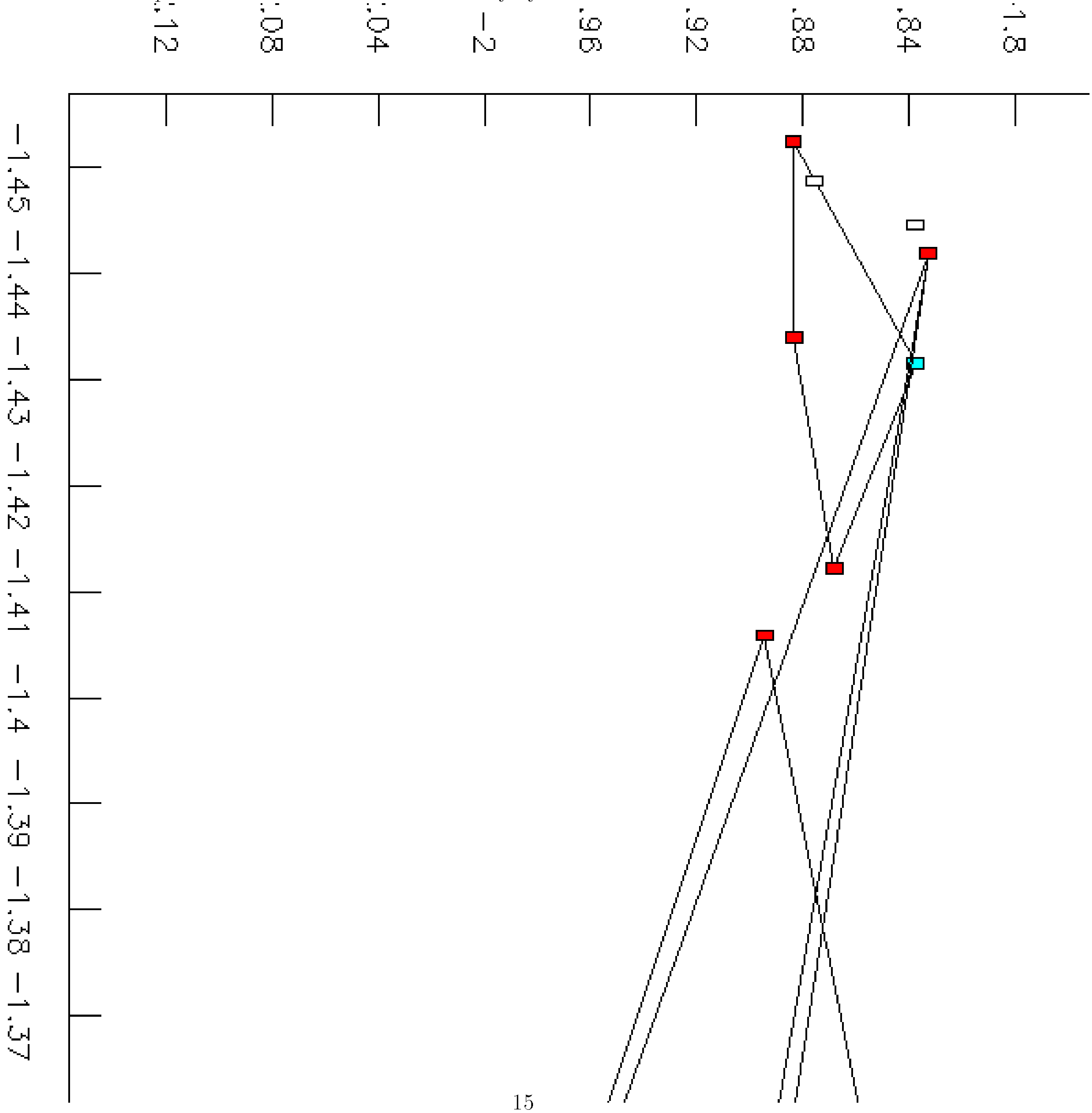

Figure 5: Track finding efficiency. Top panel is number of tracks from the primary vertex that left SVT hits in all three layers. Middle panel is subset of these that were grouped correctly. Bottom panel is ratio which is defined to be the efficiency.