

Curved Track Segment Finding Using Tiny Triplet Finder (TTF)

Jinyuan Wu, M. Wang, E. Gottschalk and Z. Shi

Abstract— We describe the applications of a track segment recognition scheme called the Tiny Triplet Finder (TTF) that involves the grouping of three hits satisfying a constraint forming of a track segment. The TTF was originally developed solving straight track segment finding problem, however, it is also suitable in many curved track segment finding problems. The examples discussed in this document are among popular detector layouts in high-energy/nuclear physics experiments. Although it is not practical to find a universal recipe for arbitrary detector layouts, the method of the TTF application is illustrated via the discussion of the examples. Generally speaking, whenever the data item to be found in a pattern recognition problem contains two free parameters, and if the constraint connecting the measurements and the two free parameters has an approximate shift invariant property, the Tiny Triplet Finder can be used.

Index Terms—Trigger, Pattern Recognition, Tiny Triplet Finder, TTF, FPGA Firmware

I. INTRODUCTION

TRACK segment finding is often the first step in many trigger systems executing complicate processes for high-energy physics experiments[1][2]. To identify and to confirm a straight-line segment in a plane with 2 parameters, for example, at least 3 hits that satisfy a constraint are needed. The 3 hits are grouped together to form a data item called “triplet”. Straightforward software implementation of such a function would require $O(n^3)$ execution time, where n is number of hits per plane, in order to examine all possible combinations of three hits using three layers of nested loop. In FPGA hardware implementation, this execution time must be reduced to $O(n)$, to match the time required to fetch the data. The execution time is reduced by “unrolling” two layers of loops, which consumes a significant amount of silicon resources in FPGA devices. The number of logic elements needed in many typical triplet finding implementations is $O(N^2)$ where N is the number of bins that each plane is divided into.

An algorithm, the Tiny Triplet Finder (TTF) [3][4] was developed for triplet finding. The logic element usage of the TTF implemented in FPGA devices is $O(N \log(N))$ which is

significantly smaller than $O(N^2)$ when N is large.

The TTF was initially developed to solve a straight track segment finding problem. Its application can be extended to other data item grouping problems if: (1) the data item is specified by two free parameters, i.e., the data item is a triplet and (2) the constraint connecting the measurements and the two free parameters has an approximate shift invariant property. In this article, we will discuss the application of the TTF in curved track segment finding.

The difficulties of curved track segment finding come in two aspects. The first is extra number of parameters. In general, a curved track in a plane has 3 parameters that require at least 4 hits to satisfy a constraint. Grouping 4 data items is more complicate than grouping 3 and is beyond the scope of this article. The second aspect is the nonlinear nature of the track which may not be shift invariant.

In high-energy physics experiments, sometimes tracks come from a know point, either the collision axis or a point-like target. In these cases, a curved track projected to a plane has 2 parameters and an approximate shift invariant property can be found which makes the TTF suitable for the track segment finding.

II. CYLINDRICAL DETECTORS

Consider a cylindrically symmetric colliding detector with 3 (or more) angular measurement layers with radii R_1 , R_2 and R_3 in the end-view as shown in Fig. 1.

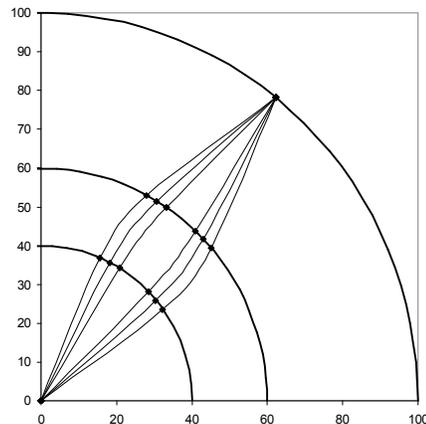


Fig. 1. Circular tracks passing a hit on the outer-most detector layer in a cylindrical detector

A circular track come from the collision beam axis can be specified with 2 parameters, the initial angle ϕ_0 and the radius

Manuscript received August 29, 2006. This work was supported in part Operated by Universities Research Association Inc. under Contract No. DE-AC02-76CH03000 with the United States Department of Energy.

J. Wu, M. Wang, E. Gottschalk and Z. Shi are with Fermi National Accelerator Laboratory, Batavia, IL 60510 USA (phone: 630-840-8911; fax: 630-840-2950; e-mail: jywu168@fnal.gov).

of curvature R_0 :

$$r = 2R_0 \sin(\phi - \phi_0) \quad (1)$$

For a hit on the outer-most layer with radius R_3 and angle ϕ_3 , there are many possible tracks or “roads” passing through it. Each road maps a hit in each of the inner layers. A coincident of two hits corresponding to same road on the two inner layers signifies a triplet with 3 hits: (R_3, ϕ_3) , (R_2, ϕ_2) and (R_1, ϕ_1) from a possible circular track:

$$\begin{aligned} R_3 &= 2R_0 \sin(\phi_3 - \phi_0) \\ R_2 &= 2R_0 \sin(\phi_2 - \phi_0) \\ R_1 &= 2R_0 \sin(\phi_1 - \phi_0) \end{aligned} \quad (2)$$

Define the angular offsets between the inner layer hits and the outer-most layer hit:

$$\begin{aligned} a_2 &= \phi_2 - \phi_3 \\ a_1 &= \phi_1 - \phi_3 \end{aligned} \quad (3)$$

A constraint between the two inner layer hits belonging to same road can be found:

$$\frac{\sin(a_2)}{R_3 \cos(a_2) - R_2} = \frac{\sin(a_1)}{R_3 \cos(a_1) - R_1} \quad (4)$$

Clearly the Equation (4) is shift invariant which is due to the axial symmetry of the detector. The constraint depends only on the angular offsets, rather than the hit angles.

An example of the FPGA implementation of the tiny triplet finder for the cylindrical detector configuration is shown in Fig. 2. Two bit arrays are used to map the hits from two inner detector layers. Usually natural detector elements are used as units of binning. For example, for scintillation fiber detector layers, a fiber corresponds to a bin in the bit arrays.

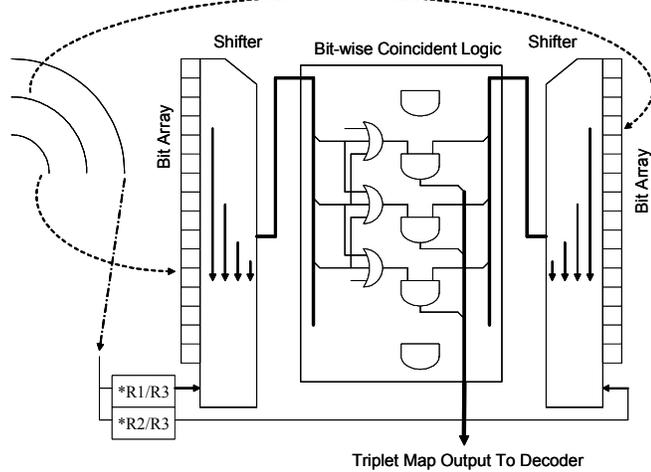


Fig. 2. The tiny triplet finder for cylindrical detectors can be implemented as shown. It contains two bit arrays, two (logarithmic) shifters and a bit-wise coincident logic block.

All hits from the inner layers are filled to the bit arrays. The filling process takes about n clock cycles where n is number of particle hits on each layer in a given event. After filling two hit patterns corresponding to the two inner layers are represented by the two bit arrays.

Then the hits on the outer-most layer are looped through, one hit per clock cycle. For each outer-most layer hit, the two inner layer bit patterns are aligned according to Equation (3). To calculate the amount of shifts for the two bit arrays from

the position (R_3, ϕ_3) of the outer-most layer hit in the FPGA, simple look-up tables are used implement constant multiplications. The outputs of the look-up tables provide appropriate angular shifts in units of fiber channels for the two inner layers. At the outputs of the two logarithmic shifters, the two inner layer bit patterns are aligned.

The shifted hit patterns are checked by a bit-wise coincident logic block according to Equation (4). Each OR-AND logic matches bins belonging a valid track road. A typical coincident logic map is shown in Fig. 3. The map is calculated based on a detector having three layers with radii: 40, 60 and 100 cm. The bin size is 2 mm on all three layers. The minimum transverse momentum is about $0.7 \text{ (GeV/c)}/1T$. The vertical and horizontal axes are offsets $(R_1 \alpha_1)$ and $(R_2 \alpha_2)$ in unit of mm respectively. Each 2mm-by-2mm box represents a possible product term in the bit-wise coincident logic. Only the quadrant with positive offsets is plotted.

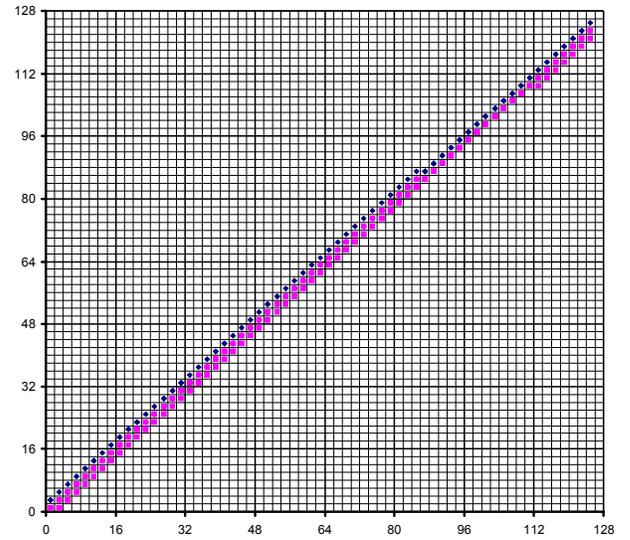


Fig. 3. The coincident map for circular tracks in cylindrical detector

Most portion of the coincident map can be implemented with logic elements configured as OR3-AND2 gates. Each non-zero bit on the output of the coincident logic block corresponding to a found triplet. The position of the non-zero bit on the output represents the transverse momentum of the triplet. The smaller offset of the non-zero bit corresponds to a higher momentum triplet.

Small non-linearity can be seen in the coincident map when the offsets are big (for low momentum tracks). It is contributed by the higher order terms in Equation (4). It should also be noted here that the slope of the map may not be 1 if other values of R_1 , R_2 and R_3 are chosen.

When the track multiplicity of the event is not too high, most of time there should be only one found triplet for each outer-most layer hit. The hits from other tracks may form fake triplets with certain probability. The user may choose from several fake triplet handling options. The simplest option is to output all triplets, real or fake, to the later stages and make finer examinations there. The user may also use momentum cut to keep triplet with highest momentum, once multiple

triplets are found for each outer-most layer hit.

In silicon pixel detectors, not only azimuth angles, but also z-positions of hits are detected. It may be desirable to find triplets not only in $R-\phi$, but also in $R-z$ projection. The constraint in the $R-z$ projection can be approximately written:

$$\frac{z_2 - z_3}{R_2 - R_3} = \frac{z_1 - z_3}{R_1 - R_3} \quad (5)$$

The triplet finding in the $R-z$ projection is the same as straight track segment finding problem for which we developed the tiny triplet finder [3][4]. The FPGA TTF in this case may use just one shifter rather than two.

III. PLANAR DETECTOR WITH POINT-LIKE TARGET

Consider three detector planes at $z = 20, 40$ and 60 cm as shown in Fig. 4. Assume the target is point-like and defined as the origin.

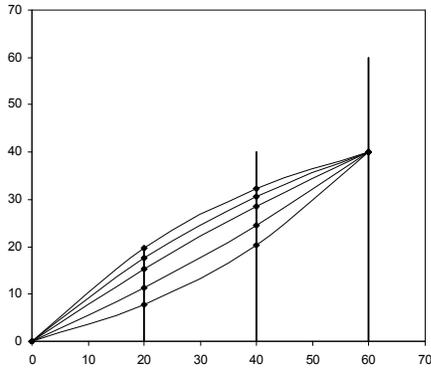


Fig. 4. Circular tracks passing a hit on the outer-most detector layer in a planar detector

The x - and y -axes are defined so that the z - x plane is the bend view and the z - y the non-bend view. The track equations with some approximations can be written:

$$y = kz \quad (6)$$

$$x = bz + cz^2$$

In the bend view, the coordinates of the hits on the three planes satisfy the following equations:

$$x_1 / z_1 = b + cz_1 \quad (7)$$

$$x_2 / z_2 = b + cz_2$$

$$x_3 / z_3 = b + cz_3$$

Consider tracks passing through a given point (x_3, z_3) on the outer-most plane. The constraint between the two hits on the two inner planes is:

$$\begin{pmatrix} x_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} z_3 - z_1 \\ z_3 - z_2 \end{pmatrix} \begin{pmatrix} x_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} z_2 - z_1 \\ z_3 - z_2 \end{pmatrix} \begin{pmatrix} x_3 \\ z_3 \end{pmatrix} \quad (8)$$

The x -coordinates of the two hits are linearly dependent on each other with a relative shift determined by x_3 in the last term. In our example, the planes are equally spaced with $z_3 = 3z_1$ and $z_2 = 2z_1$ and the constraint above can be further simplified:

$$x_1 - (x_3 / 3) = x_2 - (2x_3 / 3) \quad (9)$$

The constraint (8) or (9) is shift invariant, i.e., the functional form between x_1 and x_2 is independent of x_3 except a constant offset. This is only true under the approximation of the track

equations (6). Since the track equations are not accurate, the correlation between x_1 and x_2 are neither linear, nor independent of x_3 . Coincident logic maps are shown in Fig. 5 for the detector configuration shown in Fig. 4.

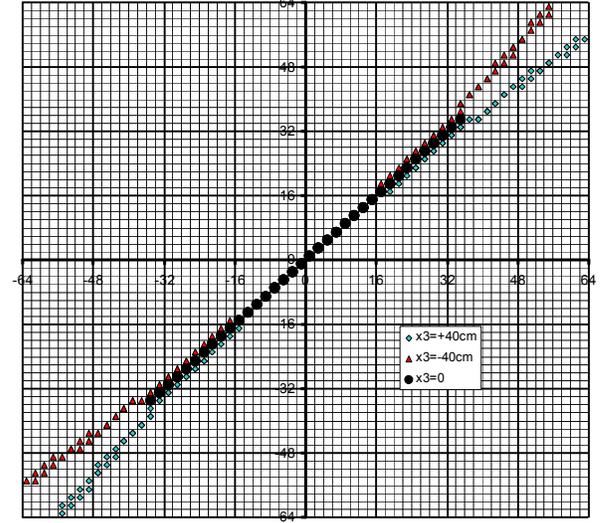


Fig. 5. The coincident maps for circular tracks in planar detector

The horizontal and vertical axes represent the hit coordinates in the inner two planes with suitable offsets, i.e., $(x_1 - x_3/3)$ and $(x_2 - 2x_3/3)$, respectively. The unit is in mm and a bin size 2mm is arbitrarily chosen. Circular tracks are used with minimum transverse momentum of 0.7 GeV/c/1T. The coincident maps for $x_3=0$, $x_3=40$ cm and $x_3=-40$ cm are plotted.

It can be seen that when x_3 is not too large, the correlation is essentially a straight line and that the dependence on the x_3 is very small. The approximation is good down to very low transverse momentum. The dependence on the x_3 coordinate appears for very low momentum tracks and larger x_3 . The coincident map for $x_3=0$ is essentially covered by the map for $x_3=40$ cm or $x_3=-40$ cm. Therefore in FPGA, the coincident map can be defined so that entire range of x_3 is covered for $x_3 > 0$ using the map for $x_3=40$ cm. For $x_3 < 0$, simply reverse the bit orders of the two bit arrays.

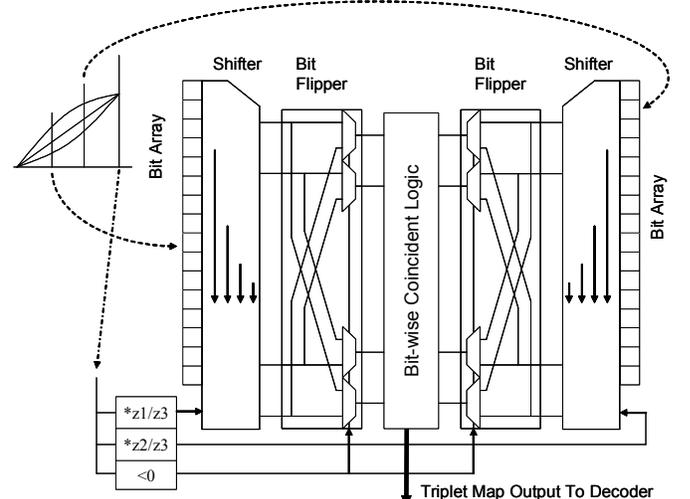


Fig. 6. Tiny Triplet Finder for curved segment finding in planar detector

The TTF shown in Fig. 6 is used for this problem. The hits

on the two inner planes are first booked to the two bit arrays. The coordinate of each hit on the third plane is used to find the amount of shift for the two arrays. The bit-wise coincident logic is implemented for $x_3 > 0$ cases. For $x_3 > 0$ cases, the bit flippers reverse the bit orders of the two hit pattern so that the same coincident map can be reused. Valid triplets are found by matching the two arrays via the bit-wise coincident logic block.

In the silicon pixel detectors, both bend and non-bend coordinates of a hit are available. For very high track multiplicity events, checking track segment constraint in only one view may result in high ghost track rate. Sometimes in addition to the bend constraint, it is necessary to also apply the non-bend view constraint of the track segments:

$$\left(\frac{y_1}{z_1}\right) = \left(\frac{y_2}{z_2}\right) = \left(\frac{y_3}{z_3}\right) \quad (10)$$

The non-bend view constraint is simply a doublet finding problem. For any hit at y_3 , any hit at y_2 or y_1 represents a candidate of the track segment. The doublet finding can be implemented in FPGA using ‘‘Hash Sorter’’ [5] or other similar schemes. It is possible to cascade a Hash Sorter stage and a TTF stage to perform a coarse non-bend view constraint checking first and then a finer triplet finding in the bend view to reduce ghost segment rate.

IV. TRIPLET FINDING FOR STRIP DETECTORS

With same detector area and measurement pitch, the channel count of silicon strip detector is significantly smaller than silicon pixel detector. However, extra considerations must be made for the strip detectors.

A. The x - and y -view planes

If a plane with x -measurement strips and a plane with y -measurement strips are placed close together, it is possible to determine both coordinate if there is only one hit strip in each view. With more than one fired strips in each view, ghost hits are produced at the wrong intersection points.

To eliminate the ghost hits information from other detector planes should be used.

A detector with only x - and y - planes is shown in Fig. 7.

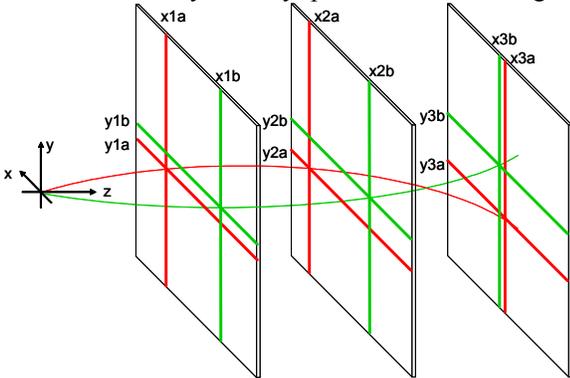


Fig. 7. A strip detector with x - and y - view planes only

Consider two tracks that fire 12 strips and form two x - triplets: $(x_{1,2,3a})$ and $(x_{1,2,3b})$ and two y - triplets: $(y_{1,2,3a})$ and $(y_{1,2,3b})$. The x - and y - triplets can be combined into four

possible valid track candidates: $\{(x_{1,2,3a}), (y_{1,2,3a})\}$, $\{(x_{1,2,3a}), (y_{1,2,3b})\}$, $\{(x_{1,2,3b}), (y_{1,2,3a})\}$, and $\{(x_{1,2,3b}), (y_{1,2,3b})\}$. Two of them are ghost tracks, but all of them, either real or ghost tracks, satisfy both constraints in x - and y -view. Note that it is not possible to identify the ghost track even in the data analysis stage without measurements from other detector components. So it is not a good detector configuration.

B. The u - and v -view planes

Consider a detector with u - and v - planes as shown in Fig. 8. It can be shown that most tracks can be correctly identified.

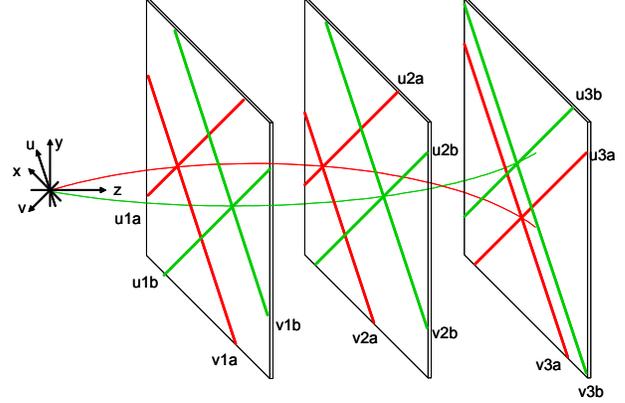


Fig. 8. A strip detector with u - and v -view planes

Assume the linear transformation from (x, y) to (u, v) is:

$$u = y \sin(\theta_u) + x \cos(\theta_u) \quad (11)$$

$$v = y \sin(\theta_v) + x \cos(\theta_v)$$

The track equations in (6) can be re-written in (u, v) coordinates with redefined parameters (See (16) also):

$$u = b_u z + c_u z^2 \quad (12)$$

$$v = b_v z + c_v z^2$$

The four parameters above are not all independent. The curvature parameters satisfy the following relation:

$$\frac{c_u}{c_v} = \frac{\cos(\theta_u)}{\cos(\theta_v)} \quad (13)$$

This is the condition that makes match between the u - and v - triplets. Note that in symmetric configuration: $\theta_u = -\theta_v$ (but u - and v - axes need not to be orthogonal to each other), the two coefficients c_u and c_v are equal.

Equations in (12) have the same format as the second equation in (6) so the same triplet finding procedures using TTF can be applied here. The entire track segment finding process takes following steps. First, the u - and v - triplets are found using two Tiny Triplet Finders. Assume that the planes are equally spaced with $z_3 = 3z_1$ and $z_2 = 2z_1$, the triplet constraints are reduced to very simple form:

$$u_1 = u_2 - (u_3 / 3) \quad (14)$$

$$v_1 = v_2 - (v_3 / 3)$$

Then match the u - and v - triplets by matching their curvature coefficients. When the u - and v -axes are symmetric about x -axis and the planes are equally spaced, the curvature matching condition (13) is reduced to:

$$(u_3 + u_1 - 2u_2) = (v_3 + v_1 - 2v_2) \quad (15)$$

The u - and v - triplets generated by different tracks, for

example: $\{(u_{1,2,3a}), (v_{1,2,3b})\}$ or $\{(u_{1,2,3b}), (v_{1,2,3a})\}$ not always satisfy the curvature matching condition. The ghost tracks are eliminated by this extra condition. The u- and v-triplet matching is a process matching two data items (triplets) with one parameter (curvature). The process can be implemented with simple FPGA functional blocks like Hash Sorters.

There are several possible choices of the curvature matching condition. The formula in (15) does not need the point like target condition. The track is allowed not to come from the origin. The reason of making this choice is the attempt to accommodate the errors caused by the approximation of the track equation.

In three detector planes, a total of 6 measurements are made to each track. These 6 measurements determine 3 independent parameters in the track equations and provide 3 constraints so that u- and v-triplets can be found and matched, eliminating most of ghost combinations. While in the x- and y-view only detectors, the 6 measurements also provide 3 independent parameters and 3 constraints. One of the constraints is in the bend view (x-view) while the other two are in non-bend view (y-view), which are essentially redundant and provide no information for ghost distinction.

There are still limitations in the u- and v-view only configuration. When the two tracks have same momentum, the ghost combinations of u- and v-triplets are not distinguishable without measurements from other detector components. This is the motivation of considering the combination of the x-, y-, u- and v- planes.

C. The combination of x-, y-, u- and v-view planes

Consider a detector configuration as shown in Fig. 9.

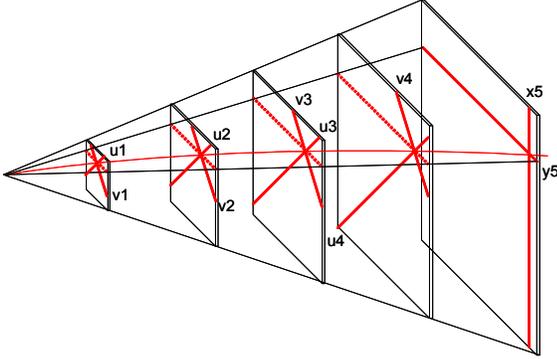


Fig. 9. A possible combination of x- y- u- and v-view strip planes

The planes 1-4 measure u- and v-coordinates of the tracks. The plane 5 provides y- along with x- measurements. As the first step, the active u- and v-strips in each of the planes 1-4 are to be matched with an active y-strip in the plane 5. Recall the track equations with symmetric configuration: $\theta = \theta_u = -\theta_v$:

$$y = kz \quad (16)$$

$$x = bz + cz^2$$

$$u = (k \sin(\theta) + b \cos(\theta))z + (c \cos(\theta))z^2$$

$$v = (-k \sin(\theta) + b \cos(\theta))z + (c \cos(\theta))z^2$$

Consider the y- measurement in plane 5 and the u- and v- measurements in any plane P = 1, 2, 3 or 4. A necessary condition that the 3 active strips belong to the same track is:

$$u_p = v_p + 2 \frac{y_5}{z_5} \sin(\theta) z_p \quad (17)$$

With this constraint, one can use Tiny Triplet Finder to identify the triplet of the active y-strip in plane 5 and the u- and v- strips in plane 1-4. The u- and v-strip pairs found in the planes 1-4 can be further linked into a track.

It is possible that more than one pairs of u- and v-strips in each of the planes 1-4 match a given y-strip in plane 5. Sometimes, two real tracks may hit the same y-strip while it is also very often that the intersections of the u- and v- strips from different tracks produce ghost hits.

To eliminate the ghost hits and match the correct active u- and v-strips, the triplet finding process in the x-view for planes 1-4 as discussed in the previous sections should be applied next. The active x-strips can be matched during this process.

V. CONCLUSION

We have described the application of the Tiny Triplet Finder for curved track segment finding through several examples. The specific detector layout is an experiment design issue that is beyond the scope of this article.

Generally speaking, the triplet finding is a process that groups three data items together with two free parameters (and therefore one constraint). Often in practical problems, the total number of the parameters is more than two. Luckily, in popular detector layouts, the parameters can be decoupled in selected subsets of measurements. If a subset contains two free parameters, triplet finding can be used. (When a subset contains only one independent parameter, two data items or doublet can be matched using Hash Sorter or similar process.)

Once the triplets are found, it can be used as a data item to group together with other two data items in a larger subset. As long as the new subset contains two or less additional parameters, triplet or doublet finding process can continue.

REFERENCES

- [1] Kulyavtsev et al., BTeV proposal, Fermilab, May 2000, BTeV-doc-66.
- [2] E.E. Gottschalk, BTeV detached vertex trigger, Nucl. Instrum. Meth. A 473 (2001) 167.
- [3] J. Wu, et al., "Tiny Triplet Finder (TTF) – A track segment recognition scheme and its FPGA implementation developed in the BTeV level 1 trigger system," in *Proc. 10th Workshop Electronics for LHC and Future Experiments*, Boston, MA, 2004, p. 68.
- [4] J. Wu, et al., "The application of Tiny Triplet Finder (TTF) in BTeV pixel trigger," *IEEE Trans. Nucl. Sci.* vol. 53 no. 3 pp. 671-676, June 2006.
- [5] J. Wu, M. Wang, E. Gottschalk, G. Cancelo and V. Pavlicek [for BTeV collaboration], "Hash sorter: Firmware implementation and an application for the Fermilab BTeV level 1 trigger system," in *Nuclear Science Symposium Conference Record, 2003 IEEE*, Portland, Oregon, 19-24 Oct 2003, vol. 2, pp. 1254-1256.