ATLAS Internal Note
DAQ -No- 008
CERN/EAST note 92-28
19 November 1992

Pattern recognition algorithms for triggering with a silicon tracker

Adrian Gheorghe , Werner Krischer

Abstract

Detection of high-p_t tracks in a silicon tracker is an important contribution to second-level triggering on electrons, in conjunction with calorimeter and preshower algorithms [1]. Silicon detectors transmit thresholded (zero-suppressed) data because of the low occupancy of the devices even at high accelerator luminosity. This way of presenting data in lists does not a priori lend itself to implementation in image processing devices. Also the variable-length data lists result in data-dependent execution times.

Using the programmability of several image processors we are familiar with (MaxVideo, DataWave), we present in this note algorithm formulations that can be implementated in these image processors, so that an average decision frequency of 100 KHz can be maintained easily even at high luminosity. Semi-custom implementations of these algorithms in field-programmable gate arrays are also possible.

We use data generated by the RD-2 collaboration, based on the ATLAS-A SIT geometry. Data are restricted to a region of interest (RoI) assumed to be extracted by an external 'Router' under control of a first-level trigger definition of that RoI.

1. Introduction

In [1] about the second-level electron triggering in the ATLAS-A detector, a Region-of-Interest (RoI) trigger algorithm for the outer silicon tracker has been described. Input to this algorithm is a list of pairs of numbers (*m,n*)'s, where *m* is an index of a silicon strip layer and *n* a measure of the arclength. From this the original 256 * 4 image had been recreated. The result of a least squares (LSQ) fit through 3 or 4 points had been stored in 16 - bit look up tables (LUTs) for p_t and goodness of fit. The method used was to move a mask through all the 256 positions and to look for a matching pattern.

Before looking for a fast way of implementing this particular algorithm, we tried out several other techniques on the test data generated by RD-2. The most obvious ones are the least squares fit as in [1] or a combinatorial method, where we picked the two outmost pixels (of 3 or 4), defined through them a narrow road and counted how many hits fell into it. This "road" method proved as reliable as a least squares fit through 3 or 4 points. Another equally successful algorithm is a variable slope histogram ("Hough" transform). We also experimented with various variations of histograms of dy/dx and/or of intersections of the straight lines defined by pairs of points with a middle line with somewhat worse results. Fig.1 shows an example where all these techniques gave the same result. Fig 1b) shows the plot of a Hough - transform with the corresponding parameter space in Fig. 1c). In Fig. 1a) one sees to the left the histogram of dy/dx , in the middle the intersections of straight lines of all pairs of points with a middle line and to the right the corresponding histogram.

Three methods had roughly the same success rate from the pattern recognition point of view, the LSQ fit, the road method and the Hough - transform. Although compute-intensive, the implementation of a Hough - transform working with lists of points can be mastered with special hardware [2], that has part of the analysis of the Hough parameter space built in (e.g. outputting per angle the maximum and its position and returning the coordinates inside the peak for further processing). For more general implementation, the best methods seem to be the road method and the LSQ - fit. For comparison reasons with [1] we restrict ourselves in this note to the LSQ - method.

We attempt to avoid to recreate the original 256*4 image, because this requires an additional computational step, and in particular because pipelined computer architectures have an execution time directly proportional to the image size. We therefore use the (*m,n*) - list directly. Processing the data 'symbolically' rather than 'iconically' we gained a factor of more than 30 in execution time producing identical results. In chapter 2 we will now describe in more detail the LSQ - algorithm and in chapter 3 its implementation on the DATAWAVE - processor and on some pipelined system. In chapter 3b) we will describe an implementation of a similar algorithm with a wider range, in both directions and with relatively short LUTs in a pipelined system (custom designed or off the shelf, like MaxVideo). We will summarize our results in chapter 4.

2. LSQ - algorithm for the ATLAS - silicon tracker

a) Geometry

The geometry of the SIT is described in more detail in [1]. We would like to emphasize a few geometrical features of the SIT which are useful for a better understanding of the following algorithms. We shall only consider the (r,Φ) - plane perpendicular to the beam (z axis) as in Fig. 2a) In this plane the SIT consists of four concentric silicon layers with the radii $\mathbf{r} = [\ ra\ rb\ rc\ rd\ ]^T = [\ 70\ 80\ 90\ 100\ ]^T$ cm. In the Φ direction each layer is divided into small detector elements of 0.01875 cm. Because the SIT is in a magnetic field the projected tracks will have a circular shape in a cartesian system, and to a good approximation a straight line in the (r,Φ) space. If a particle crosses the layers it has a high probability to hit each layer. In the best case a particle will hit each layer of radii $\mathbf{r} = [\ ra\ rb\ rc\ rd\ ]^T$ once in certain positions a,b,c,d, combined later in the vector $\mathbf{y} = [a\ b\ c\ d]^T$. In general we will have a few background hits around like in Fig. 2c) which represents data from single electrons with $p_t > 40$ at high luminosity in the space (r,n) (radius, position of hit). We frequently simply speak e.g. of the "point" b , if we mean the point (rb,b). The task is to find four or three points in different layers which correspond to a track. Tracks of interest for triggering are the ones with high p_t , i.e. with a curvature R > 25m, they can thus be considered in first approximation as straight.(see Appendix 1)

b) Mathematical formulation

To fit a straight line we have to solve the following overdetermined system of linear equations

$$\mathbf{A x} = \mathbf{y}$$

with the vector **x** of the 2 unknowns, $\mathbf{y} = [a\ b\ c\ d]^T$ and $\mathbf{A} = [\ \mathbf{r}\ \mathbf{u}\ ]$, where $\mathbf{r} = [\ ra\ rb\ rc\ rd\ ]^T$ and $\mathbf{u} = [\ 1\ 1\ 1\ 1\ ]^T$. We used the method of orthogonal triangularization $\mathbf{A} = \mathbf{QR}$ with the solution $\mathbf{x} = \mathbf{R}^{-1}\ \mathbf{Q}^T\ \mathbf{y} = \mathbf{T}\ \mathbf{y}$. Of course we had to compute altogether 5 of these transformation matrices **T** in advance for all the cases abcd, abcM, abMd, aMcd and Mbcd, where the M stands for a missing layer.

In the following implementations we first looked for the minimum residual, expressed as a function of the y-s :

$$res2 = \| \mathbf{Ax - y} \|^2 = \| \mathbf{ATy - y} \|^2 = \| (\mathbf{AT - E})\ \mathbf{y} \|^2 = \mathbf{y}^T\ \mathbf{C}\ \mathbf{y} \qquad (1)$$

where **E** is the 4*4 identity matrix and

$$\mathbf{C} = (\mathbf{AT - E})^T(\mathbf{AT - E}) = \mathbf{E - AT} \qquad (2)$$

and only then computed the p_t for the track with minimal residual.

To use small numbers in the implementation we used the fact that if $\mathbf{y} = \mathbf{z} + \mathbf{w}$, the sum of a vector $\mathbf{z}$ with $\mathbf{z}^T \mathbf{C}\, \mathbf{z} = 0$ and a vector $\mathbf{w}$, then

$$\text{res2} = \mathbf{y}^T \mathbf{C}\, \mathbf{y} = \mathbf{w}^T \mathbf{C}\, \mathbf{w} \qquad (3)$$

To fit a straight line through the four points (ra,a),..,(rd,d) we shall consider :

$$\mathbf{y} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} a \\ b_1 \\ c_1 \\ d \end{bmatrix} \qquad \text{and} \qquad \mathbf{w} = \Delta\mathbf{y} = \mathbf{y} - \mathbf{z} = \begin{bmatrix} 0 \\ b-b_1 \\ c-c_1 \\ 0 \end{bmatrix}$$

$$\mathbf{y} = \mathbf{z} + \Delta\mathbf{y}$$

where {a b_1 c_1 d} are collinear points (Fig. 3b). Then applying formula (3) one obtains :

$$\mathbf{res}2 = \mathbf{y}^T \mathbf{C}\ \mathbf{y} = (\mathbf{z} + \Delta\mathbf{y})^T \mathbf{C}\, (\mathbf{z} + \Delta\mathbf{y}) =$$
$$= \mathbf{z}^T \mathbf{C}\, \mathbf{z} + \mathbf{z}^T \mathbf{C}\, \Delta\mathbf{y} + \Delta\mathbf{y}^T \mathbf{C}\, \mathbf{z} + \Delta\mathbf{y}^T \mathbf{C}\, \Delta\mathbf{y} = \Delta\mathbf{y}^T \mathbf{C}\, \Delta\mathbf{y},$$

i.e. the residual of a line fit through the four points {a b c d} is identical to the residual of a fit through the points {0 b-b1 c-c1 0} (Fig. 3c)**.**

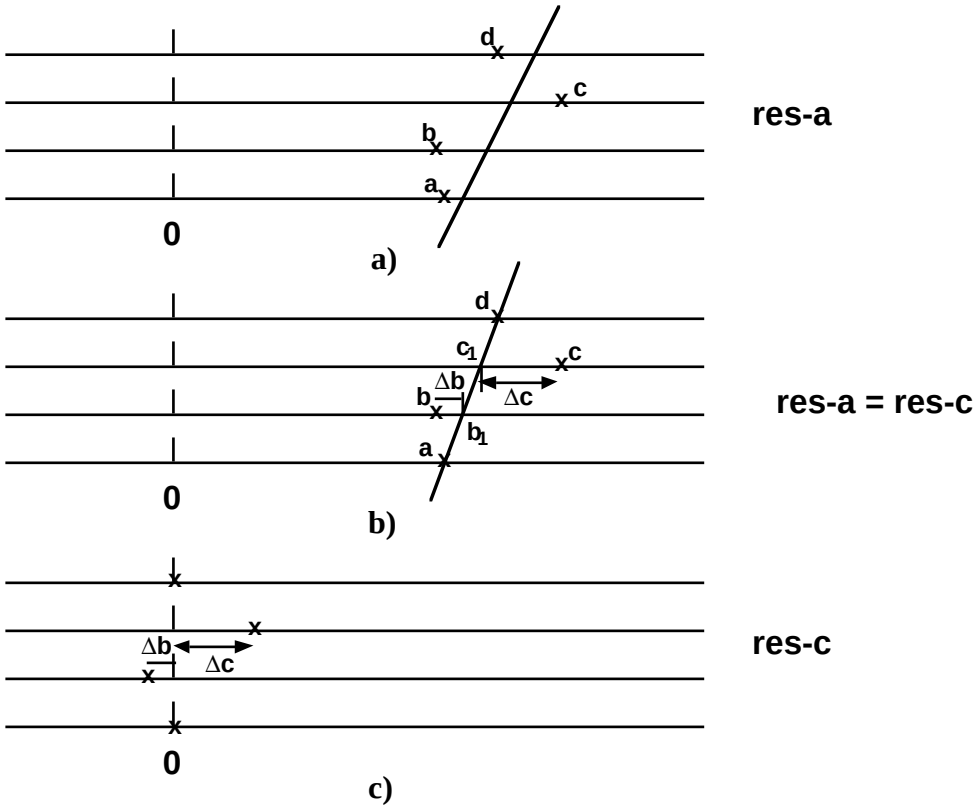

Fig. 3

3. Implementations

a) Datawave processor

We assume the input lists have been read and the vectors $\mathbf{y} = [a\ b\ c\ d]^T$ have already been created. Fig. 5 shows the essential part of the Datawave implementation. **y** - vectors arrive in cell (5,0). Columns 1 and 2 compute res2 corresponding to the 5 cases

abcd , abc , ab d , a cd , bcd

in rows 5, 4, 3, 2, 1, respectively. abcd is transmitted via column 0 and row 0 until cell (0,3). The res2 arrive in column 3 , where their minimum is chosen and passed on to the north cell (in this implementation we did not give preference to the case without missing layers).
Cell (0,3) compares the old with the new minimum, updates the new minimum and keeps the corresponding **y** for further computations**.**

The timing corresponds roughly to 30 cycles per **y,** i.e. 120 or 240 nanoseconds per 5 least square fits. By parallelizing one easily can build up a system that runs at the desired 100 kHz speed for the second level trigger. To gain higher performance figures we could use some of the ideas we developed later , e.g. in chapter 3 c.

b) Pipelined system 1

Here we would like to describe a hardware implementation of a similar algorithm as in [1]. We shall describe only the time critical part, the computation of the residual and the choice of the best fit. The input part which creates the quadruples is common to all our implementations and is only a small piece of hardware which we shall not describe here. (see Fig. 6)

This implementation uses the fact that the $\Delta n_i = n_i - n_{ia} = \frac{r_i(r_i - r_a)}{2Rd}$ $i\{\{a,b,c,d\}$ is only a function of the curvature R and the layer radius r_i (see Appendix 1). On the other hand for high p_t ($p_t > 15$ GeV/c -> R > 25m) Δn_i has small values ($\Delta n_i < 30$).

Having a quadruple $\mathbf{y} = [a\ b\ c\ d]^T$ we can compute:

1) $\Delta n_a = a - a\frac{r_a}{r_a} = 0$

2) $\Delta n_b = b - a\frac{r_b}{r_a}$ (LUT-b, ALU-B)

3) $\Delta n_c = c - a\frac{r_c}{r_a}$ (LUT-c, ALU-C)

4) $\Delta n_d = d - a\frac{r_d}{r_a}$ - (LUT-d, ALU-D)

5) check if the differences are in the accepted range (in the mask)
(LUT-B, LUT-C, LUD-D)

6) read the residual (pt,.....) from LUT-R and choose the best fit.

At the corresponding address of $\mathbf{y} = [a\ b\ c\ d]^T$ LUT-R keeps the best fit of the combinations of three or all four points

This system solves the problem of fitting four points in less than $T_f = 40ns$ assuming we use pipeline registers between stages. If one used the MAXVIDEO 20 system, which is very feasible, the time would be $T_f = 50ns$.

If n_a, n_b, n_c, n_d are the numbers of hits per corresponding layer the total computing time for one RoI would be :

$$T \approx (n_a+1)\, n_b n_c n_d\, T_f$$

For example in the MAXVIDEO case $T_f = 50ns$, if $n_a = n_b = n_c = n_d = 3$ the total computing time for one event is $T = 5.4\mu s$.

Of course we would only implement such an algorithm in this serial way, if the numbers of hits per layer are relatively small as in the present RD-2 data. Otherwise we would only have to parallelize just a few stages. More realistic simulation is being done.

c) Pipelined system 2

Fig 7 shows the block diagram of that implementation.

The main part in the hardware implementation is computing the residual for a combination of four points. To find the residual for the five fits (abcd, abc , ab d, a cd, bcd) one has to compute (see Appendix 2) :

1) type023 = mod((d-a),3) in ALU0, LUT0
   type1 = mod((c-a),2) in XOR1
   type4 = mod((d-b),2) in XOR4
   (ALU0, LUT0, XOR1, XOR4)

2) $\left[b - a - \frac{d-a}{3}\right], \left[c - a - 2\frac{d-a}{3}\right], \left[b - \frac{c+a}{2}\right], \left[c - \frac{d+b}{2}\right]$
   (ALU2, ALU3, ALU1,ALU4)

3) four select signals for the five LUTs depending on whether the four above values are in the accepted range (-4,4) or not
   (AND-OR2, AND-OR3, AND-OR1, AND-OR4).
   In a real implementation the five LUTs can be combined in two without any penalty.

4) reading the residuals from LUTs and choose the best fit.
   (LUT1234, LUT12x4, LUT1x34, LUT123x, LUTx234)

For all these five fits the time can be less than 40ns assuming we use pipeline registers between stages or using MAXVIDEO 20 system, $T_f = 50ns$.

The same comments about the total time T as in the previous example apply also to this implementation.

4. Conclusions

We described in this note some silicon strip tracker algorithms and several implementations. Because of the low occupancy of a silicon tracker we only work with lists of points (symbolically) and not on images (iconically). The performance will depend on the choice of the algorithm, which in turn depends on the way the data will be input, the role of a router, the number of hits in total and / or per layer, the degree of parallelism, the availability of processors (e.g. DAVIS) , at which frequency , etc.

We have shown that second level silicon strip tracking algorithms can be implemented easily at the desired 100 kHz rate with today's technology.

5. Acknowledgements

We gratefully acknowledge the help and suggestions from Rudolf Bock and the discussions we had with Richard Hawkings and Tony Weidberg from Oxford University, who supplied us with the simulated data.

6 References

[1] R.J. Hawkings, A.R. Weidberg, N.N. Ellis and A.T.Watson : Second Level Electron Triggering in the ATLAS-A Detector, ATLAS Internal Note INDET-NO-013

[2] Y. Ermolin and C. Ljuslin : Hough Transform Processor, CERN - LAA RT/88-06

Appendix 1

To see how a track looks in the plane (r,n) assuming R >> r , we have (see Fig. 4a):

$$L \approx \frac{r^2}{2R} + \phi_0 r \qquad n = \frac{L}{d}$$

n is the detector element which is hit by the track.

With that assumption R >> r in the plane (r, n) the track is a parabola. Because tracks of interest for triggering are the ones with high p_t , i.e. with a curvature R > 25m, the track can be considered in first approximation as straight.

In (Fig. 4b)

$$\Delta n_i = n_i - n_{ia} = \frac{r_i^2}{2Rd} - \frac{r_i}{r_a}\frac{r_a^2}{2Rd} = \frac{r_i(r_i - r_a)}{2Rd}$$

Δn_i is not a function of Φ_0 like L, but only a function of R and r_i.

Appendix 2

Equidistant layers

For equidistant layers as in our special case : $\mathbf{r}$ = [ra rb rc rd]^T = [70 80 90 100]^T cm

$$\mathbf{z} = \begin{bmatrix} a \\ \frac{2a+d}{3} \\ \frac{a+2d}{3} \\ d \end{bmatrix} \qquad \Delta\mathbf{y} = \begin{bmatrix} 0 \\ b-b_1 \\ c-c_1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ b - a - \frac{d-a}{3} \\ c - a - 2\frac{d-a}{3} \\ 0 \end{bmatrix}$$

Then the residual is just a function F of $b - a - \frac{d-a}{3}$ and $c - a - 2\frac{d-a}{3}$:

$$res2 = F\left(b - a - \frac{d-a}{3},\ c - a - 2\frac{d-a}{3}\right)$$

With **Z**, the space of integers {...-2 , -1 , 0 , 1 , 2 , ...}, and **Q**, the space of rational numbers, we can formulate the three possible cases of $b - a - \frac{d-a}{3}$, $c - a - 2\frac{d-a}{3}$ in the following way :

type023 = mod((d-a),3) = 0 => $b - a - \frac{d-a}{3}$, $c - a - 2\frac{d-a}{3}$ [Z

type023 = mod((d-a),3) = 1 => $b - a - \frac{d-a}{3}$ [{x [**Q** | $x = z + \frac{2}{3}$, z [**Z**}

$c - a - 2\frac{d-a}{3}$ [{x [**Q** | $x = z + \frac{1}{3}$, z [**Z**}

type023 = mod((d-a),3) = 2 => $b - a - \frac{d-a}{3}$ [{x [**Q** | $x = z + \frac{1}{3}$, z [**Z**}

$c - a - 2\frac{d-a}{3}$ [{x [**Q** | $x = z + \frac{2}{3}$, z [**Z**}

We consider now

$$res2 = \mathbf{F}\left(\left[b - a - \frac{d-a}{3}\right],\left[c - a - 2\frac{d-a}{3}\right],type023\right)$$

where [x] = integer part of x and type023 [{0,1,2} is one of the cases above.

In our case the fit is accepted if $\left[b - a - \frac{d-a}{3}\right]$, $\left[c - a - 2\frac{d-a}{3}\right]$ are small numbers (e.g. [(-4,4))

For a hardware implementation we store the function $\mathbf{F}\left(\left[b - a - \frac{d-a}{3}\right],\left[c - a - 2\frac{d-a}{3}\right],type023\right)$ in a LUT with an 8 bit address : 3bits for $\left[b - a - \frac{d-a}{3}\right]$, 3bits for $\left[c - a - 2\frac{d-a}{3}\right]$ and 2bits for the "type023"

Because the algorithm has to find the best fit for three or four points we have to solve also the problem of fitting three out of the four points {a b c d}.

To fit ab d on the layer of radius ra,rb, rd res2 = $\mathbf{F}\left(\left[b - a - \frac{d-a}{3}\right],type023\right)$

a cd on the layer of radius ra,rc, rd res2 = $\mathbf{F}\left(\left[c - a - 2\frac{d-a}{3}\right],type023\right)$

Similarly to fit abc on the layer of radius ra,rb, rc res2 = $\mathbf{F}\left(\left[b - \frac{a+c}{2}\right],type1\right)$,

with type1[{0,1} type1 = mod((c-a),2)

and to fit bcd on the layer of radius rb, rc, rd res2 = $\mathbf{F}\left(\left[c - \frac{b+d}{2}\right],type4\right)$,

with type4[{0,1} type4 = mod((d-b),2).

In all four cases one needs less than 8bits (3+2bits) for the LUT address