# A Fast Track Pattern Recognition

**Pablo Yepes**
Bonner Nuclear Laboratory, MS 315
Rice University
6100 Main Street, Houston TX 77005
*E-mail: yepes@physics.rice.edu*
**STAR Note 248**

March 11, 1996

**Abstract**

An algorithm for fast track pattern recognition is presented. Its main features are an optimized data organization and a conformal mapping to speed up fitting procedures. The code is applied to a generic collider detector. Reconstruction time, track finding efficiencies, and momentum resolution are presented.

*To be published in Nuclear Instruments and Methods A*

## 1 Introduction

The complexity of events in particle and nuclear physics has increased greatly in recent years. When analyzing complex events, track information could be essential to decide whether an event is interesting for further processing. In general, track information is obtained from a set of space points provided by tracking detectors. Therefore, a fast way to reconstruct tracks from those space points could be a fundamental ingredient of a high-level trigger or off-line data filtering.

The time required by traditional tracking programs scales with the square of the number of space points. This presents a challenge for experiments where the number of tracks is of the order of hundreds or even thousands.

In this study, a fast track finding algorithm initially developed for a high level trigger in the framework of the STAR Collaboration [1] is presented. The algorithm has been implemented in a generic program which can be applied to different tracking scenarios.

# 2 Algorithm

The algorithm has been developed to perform track pattern recognition with two or three dimensional space points. Even though the main emphasis of the program is on obtaining a very fast algorithm, very reasonable efficiency and momentum resolution are also achieved.

## 2.1 Conformal Mapping

The program assumes particles follow helical trajectories. The assumption is exact when the magnetic field is constant which is a good approximation in most current experiments. Under such conditions, particles follow a circular path in the plane perpendicular to the magnetic field. In addition, trajectories are straight lines in the (s,z) plane, where s is the trajectory length in the 3 dimensional space and z is the coordinate along the magnetic field. Therefore, reconstructing a track is equivalent to finding the parameters defining its path in those two planes. Since fitting straight lines is easier and faster than fitting circles, it is desirable to transform the circular trajectories into straight lines. This is achieved by applying a conformal mapping to the coordinates in the plane perpendicular to the magnetic field.

A space point (x,y) produced by a track crossing the space point $(x_t, y_t)$ is transformed [2]:

$$x' = \frac{x - x_t}{r^2} \quad y' = \frac{y_t - y}{r^2}$$
$$where: \quad r^2 = (x - x_t)^2 + (y - y_t)^2$$

This transformation requires the knowledge of a point included in the track trajectory. If tracks are assumed to have been produced at the interaction point, IP, $(x_t, y_t)$ is replaced by the IP coordinates. This case will be referred to as *vertex constraint*. If no assumptions are made about the production point of a track, $(x_t, y_t)$ is replaced by the coordinates of the first space point associated with the track. This option will be referred to as *no vertex constraint*.

When tracks are reconstructed with *vertex constraint* the conformal mapping is applied at initialization time. However when this is not the case, the mapping is performed as the program explores the different track candidates.

## 2.2 Data Organization

An essential component in a fast algorithm is an efficient handling of the data. This is achieved by organizing space points in sub-volumes. Sub-volumes are obtained by segmenting the tracking volume into cubes in the (r,$\phi$,$\eta$) space, where r, $\phi$ and $\eta$ stand for the radius, azimuthal angle, and pseudorapidity [1] respectively. Each volume is identified by three indexes, $i_r$,$i_\phi$,$i_\eta$, corresponding to the three coordinates used: r,$\phi$ and $\eta$. Each space point is uniquely associated with a sub-volume. The index in the radial direction plays a special role in the track finding and is referred to as *the layer*.

[1] However since the calculation of the pseudorapidity may be time consuming, for the purpose of the pattern recognition $\eta$ is approximated by the function: $3z/(|z|+2r)$, where z and r are the coordinates along the beam line and the radius of the space point respectively.

Linked chains of space points are established to allow fast loops over all the space points in sub-volumes and layers.

### 2.3 Track finding

The pattern recognition is based on the *nose-following* method with speed optimized procedures for track extension and fitting. Figure 1 shows a diagram of the algorithm.

The program proceeds starting from the outer most layers towards the inner most ones. The track finding is initiated by building track segments. A segment is formed by linking a space point, k, to the closest unused space point, j. In this context distance, d, is defined as:

$$d \ = \ |i_r^k - i_r^j| \ \ (|\phi^k - \phi^j| + |\eta^k - \eta^j|)$$

where $i_r^k$ stands for the layer of the space point k. If point k is in sub-volume $(i_r^k, i_\phi^k, i_\eta^k)$ the closest point is only searched in sub-volumes $(j_r, j_\phi, j_\eta)$ satisfying the conditions:

$$i_r^k - N \leq j_r \leq i_r^k - 1, \quad i_\phi^k - 1 \leq j_\phi \leq i_\phi^k + 1 \ \ and \ \ i_\eta^k - 1 \leq j_\eta \leq i_\eta^k + 1$$

where N is a parameter in the program. The above algorithm is iterated until track segments reach a certain number of space points. The length of these segments is another parameter in the program.

When the track segment is long enough, points are fitted to straight lines in the conformal space for the bending plane and in the (s,z) plane [2]. Then, tracks are extended by including the unused space point closest to the fit. As in the segment formation case, only points in close sub-volumes are included in the search. If the space point closest to the fit does not give a fit with an acceptable $\chi^2$, the track is rejected and deleted as a candidate.

Unless stated otherwise, the code was set to pass twice looking for tracks in the results presented in this study. The first pass with a vertex constraint and the second without it. This second pass only uses space points which were not associated with tracks in the first pass.

## 3 Results

A tracking detector in a generic experiment is simulated with a simple Monte Carlo. A cylindrical geometry is chosen since that is the most common setup in collider detectors. However the program can be also utilized with other geometries more common in fixed target experiments.

The simulated detector consists of concentric sensitive cylinders centered around the beam axis corresponding to the different layers mentioned above. The radii of the layers range from 40 to 100 cm, increasing in constant steps. Tracks are generated with a flat distribution in pseudorapidity in the interval (-2,2) and with an exponential transverse momentum, $p_t$, distribution $e^{-p_t/2.}$. Tracks originate at the center (0,0,0) of the detector and propagate through a constant magnetic field of 0.5 Tesla. Space

[2] There is an option to work only in the bending plane.

points are obtained by intersecting the generated tracks with the layers. The position of the space points is smeared in the three spatial coordinates. In the present study a spatial resolution of 300 microns is used. For simplicity, no physics effects, like multiple scattering, energy loss, etc., are considered.

CPU times are obtained running the code on an IBM RISC 6000/370. The machine benchmarks are summarized in Table 1.

Figure 2.a shows the CPU time needed to reconstruct one track as a function of the number of tracks per event in the case of a detector with 10 layers. In addition to the total time per track, the setup time is also shown. The initial phase of the algorithm, the setup phase, consists of organizing the data in a convenient way. As can be observed, the setup phase takes between 50 and 70% of the total time depending on the number of tracks. It is to be noted that the setup procedure could be sped up if the incoming data were to be organized in a convenient way. In the present code, no assumptions about the organization of the incoming data are made.

In general, the time needed per track in pattern recognition algorithms is proportional to the number of tracks to be reconstructed [3]. As can be observed, the total reconstruction time per track ranges from 550 to 360 $\mu$s per track for the present algorithm. In fact the dependence on the number of tracks is opposite to the naive expectation. The higher the number of tracks the faster per track the algorithm becomes. As is obvious from the plot, this behavior is driven by the setup time.

The CPU time required per track as a function of the number of layers in presented in Figure 2.b for events with 1000 tracks. As can be observed, the time required to reconstruct a track with 40 layers is less than a factor of 2 larger than the time needed for a 10 layer track. This indicates that a large fraction of the time is utilized to start a track. Once a good track is found, its extension is not very time consuming.

Each reconstructed track is associated with the generated track with which it has the most space points in common. A generated track is considered well reconstructed if either of the following conditions are fulfilled:

1. its associated reconstructed track contains at least 80% of the space points generated by it.

2. the difference between its momentum and that of the reconstructed track is less than 10%.

The reconstruction efficiency obtained following the criteria above is shown in Figure 3 versus $p_t$ for three sets of events with 10, 100 and 1000 tracks respectively. In this case, a detector with 10 layers is considered. As can be seen an efficiency of 98% or larger is obtained over the whole range of transverse momentum. The statistical error per bin is of the order of 2%. When averaging over $p_t$, a reconstruction efficiency of 99.7$\pm$0.07 % is obtained for 1000 tracks per event. The same efficiency within errors is obtained for the other two cases.

In order to test the ability of the code to handle low efficiency detectors, events with 90% hit efficiency are also studied. Hit efficiency is defined as the probability that a track crossing a layer produce a space point. For events with 1000 tracks and 10 layers, the overall track finding efficiency is only reduced by 2%.

As mentioned above, the previous results are obtained with the track search performed in two passes. In the first pass a vertex constraint is imposed. The second pass looks to form tracks with the remaining hits without assumptions about their vertex. The main difference between both methods is the momentum resolution. When a primary particle is reconstructed with the vertex constraint, the momentum obtained is much more accurate than when this piece of information is ignored. This is illustrated in Figure 4, where the momentum resolution is shown versus $p_t$ for tracks reconstructed with and without vertex constraint in events with ten layers. The results do not depend on the number of tracks per event.

# 4 Conclusions

A very fast algorithm has been developed to reconstruct tracks from a set of space points. The time required to reconstruct tracks with 10(40) space points is 330(550) microseconds in events with 100-1000 tracks. This very fast time is achieved maintaining a track finding efficiency above 98% and a good momentum resolution.

# 5 Acknowledgments

I would like to thank Dirk Schmischke and Michael Levine for his comments and support in developing the code and Geary Eppley for reading this manuscript. This work was supported by DOE under grant DE-FG03-92ER40772.

# References

[1] STAR Conceptual Design Report, LBL PUB-5347, June 1992.

[2] This is an extension of a similar well known transformation. See for example: M. Regler and R. Fruhwirth, *Reconstruction of Charged Tracks* (Vienna, OAW). PRINT-90-0279 (VIENNA).
From *Techniques & Concepts of High Energy Physics V.*. Ed. by Thomas Ferbel, Plenum Publ., 1990. For the 65th birthday of Otto Hittmair.
In *St. Croix 1988, Proceedings, Techniques and concepts of high-energy physics* 407-499.

[3] See for example the conventional method presented as comparison in
Georg Stimpfl-Abele and Lluis Garrido, Comput. Phys. Commun. **64** (1991) 46.

| | |
|---|---|
| SPECint92 | 70.0 |
| SPECfp92 | 121.0 |
| MFLOPS-DP | 25.9 |
| MFLOPS-SP | 36.7 |

Table 1: Characteristics of the IBM RISC 6000/370 used to test the time performance of the fast tracking program.

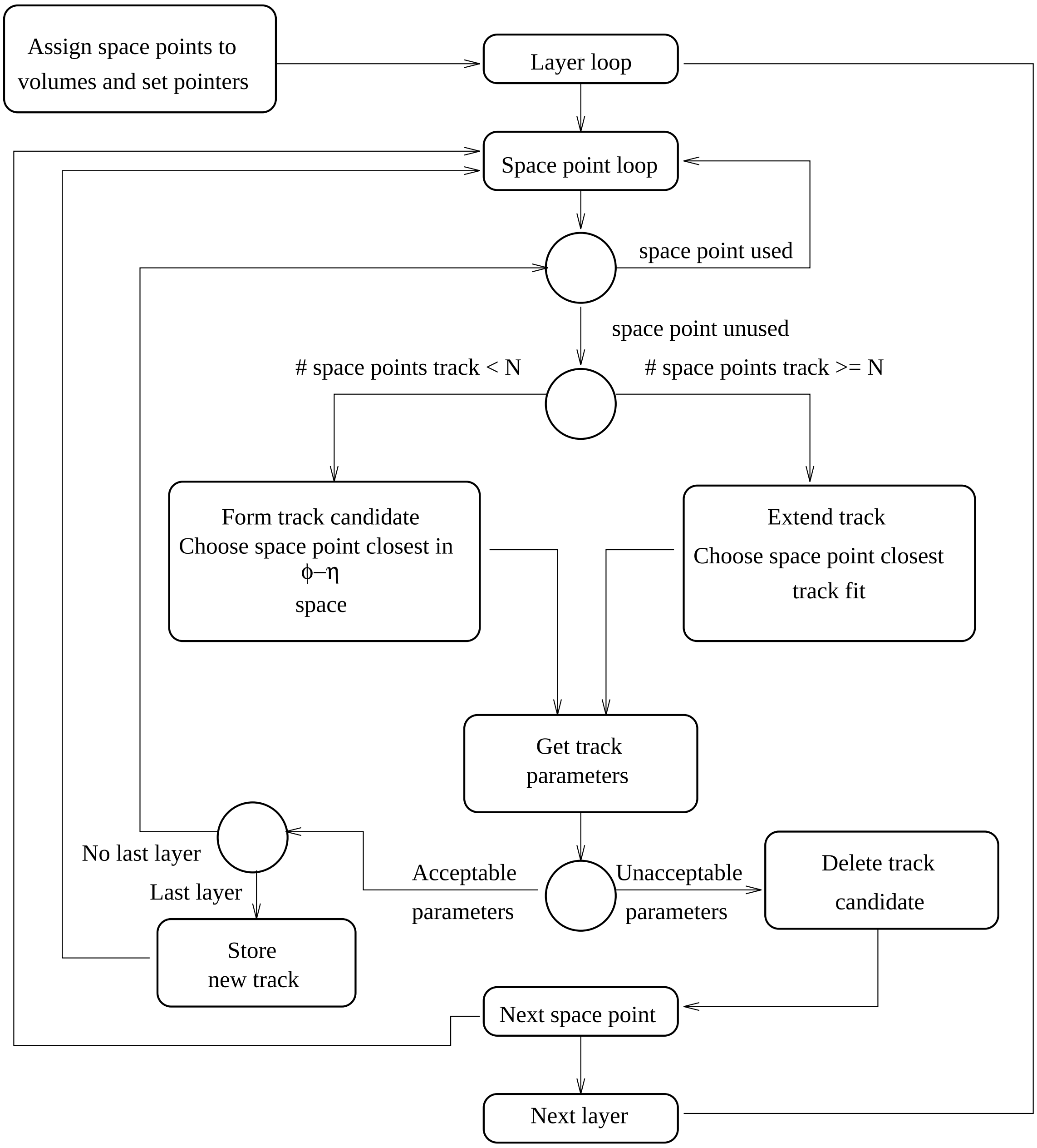


Figure 1: Flow Diagram of the fast tracking code.

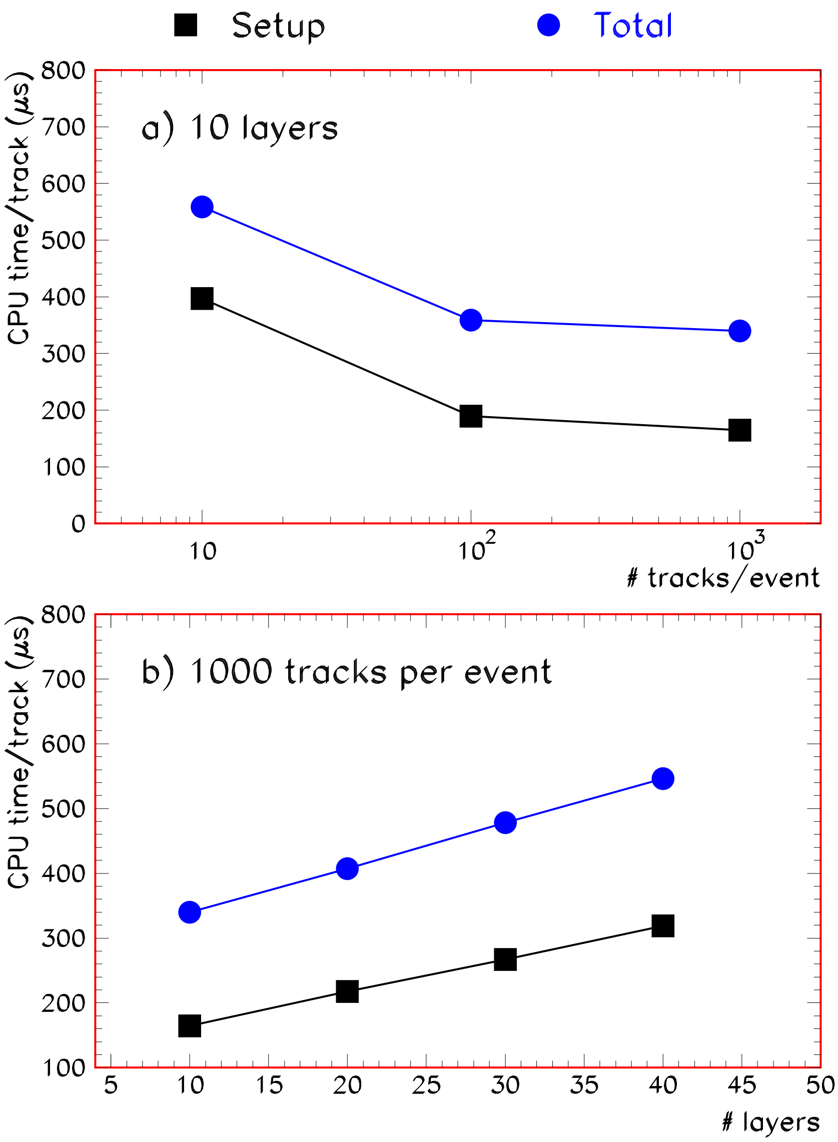


Figure 2: a) Total and setup CPU times per track as a function of the number of tracks per event for a detector with 10 layers. b) Total and setup CPU time per track as a function of the number layers in the detector for events with 1000 tracks. Times are measured on an IBM rs6000/370 machine.

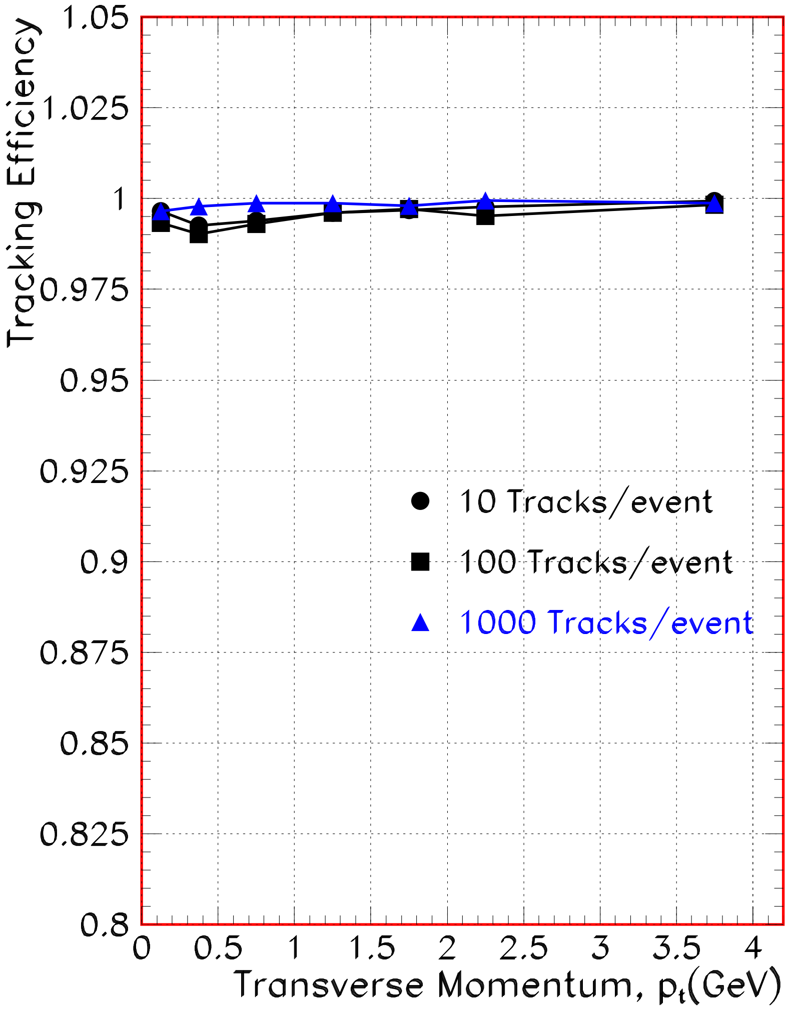


Figure 3: Track finding efficiency as a function of transverse momentum for a detector with 10 layers and events with 10, 100 and 1000 tracks. Statistical errors per bin are 2%.

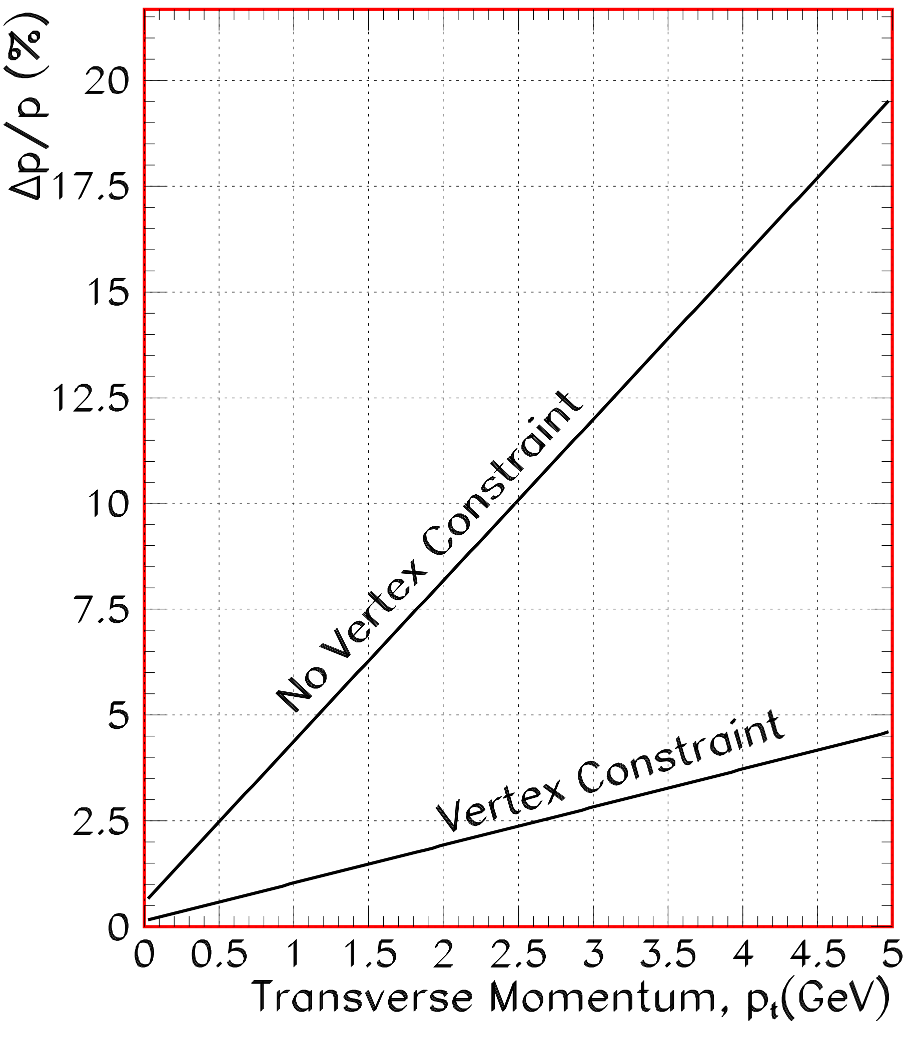


Figure 4: Transverse momentum resolution with and without vertex constraint as a function of transverse momentum for a detector with 10 sensitive layers.