SLAC–PUB–8389
Mar 2000

TRACKING IN FULL MONTE CARLO DETECTOR SIMULATIONS OF 500 GeV e^+e^- COLLISIONS [a]

M.T. RONAN
Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA
and
Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309, USA

In full Monte Carlo simulation models of future Linear Collider detectors, charged tracks are reconstructed from 3D space points in central tracking detectors. The track reconstruction software is being developed for detailed physics studies that take realistic detector resolution and background modelling into account. At this stage of the analysis, reference tracking efficiency and resolutions for ideal detector conditions are presented.

1 Introduction

High performance detectors are being designed to carry out precision studies of e^+e^-annihilation events in the energy range of 500 GeV to 1.5 TeV. Physics processes under study include Higgs mass and branching ratio measurements, measurements of possible manifestations of Supersymmetry (SUSY), precision Electro-Weak (EW) studies and searches for new phenomena beyond our current expectations. The relatively-low background machine environment at future Linear Colliders will allow precise measurements if proper consideration is given to the effects of the backgrounds on these studies. In current North American design studies, full Monte Carlo detector simulation and analysis is being used to allow detector optimization taking into account realistic models of machine backgrounds.

In this paper the design of tracking software that is being developed for full detector reconstruction is discussed. In this study, charged tracks are found from simulated space point hits allowing for the straight-forward addition of background hits and for the accounting of missing information. The status of the software development effort is quantified by some reference performance measures, which will be modified by future work to include background effects.

2 Linear Collider Detector Simulations

The current Linear Collider Detector (LCD) simulation effort on detailed Monte Carlo tracking and event reconstruction studies is focused on two detector design options, known as Large and Small [1]. The Large detector is based on a central Time Projection Chamber (TPC) tracking system with a 3 Tesla superconducting

[a]Presented at the International Workshop on Linear Colliders, April 28 - May 5, 1999 Sitges, Barcelona, Spain

Work supported by Department of Energy contracts DE–AC03–76SF00515 (SLAC) and DE–AC03–76SF00098 (LBNL).

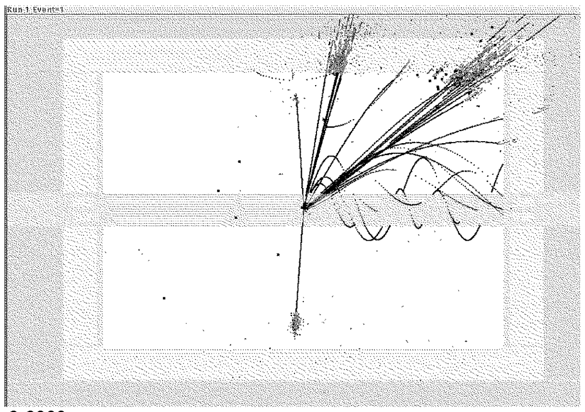

Figure 1: Display of a simulated 500 $GeVe^+e^- \to W^+W^-$event for the Large TPC detector model in the (y-z) non-bend plane. The first-level pattern recognition algorithm described in this paper finds helical tracks originating from the origin from the simulated 3D space points. Note that the spiraling low-p_t tracks would be absorbed by beam line magnets not included in these simulations.

coil mounted outside of Electromagnetic (EM) and Hadronic (HAD) calorimeters. The Small silicon tracking detector has a 6 Tesla field with the coil located inside of the Hadronic (HAD) calorimeter. Both detector models include an inner vertex detection system based on similar CCD designs, and an outer muon identification system. Complete parameter lists for the Large and Small reference detector models are described elsewhere [1,2]. Here we summarize the relevant parameters of the tracking systems.

2.1 Large "TPC" Detector Model

The simulated TPC tracking detector has resolutions of 140 microns in r-ϕ and 1.4 mm in z. The TPC inner radius is 25 cm, the outer radius is 200 cm and the TPC half-length is 290 cm. The magnetic field is taken to be 3 Tesla. The CCD detector has 5 layers starting at an inner radius of 2.0 cm with a measurement resolution of 5 microns. A simulated event for this detector model is shown in Fig. 1.

2.2 Small "Silicon" Detector Model

The simulated Silicon tracking detector has resolutions of 10 microns in r-ϕ and in z. The inner radius of the central tracker is 14 cm, the outer radius is 75 cm and the half-length varies from 31 to 89 cm. The magnetic field is taken to be 6 Tesla. The CCD detector has 5 layers starting at an inner radius of 1.0 cm with a measurement resolution of 5 microns.

2.3 LCD Simulation System

The GISMO C++ Monte Carlo tracking package is used to model the detectors. The GISMO program reads generated physics events from StdHEP format files, simulates hits in the various detectors without resolution smearing and writes out a portable ASCII format data file. A separate Java program reads these files and writes fully formated object data files for input into the JAS / hep.lcd reconstruction and analysis framework [3]. A **Driver** module in the LCD analysis system, **SmearDriver**, smears the points with the appropriate detector resolution functions.

3 Tracking Code

The overall Java reconstruction and analysis software design [3] emphasizes flexibility and extensibility in following object-oriented (OO) design rules. The hep.lcd framework defines **Drivers** to control the data processing sequence and modular **Processor** objects to carry out the actual event processing. A software design goal is to provide a light-weight tracking package that can be easily adapted to different detector designs. Currently, the hep.lcd.recon.tracking package contains full pattern recognition and preliminary track fitting software, including simple vertex detector hit association. In the Java OO design, track reconstruction interfaces are specified for the **Tracker**, **TrkFinder**, **TrkFitter** and **VertexDetector** objects.

3.1 Tracker

A base implementation of the **Tracker** is provided by an abstract **AbsTracker** class implementing the **Processor** methods. Detector specific classes, such as **TPCReco** and **SiliconReco** for the Large and Small detector options, respectively, extend the base class in specifying the dimensions and parameters of the different trackers.

AbsTracker $\rightarrow$ Tracker

The LCD processor **AbsTracker** picks up the simulated object-formatted **LCDEvent** data, reorders the 3D **TrackerHits** by layer and passes them to the **TrkFinder** pattern recognition software. The **AbsTracker** invokes the **TrkFinder**'s findTracks method to perform the pattern recognition. The implemented **Tracker** public methods include:

- setupParameters - sets the number of layers in the tracking detector and the number of levels of pattern recognition to be performed, and then executes the TrkFinder's setupFinder method to set the pattern recognition strategy.
- getNTracks - returns the number of tracks found.
- getTrkParam(int) - returns a double array of the parameters for the given track number.
- getNPoints(int) - returns the number of points on the given track number.

- getTrkPoint(int,int) - returns a double array of the x,y,z coordinates of the given point on the given track.

3.2 Track Finding

AbsTrkFinder → TrkFinder

The **AbsTrkFinder** class implements common methods for returning track information, leaving the actual pattern recognition to concrete implementations.

TPCPat2 → AbsTrkFinder → TrkFinder

The track finding is accomplished in a 3D pattern recognition class, **TPCPat2**, using various triplets of layers to find helical tracks that originate from the origin and satisfy minimum p_t requirements. Graded levels of pattern recognition are employed to optimize tracking efficiency, whereby large-angle high p_t tracks are found first, followed by smaller angle tracks and then lower p_t tracks in different angular regions.

The track finder contains parameters that limit the number of tracks, MaxTracks, and the number of points on a track, MaxTrkPts. The pattern recognition strategy (**Strategy**) is defined by the number of levels, NLevels, the minimum radius of curvature for each level, minRadius[], the number of patterns, NPatterns, and the layer numbers, triplets[][], and number of points to require, NPoints[], for each pattern. The following public methods are implemented:

- setupFinder - initiates a call-back through the setupParameters method to set pattern recognition strategy.
- setupParamters - implemented by subclasses to set parameters then to call back the setupStrategy method.
- setupStrategy - sets up the pattern recognition strategy defined by the number of levels, the minimum radius of curvature for each level, the number of patterns, and the layer numbers and number of points to require for each pattern.
- findTracks - at each level, it uses the 3D points to find tracks for the specified minimum radius of curvature. It finds all possible tracks passing through points on the triplet of layers used for each pattern.
- assignPoints - searches through the given set of points to find points on the trajectory of the selected track.

Basically, the track finder uses all points in a number of layer triplets to find tracks that originate from near the beam-beam interaction point. In the process it varies the minimum radius of curvature allowed as it steps through a number of levels of pattern recognition. At each level, all points in three different layers are considered. Pairs of points in the inner and outer pivot layers are used to restrict the track's origin before searching through points in the third layer. Points in the middle verification layer determine the radius of curvature (xy plane) of possible tracks, and helical position (z deviation). Tracks above the minimum radius of curvature with small z deviation from an ideal helix are formed from unused points in all layers that are within tolerance of the determined trajectory. If the minimum number of hits required is met then the track is saved and its points are flagged as being used.

3.3 Vertex Detector

CCDReco [Processor] $\rightarrow$ **VertexDetector**

The present level of track reconstruction includes associating points in the vertex detector to the found tracks. The association is implemented for a 3D CCD vertex detector in the hep.lcd.recon.tracking.ccd package. The processor class, **CCDReco**, specifies the vertex detector option and accesses the **VXDHits** from the **LCDEvent** data and orders them by layer. It uses the **Tracker** assignPoints method to associate hits in the vertex detector with the ideal helical trajectories calculated for the found tracks. These points are later accessed in forming track candidates. The following **VertexDetector** methods are implemented

- getNPoints(int) - returns the number of vertex detector points on the given track number.
- getTrkPoint(int,int) - returns a double array of the x,y,z coordinates of the given vertex detector point on the given track.

New code for combined vertex detector and tracker pattern recognition are presently being tested.

3.4 Track fitting

Several track fitters have been modified to work in the Linear Collider Detector simulation environment: a SLD Weight Matrix fitter and a Kalman fitter. However, because of a lack of time, track fitting was not used for the results presented at Sitges, Barcelona 1999. The track momentum resolution obtained from a set of 3 averaged points in the Tracker/Vertex detector, as described for track candidates below (Sec. 3.5), was adequate in outlining the tracking effort and for presenting preliminary results.

3.5 Reconstructing tracks

Track candidates

A track candidate **TrkCandidate** object is created from the tracker and vertex detector hits for each track found. It contains a candidate flag, the track number, the list of hits on the track (**TrkHotList**), the track parameters (**TrkParams**), the extrapolated positions of the track (**TrkExtrap**) and the overall chi-squared. The getTrackHits method selects up to 9 hits at the middle and inner/outer extremes of the track to be averaged. The average position and reduced error are used to create average points (**TrkHitOnTrk**) for improving the determination of the track's parameters. The recalculateTrack method uses these average points to recalculate the parameters of an ideal helical trajectory, calculates the chi-squared for the track and returns the result of a test on the quality of the track. The extrapolated positions at the entrance and exit of the outer detectors are determined by swimming the track through the magnetic field to the middle of the magnet coil then extrapolating to the outermost detectors (using the **HelicalSwimmer**). In summary, the **TrkCandidate** methods are

- getTrackHits - returns an array containing average inner, mid and outer points on the track.
- recalculateTrack - given the detector specifications, it uses the array of 3 points on the track to recalculate the parameters of an ideal helical trajectory, calculates the chi-squared for the track and returns the result of a test on the quality of the track.

Track parameters

Standard parameters are chosen for describing a track. The five parameters (**Trk-Params**) are:

- d_0 [cm]- the closest distance in the x-y plane from the origin to the orbit.
- ϕ_0 - the azimuth corresponding the track direction in the x-y plane.
- k - the curvature of the track. The magnitude of k = 1/p_t in $(\mathrm{GeV/c})^{-1}$.
- z_0 [cm] - the z position of the orbit.
- s = tanLambda - the slope of the track (tangent of the dip angle).

AddReconTrks [Processor]

The LCD processor **AddReconTrk** creates the track candidates for each track found by the pattern recognition code. After the track candidate's parameters have been updated and an improved chi-squared has been calculated, a cut on the quality of the track is made. A reconstructed track, **ReconstructedTrack**, implementing the **hep.lcd.event.Track** definitions[3], is created for each selected track and added to a track list, **ReconTrackList**, to be put into the **LCDEvent** record.

TrackReco [Driver]

A LCD track reconstruction driver **TrackReco** is used to execute the track finding and fitting processes described above. The present performance of the track reconstruction code for a 300 MHz processor running Java JDK 1.1.8 is 1-3 sec/event for 500 $GeV e^+e^- \rightarrow W^+W^-$events. The reconstruction and analysis is performed on a Windows NT server (**sldnt0.slac.stanford.edu**) from clients running on Solaris or Linux platforms.

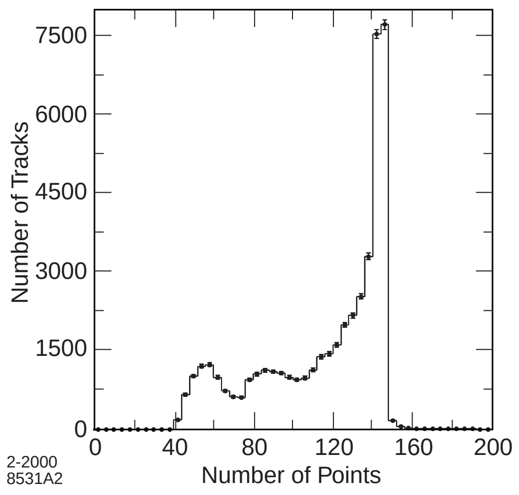

Figure 2: Number of points on tracks found in Large TPC detector simulations. The peak at 144 points corresponds to large angle tracks hitting all of the TPC layers.

4 Tracking Analysis

4.1 Analyzing track distributions

The analysis of tracking distributions provides an important check on the code. An example is given in Fig. 2 where the number of points on TPC tracks is displayed for a sample of reconstructed 500 GeV $e^+e^-\to W^+W^-$events. For large-angle high p$_t$ tracks, one finds the expected peak at the number of tracking layers. Smaller angle tracks passing out of the endcap region are picked up by tracking strategies involving different TPC layers.

Additional analysis checks were performed on both detectors.

4.2 Event scans

A simple event display plugin (**hep.lcd.plugin.LCDPlugin**) was written for the JAS/hep.lcd analysis framework[3]. Scans of simulated 500 GeV $e^+e^-\to W^+W^-$and $e^+e^-\to t\bar{t}$ events, such as the one shown in Fig. 1, found that the tracking code maintains its high efficiency pattern recognition even in high multiplicity environments. In the scans, a small contamination of fake tracks was found that will be easily eliminated in the next software development phase.

4.3 Tracking Efficiency and Resolutions

Tracking efficiencies and resolutions can be measured by comparing reconstructed tracks with the Monte Carlo generator.

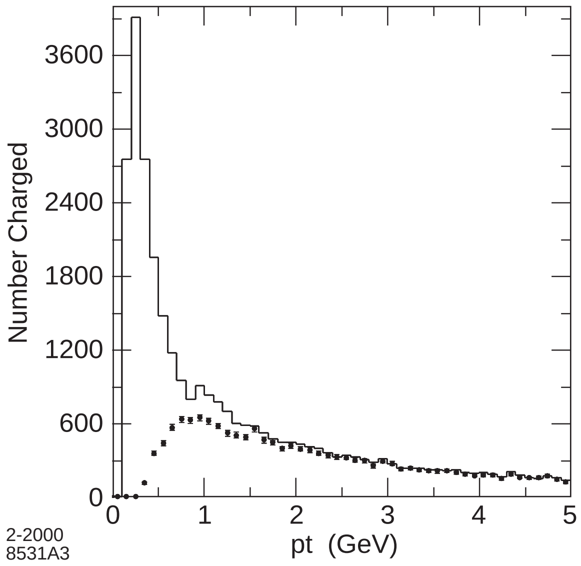

Figure 3: Tracking efficiency curves for the Large detector model. The solid histogram displays the transverse momentum distribution for all Monte Carlo charged tracks within the detector acceptance ($|cos(\theta)| \leq 0.95$). The data points display the same distribution for reconstructed tracks. The ratio of the two curves gives the tracking efficiency which is not shown.

SimpleAnlReconTrks [Processor]

A simple track analysis processor **SimpleAnlReconTrks** compares reconstructed tracks with the original Monte Carlo charged particles. The processor searches through all of the MC charged particles and selects the closest in angle (θ) to the reconstructed track requiring that $cos\theta \geq 0.99$. If no MC particle is found then the reconstructed track is considered to be mismeasured or a fake track.

Tracking Efficiency

Since the present tracking is limited in angle coverage and transverse momentum range by the central tracker and the high magnetic field, inefficiencies at low p_t are expected. Fig. 3 shows the number of reconstructed tracks and charged MC particles as a function of p_t. One observes that the efficiency reaches nearly 100% only above 1-1.5 GeV/c. Variations of the tracking strategies to concentrate on inner tracker layers improves the low p_t efficiency significantly at the cost of increased reconstruction time, and may not stand up to high background conditions. Combined vertex detector and tracker pattern recognition algorithms are being persued to reach high efficiency for very low p_t tracks while physics studies have been initiated based on the present "first-pass" algorithms.

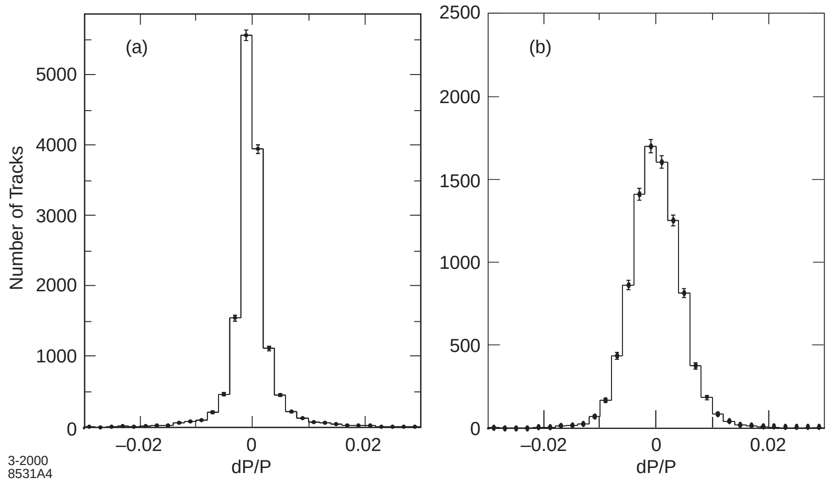

Figure 4: Large(a) and Small(b) detector momentum resolutions based on ideal helical trajectory determination, with no track fitting, from the first-level pattern recognition algorithm.

Momentum Resolution

Figure 4 shows the momentum resolution for reconstructed tracks based on the difference in momentum of their corresponding Monte Carlo particles. At this stage of the analysis, with no track fitting, the average momentum resolutions for the two detector designs are $\sim 1.5 - 4 \times 10^{-3}$. Here the difference between the two detector designs is a result of the larger multiple scattering at low momentum for the Small detector. Actual track fitters are being developed to better characterize the performance of the different detector models.

Energy Loss

From the average difference in energy between the reconstructed track and the Monte Carlo particle, one finds the energy loss in the inner detector material to be $\sim 2.0\ MeV$ for both detector designs.

5 Plans

Basically, the plan is to continue to optimize the pattern recognition and track fitting for this stage of the analysis. That is, high-level optimization of the full reconstruction system may well be beyond the scope of design level studies. Techniques for mixing reconstructed tracks with parameterized simulations of missing Monte Carlo particles are being developed[4] to allow precision physics studies to be made. As the physics and detector issues are better defined, detailed studies of the variation in performance of the detectors for different effects will be determined by understanding the sensitivity to deficiencies in the reconstruction (e.g. the effect of missing low p_t or small angle tracks) and by controlling the mixing of reconstructed and Monte Carlo particles that are missed[4].

5.1 Background simulations

Detailed machine background simulations have been made for the most significant sources of backgrounds. These backgrounds will be superimposed on the simulated tracker hits before executing the pattern recognition software.

Acknowledgments

The track finding code in **AbsTrkFinder** was derived from **PAT2**, written by Henri Videau and modified by Gerry Lynch and Orin Dahl for PEP4-TPC tracking. It was modified for BaBar Silicon vertex detector standalone tracking by Gerry Lynch. A C++ version, **BaBarPat2** written by Natalia Kuznetsova, was ported to Java by the author.

I'd like to thank Tony Johnson and Nick Sinev for numerous comments and checks of the Java track reconstruction code.

References

1. J. Brau, *Overview of the American Detector Models*, in these proceedings.
2. R. Dubois, *LCD Small and Large Calorimeter Single Particle Resolutions*, in these proceedings.
3. M.T. Ronan *et al.*, *Java Analysis Studio and the hep.lcd Class Library*, in these proceedings.
4. M.T. Ronan, *A Hybrid Monte Carlo System for Detector Simulations*, in preparation.