Pattern Recognition and Track Fitting in Central Trackers

Norman A. Graf

Stanford Linear Accelerator Center

Abstract. We present a fully object-oriented framework which provides the tools for track finding and fitting in a particle physics detector. The software is modular and extensible so that it can be applied to a wide range of detectors and can be finely tuned to optimize performance. We believe that this package can provide a common solution to the requirements of the Linear Collider simulation community.

INTRODUCTION

Reconstructed tracks are an important component in identifying all of the elementary particles in a physics event: electrons require the presence of a track with momentum matched to the energy of a calorimeter cluster, photons require the absence of such a match, muons require a track matching to a minimum-ionizing trace in the calorimeter or a track in tracking chambers outside the calorimeter, taus are narrow jets with one or three matching tracks, heavy quarks are tagged by tracks arising from secondary vertices within jets, etc. The tracking system is tasked with finding and reconstructing the trajectories of charged particles with high efficiency and precision over a wide range of track momenta and track densities.

Designing central tracking detectors for high energy and high luminosity linear colliders requires sophisticated analysis tools to be available to answer detailed questions regarding pattern recognition and momentum resolution. Track finding is, however, one of the most challenging pattern recognition problems in the processing of data from a particle physics detector and mature software is traditionally not available when design decisions are being made. Efficient use of resources is also an essential part of this task, both computing cycles during processing and (perhaps more importantly) physicist's time during development. The process of identifying energy deposits in various tracking detectors and using these signals to reconstruct the trajectories of the original charged particles lends itself naturally to an object-oriented implementation. We describe here an existing fully object-oriented framework which provides the tools for track finding and fitting in a particle physics detector. We believe that the linear collider tracking community would benefit by adopting this package as a common solution.

Work supported by the Department of Energy contract DE-AC02-76SF00515.

Stanford Linear Accelerator Center, Stanford University, Stanford, California, 94309

PACKAGE COMPONENTS

Tracks

We define a reconstructed track as a list of hits deduced from energy depositions in a detector and one or more kinematical fits based on these signals. In principle, six parameters are required to determine a charged particle's path in a magnetic field. However, specifying these parameters at a single point (e.g. the distance of closest approach to the beam (DCA) is insufficient for precision fits due to material effects such as energy loss (dE/dx), multiple Coulomb scattering, and bremsstrahlung, and the effects of magnetic field inhomogeneities. There is, in practice, no global functional form for the fit: we use instead optimal fits at each surface. These fits themselves are referred to as tracks, as they are composed of a surface, a set of track parameters at that surface, and a corresponding error matrix for those parameters. Typically, there are five track parameters (two specifying the track position on the surface, two characterizing the direction and one determining the momentum) and one parameter provided by the constraint that the track lie on the surface. It should be noted, however, that this number can be different; the momentum parameter may be unnecessary in the absence of a magnetic field, or if additional measurements are involved, such as time or energy, they may be included.

Clusters and Hits

Clusters represent the observed signals from detector elements. Signals from nearby particles may not be separable and thus be merged into a single cluster. The measurements used to fit tracks are derived from hits. A cluster takes a track as input and returns a list of viable hits. A hit returns a measurement of any dimension along with an error matrix for that measurement. A hit can also provide a prediction of the measurement from the track along with its error matrix, and therefore the difference between the prediction and the measurement and the derivative of the prediction with respect to the track parameters. This is all the information required to fit the track with the hit.

Surfaces and Layers

A Surface describes an (N-1)-dimensional surface in the N-dimensional track parameter space. In practice, this almost always reduces to an ordinary two-dimensional surface in the usual three-dimensional coordinate space. Surfaces come in two flavors: pure surfaces which are unbounded and bounded surfaces. Pure surfaces have the job of giving meaning to the parameters in a track vector. They provide methods to return the position and direction of the track and the difference between two track vectors (accounting for circular variables). Bounded surfaces always inherit from a pure surface, and provide additional methods to return whether a specified track vector falls within their bounds. Several concrete surfaces covering

most of the detector geometries typically found in high energy collider detectors are provided in the base distribution.

We define a Cylinder as coaxial with the z axis and is therefore specified by a single parameter representing the radius of the cylinder. A bounded Cylinder adds a minimum and maximum extent in z. This surface would naturally be used for central drift chambers or other barrel-shaped detectors. We currently support pure axial and stereo hits (e.g. drift chamber or scintillating fiber detectors) as well as two-dimensional hits (e.g. TPC) on this surface.

An XYPlane is defined as a planar surface parallel to the z axis. It is specified by a distance from the axis and an angle with respect to the x axis. A bounded XYPlane adds rectangular boundaries to this plane. Silicon wafers in a vertex detector or silicon strip tracking detector would be modeled with this surface. Axial, stereo and 2D hits are supported.

A plane perpendicular to the z axis is represented as a ZPlane, and specified by a single parameter z, the distance from the origin. Silicon wafers in a disk configuration or straw tubes in a forward tracking detector would be modeled with a bounded ZPlane, which introduces arbitrary closed, concave boundaries on the plane. Again, both one and two-dimensional hits are supported.

The final surface is the distance of closest approach, DCA, defined to be where the track direction is normal to the track position in the x-y plane. Unlike the other surfaces discussed, it is not a 2D surface in 3D space but is unique to each track.

Layers describe the geometry of the detector by holding surfaces, either directly or through sub-layers. Each cluster in an event is associated with a surface in the layer. A Detector is built from a collection of Layers which again may be organized in a hierarchy of detectors.

Propagators and Interactors

Propagators propagate track parameters and their errors from one surface to another. In general, they will account for the magnetic field and the effects of material interactions along the way. Propagators between and amongst all of the defined surfaces are provided in the base distribution. Currently, however, only homogeneous fields are handled, and interactions with material are handled separately. Interactors are defined for abstract interactions, with concrete implementations provided for thin scatterers and gaussian energy loss.

Track Finding and Fitting

Tracks are found by propagation between and through layers. When a track reaches a surface with associated clusters, each of the nearby clusters is used to generate hits and each of these hits is used to extend the original track. The default extension proceeds via a Kalman filter update, allowing us to take the track parameters and their error matrix and optimally add the information from a hit and its uncertainty to generate a new set of track parameters and error matrix. This process of propagation and updating can be used to perform pattern recognition in the same loop as the track fitting, or a list of hits can be provided from some external pattern recognition stage

and simply fit. If a track encounters a surface without clusters, a Miss is added to the track. A Miss encapsulates the probability that no hit was found (representing for example the intrinsic efficiency of the detector, or the proximity of the track to a detector boundary), and at the end of the track finding one can cut on a combined miss probability instead of a fixed number of hits or misses.

SIMULATION

In order to test the various components of the toolkit, simulation tools were written during the development phase. These can now be used for fast simulations to model the response of various detector designs without having to write detailed GEANT-level descriptions. The detector can be assembled from the provided surfaces, and interactors added to each appropriate layer. Particles from a Monte Carlo event can be propagated to each detector element in turn, generating the appropriate hit at the intersection of the track with the surface. The track vector is smeared to account for multiple coulomb scattering and modified for energy loss, then propagated to the next element. The full pattern recognition and fitting software can then be run on the event, providing fast feedback in the development cycle.

SUMMARY

The TRF++ track finding and fitting toolkit described here was originally written in object-oriented C++ for use in the DØ experiment at the Fermilab Tevatron [1,2]. From its inception, however, it was intended to be experiment-neutral in its implementation. Much of the software represents abstract base classes, providing developers the opportunity to extend the base functionality. We propose that substantial time and effort could be saved by adopting this toolkit for Linear Collider simulations. Most of the code has also been ported to Java to plug into the Java Analysis Studio [3] which is currently the framework for full reconstruction for the North American Linear Collider Detector simulation group.

ACKNOWLEDGMENTS

I would like to acknowledge David Adams as the principal architect of the TRF++ toolkit and the major contributions of Slava Kulik to the planar extensions of the package. The author's work was supported in part by the U.S. Department of Energy under Contract DE-AC03-76SF00515.

REFERENCES

1. The TRF++ home page is http://www-d0.fnal.gov/~dladams/trf/ .
2. TRF++: an object-oriented framework for finding tracks, D.L. Adams et al. Computing in High Energy Physics (1998) Proceedings.
3. LCD Reconstruction and Analysis Tools, G. Bower et al., these proceedings.