Table of Contents

Ckf Trajectory Builder

Complete: 🟩

Goal of the page

This page is intended to explain the user the most relevant features of the Ctf Trajetory Building, including the working principles, the classes involved and the parameter configuration.

Contacts

BorisMangano, WolfgangAdam, GiuseppeCerati, GiovanniPetrucciani

Introduction

The Ckf Trajectory Builder is the core of the pattern recognition of the Combinatorial Track Finder: it is the algorithm dedicated to the search of the tracker hits associated to a track.

Software architecture

The generic trajectory builder is defined in two base, abstract classes, TrajectoryBuilder and BaseCkfTrajectoryBuilder. There are two concrete implementations of such classes: CkfTrajectoryBuilder and GroupedCkfTrajectoryBuilder. The first one is described in this page and includes most of the features used in the second. The "grouped" version of the trajectory builder is used by default in the standard track reconstruction sequence and is described in the page SWGuideGroupedCkfTrajectoryBuilder. Basically, it allows for finding more than one hit per tracking layer, thus accounting for geometric regions with overlapping sensors.

Usage of the algorithm

The configuration parameters are defined in CkfTrajectoryBuilderESProducer_cfi.py.

The trajectory builder algorithm, based on a Kalman Filter approach, works as follows: it proceeds iteratively from the seed layer, starting from a coarse estimate of the track parameters provided by the trajectory seed, and includes the information of the successive detection layers one by one. On each layer, i.e. with every new measurement, the track parameters are known with a better precision, up to the last point, where they include the full tracker information. Each iteration of the Kalman Filter is implemented in separate steps that are executed in the following order:

1. Navigation: given the track-candidate s updated state on a specific layer, a dedicated navigation service determines which layers of the tracking system, among the adjacent ones, are expected to be intersected by the extrapolated trajectory. In determining the compatibility of a layer surface with the path of the particle, the uncertainty on the trajectory s parameters is considered.
2. Search for compatible detectors: after the track-candidate state is propagated to the surface of a compatible layer, the list of layer s detectors that are compatible with the trajectory is determined. A detector is considered incompatible with the trajectory when the position of the trajectory s intersection with the detector surface is n sigmas outside the detector s bounds. The default value of n is 3, but this parameter is configurable (via the `estimator` parameter in the cfg file) and it can be changed if track reconstruction efficiency is preferred to the algorithm s speed or vice versa.
3. Search for compatible measurements: for each compatible detector, the direction of the trajectory on the sensor surface is used to calculate more accurately the drift of the ionization charge carriers inside

   the silicon bulk and to improve the estimation of the position of the pixel or strip clusters in the detector. A reconstructed cluster is considered compatible with the trajectory when the 2 of the predicted residual is smaller than a configurable cut (also this parameter is defined by the `estimator` parameter, default is equal to 30).
4. State update: if a measurement is compatible with the trajectory, the track-candidate is extended by this additional point and the trajectory s parameters are updated to the filtered state (the updating algorithm is configurable via the `updator` parameter in the cfg file). When several measurements on the new layer are compatible with the predicted trajectory, several new track-candidates are created, one per each compatible hit. Finally, one additional track-candidate is created, in which no position measured is added, to account for the possibility that the charged particle did not leave any hit on that particular layer. This fake hit is called an invalid hit .

All resulting track-candidates are then grown in turn to the next compatible layer(s) and the procedure is repeated until either the outermost layer of the tracker is reached or a stopping condition is satisfied. To avoid an exponential increase of the number of candidates, only a limited number of these are retained at each step, based on their normalized 2 and the number of valid and invalid hits. The maximum number of new candidates per step is defined in the `maxCand` parameter, default is 5). The stopping conditions are defined in TrajectyoryFilter_cff.py (see SWGuideTrajectoryFiltering also), included in the trajectory builder via the `trajectoryFilterName` parameter. The main stopping conditions are: `maxLostHits` (maximum number of invalid hits per track, default 1), maxConsecLostHits (maximum number of consecutive invalid hits per track, default 1) and maxNumberOfHits (maximum number of total hits per track, default 100). The TrajectoryFilter also defines the minimum number of hits per track (`minimumNumberOfHits`, default 5) and the minimum pT of the track (`minPt`, default 0.9). Such parameters can be overwritten is some IterativeTracking steps or during HLT tracking.

Ambiguity resolution

Ambiguities in track-finding arise when the same track is reconstructed twice starting from different seeds or when a given seed develops into more than one final track-candidate. The ambiguity among mutually exclusive track-candidates must be resolved in order to avoid that the same charged particle is counted twice, or even multiple times. The pattern recognition module of the CTF algorithms aims to identify track duplicates through an ancillary service, called trajectory cleaner, that determines the fraction of shared hits between two track-candidates. If this fraction exceeds the cut value of 50%, the trajectory cleaner removes the track with the least number of hits from the candidates collection; if both tracks have the same number of hits, that with the highest 2 value is discarded. The procedure is repeated iteratively until all the pairs of track-candidates in the collection shares less than half of their hits. For more details, see SWGuideTrajectoryCleaning.

Review status

Reviewer/Editor and Date (copy from screen)	Comments
GiuseppeCerati - 17 Mar 2009	created template page

Responsible: BorisMangano
Last reviewed by: Most recent reviewer

This topic: CMSPublic > SWGuideCkfTrajectoryBuilder
Topic revision: r3 - 2009-11-16 - GiuseppeCerati