LCFIPlus Tutorial

T. Suehara (Kyushu U.)

https://github.com/lcfiplus/LCFIPlus

Publication of LCFIPlus

Nuclear Instruments and Methods in Physics Research A 808 (2016) 109–116

Contents lists available at ScienceDirect

Nuclear Instruments and Methods in Physics Research A

journal homepage: www.elsevier.com/locate/nima

LCFIPlus: A framework for jet analysis in linear collider studies

Taikan Suehara [a,*], Tomohiko Tanabe [b,**]

[a] Department of Physics, Faculty of Science, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan
[b] ICEPP, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

ARTICLE INFO

Article history:
Received 29 June 2015
Received in revised form
11 November 2015
Accepted 11 November 2015
Available online 29 November 2015

Keywords:
Linear collider
Flavor identification
Vertex finding
Jet finding

ABSTRACT

We report on the progress in flavor identification tools developed for a future e^+e^- linear collider such as the International Linear Collider (ILC) and Compact Linear Collider (CLIC). Building on the work carried out by the LCFIVertex collaboration, we employ new strategies in vertex finding and jet finding, and introduce new discriminating variables for jet flavor identification. We present the performance of the new algorithms in the conditions simulated using a detector concept designed for the ILC. The algorithms have been successfully used in ILC physics simulation studies, such as those presented in the ILC Technical Design Report.

1. Introduction

In high-energy collider experiments, the identification of the jet flavor plays an important role in event reconstruction. The dominant decay signatures of top quarks and Higgs bosons include bottom (b) jets. Many new physics scenarios, such as those in supersymmetric theories, have discovery signatures involving third-

Higgs boson $H \to b\bar{b}$, $c\bar{c}$, gg as well as flavor-changing neutral currents in the top sector such as $t \to cZ$ and $t \to cH$. Since charm hadrons have smaller masses and shorter lifetimes compared to bottom hadrons, charm hadrons are more difficult to identify, although the excellent point resolution in the vertex detectors to be employed l

An importa

NIM A 808 (2016) 109-116

Functions of LCFIPlus

- Vertex finder
  - PrimaryVertexFinder - Tear-down method
  - SecondaryVertexFinder - Build-up method
- Jet Clustering
  - Durham, Kt, Valencia, etc.
  - Clustering using vertex information
  - Removal of beam jets (by R parameter) also possible
- Jet Vertex Refiner
  - Connecting jets and vertices (can accept jets from external software)
  - Single track vertex finder
  - Combining vertices (to maximum 2 / jet)
- Flavor tagging
  - 3-class tagging: b, c, q (more categories possible)
  - Using boosted decision trees

Data/process flow

All in “lcfiplus” namespace

EventStore

singleton for data pool

vector<Track *>
vector<Neutral *>
vector<MCParticle *>
vector<Vertex *>
vector<Jet *>
any other types

- Automatic type identification
  (Allow one name with multiple types)
- Automatic creation/deletion
  (using ROOT class dictionary)

LCIO

LCIOStorer

- Automatic conversion from LCIO to lcfiplus classes (using hook in EventStore)
- Conversion to LCIO is manually invoked by LcfiplusProcessor

configuration

Algorithm

PrimaryVertex JetVertexRefiner
BuildUpVertex FlavorTag TrainMVA
JetClustering MakeNtuple ReadMVA etc.

- Parameters class used for type-safe configuration

LcfiplusProcessor

- Marlin processor
- Process Marlin parameters to be passed to Algorithm
- LCIO I/O configuration

Internal algorithms

Independent

Tai… …ware tutorial, 2021/8/1

Marlin

LCFIPlus framework (1)

- LCFIPlus data structures
  - Track, Neutral, MCParticle, Jet, Vertex, …
  - Original classes, converter to LCIO available
- LCFIPlus EventStore
  - Global pool for storing data structures
  - Name – object (or vector of objects) map
  - Any classes with ROOT dict can be stored
    - Dynamic creation/destroy with Tclass
    - Might have some compatibility issues with ROOT6
  - Export/Import to LCIO/Marlin

LCFIPlus framework (2)

- Algorithms
  - PrimaryVertex, BuildUpVertex, JetClustering etc.
  - Dynamically loaded algorithms classes (with TClass)
- LcfiplusProcessor
  - Marlin processor to interface Marlin/LCFIPlus
  - Marlin parameters (in steering xml file) are automatically converted to LCFIPlus parameters
  - Multiple algorithms can be loaded
  - Multiple LcfiplusProcessor allowed
    - LCIO structures loaded just once

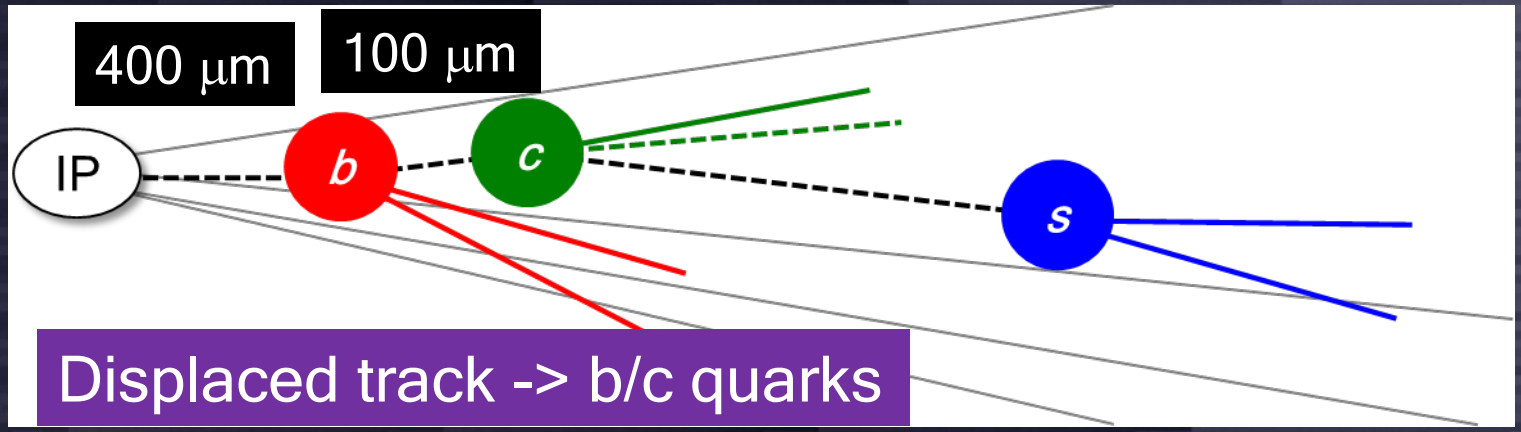
LCFIPlusConfig

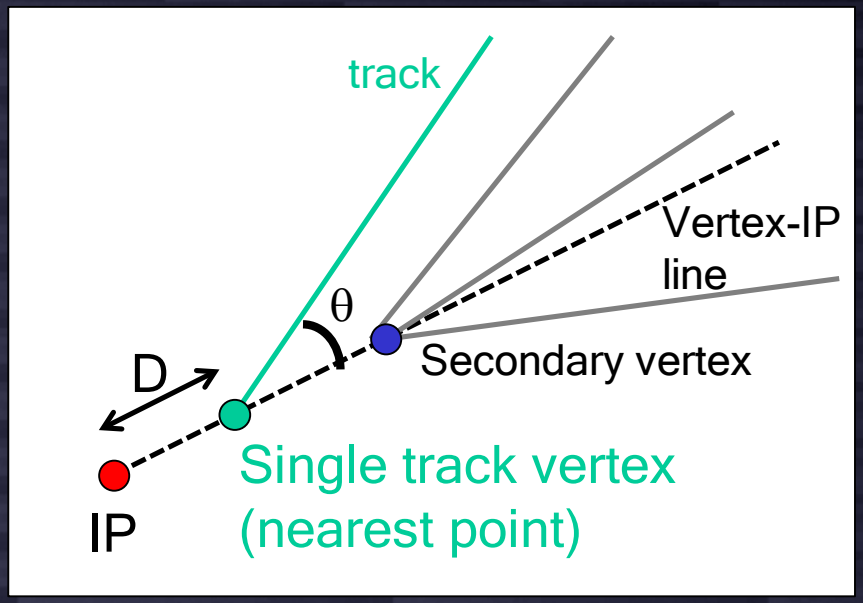
- ILDConfig
  - Configuration files for ILD reconstruction
  - https://github.com/iLCSoft/ILDConfig
- Under ILDConfig/LCFIPlusConfig
  - steer/: Marlin steering files (xml)
  - vtxprob/: vertex probability file
    (used for flavor tagging: unless you create own)
  - lcfiweights/: weights for BDT of flavor tagging
    (not necessary if you train flavor tagging
    with your favorite configuration)

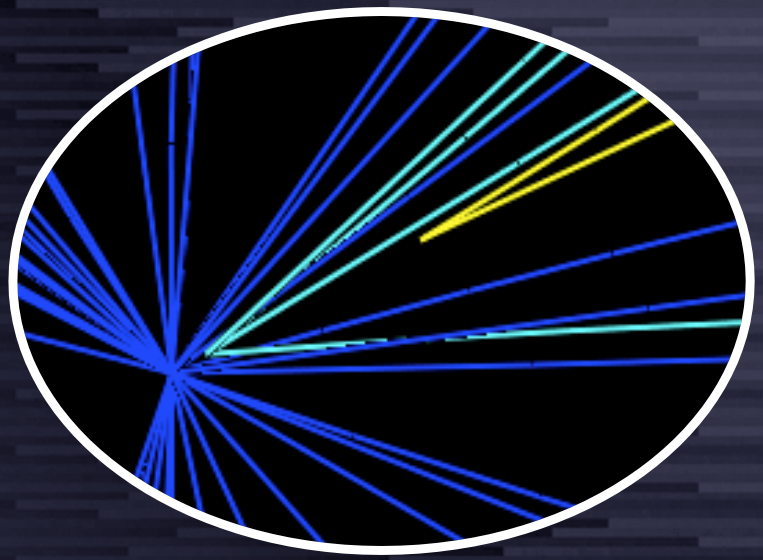
Hands on today

- Vertex finder
- Jet clustering
- Flavor tagging
  - Track ntuple (skip today)
  - Make ntuple
  - Training
  - Flavor tagging

Vertex finding







MC event

Vertex finding algorithms

- PrimaryVertex
  - Tear-down method
    - Fit all tracks with beam constraint
    - Remove a track with worst chi2
    - Repeat until all tracks within acceptable chi2
- BuildUpVertex
  - Secondary vertex finder to be run after PrimaryVertex
  - Build-up method
    - Examine every track pair if it is consistent with a vertex (chi2 etc)
    - List “good” vertex candidates
    - Try to attach more tracks to the candidates
  - “Adaptive vertex fitter” recently added (optional)

Hands on today

- Vertex finder
- Jet clustering
- Flavor tagging
  - Track ntuple (skip today)
  - Make ntuple
  - Training
  - Flavor tagging

Jet Clustering: algorithms

- Durham with beam rejection

$$y_{beam} = 2E^2\alpha^2(1-\cos\theta)/E_{vis}^2$$

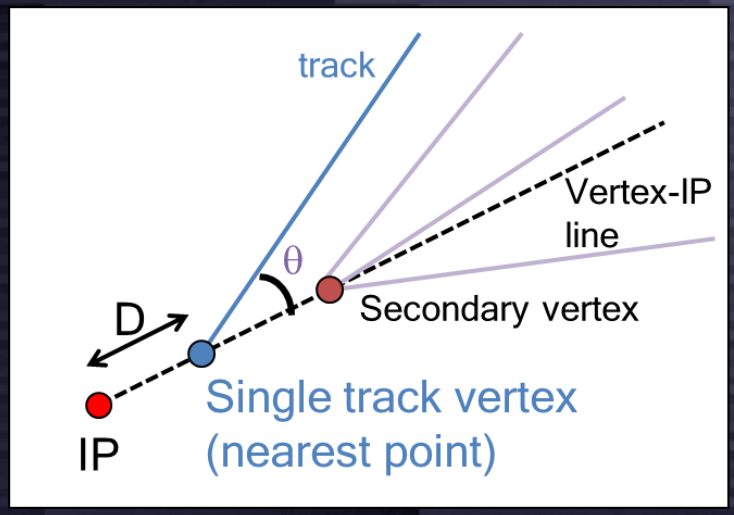
- Plain Durham (still available, of course)
- kT algorithm
  - No need to run it outside any more
- Valencia algorithm
  - Intermediate algorithm of Durham ant kT

$$d_{ij} = min(E_i^{2\beta}, E_j^{2\beta})(1-\cos\theta_{ij})/R^2$$

$$d_{iB} = p_T^{2\beta}$$

Algorithms (2)

- JetVertexRefiner
  - Assign vertices to jets if jets are created externally
  - Looking for “single track vertices”
  - Limit vertices/jet to two, by forced combining of existing vertices if # vtx > 3 (based on χ^2)
  - “Bness” recently added

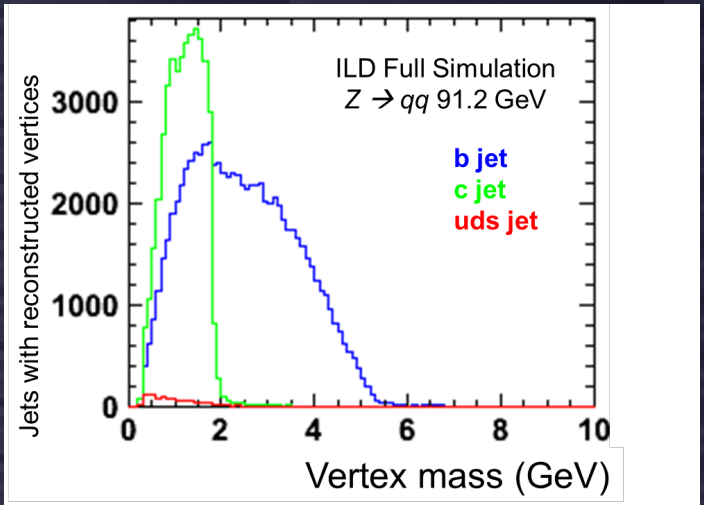


Hands on today

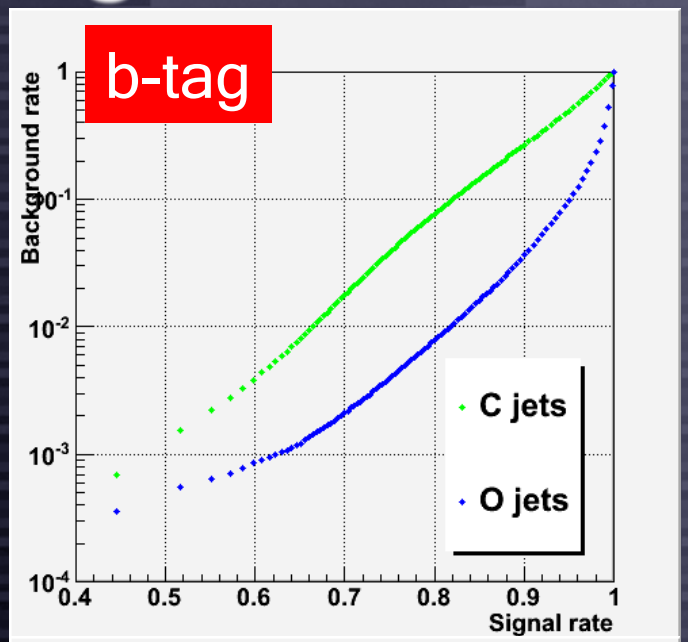
- Vertex finder
- Jet clustering
- Flavor tagging
  - Track ntuple (skip today)
  - Make ntuple
  - Training
  - Flavor tagging

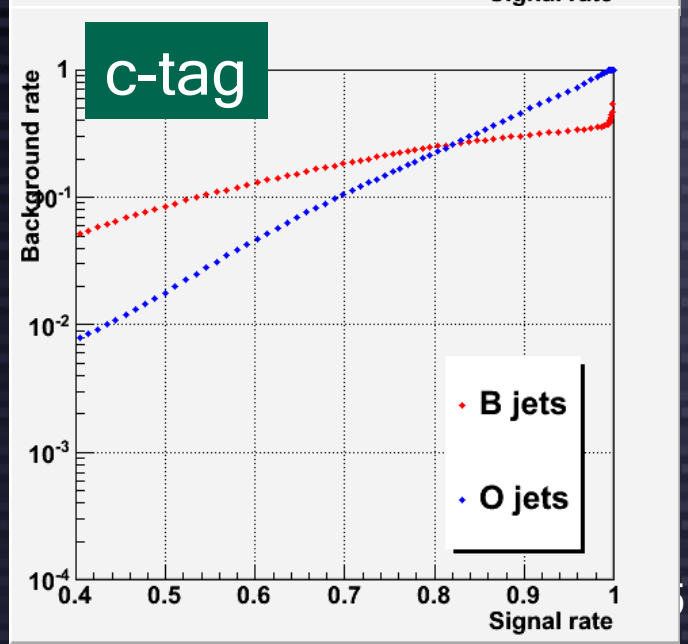
Flavor tagging

- Boosted decision trees
  - Number of vertex
    - Used for categorization
  - vertex mass
  - impact parameter
  - Related variables (~20)



with pT correction from neutral particles





Algorithms (3)

- Flavor tagging
  - Training algorithms
    - TrackNtuple and TrackProb.C
      - To prepare data files to be used in flavor tagging
    - MakeNtuple
      - To prepare training data
      - Output of TrackProb.C needed
    - Train
      - Run TMVA to train the flavor tagging BDT with input from MakeNtuple output
  - FlavorTag/ReadMVA
    - Evaluate RefinedJets with trained BDT data

Re-training procedure

- Case for re-training
  - New geometry
  - New algorithm (eg. New input variables)
  - Optimization for specific # jets and jet energy
    - Eg. 6q500, 4q250, 6q1000, etc.
  - Comparing performance with parameters

Re-training procedure

- TrackNtuple (see trackntuple.xml)
  - Necessary with different geometry (esp. different vertex detector configuration)
  - Currently doing with Z-pole bb/cc/qq
    - Energy dependence should be small, to be checked
  - Prepare 10k event each
  - Run TrackNtuple after JetClustering
- TrackProb.C (ROOT macro)
  - Prepare histograms based on the result of TrackNtuple outputs
  - Available in LCFIPlus/macro/TrackProb.C

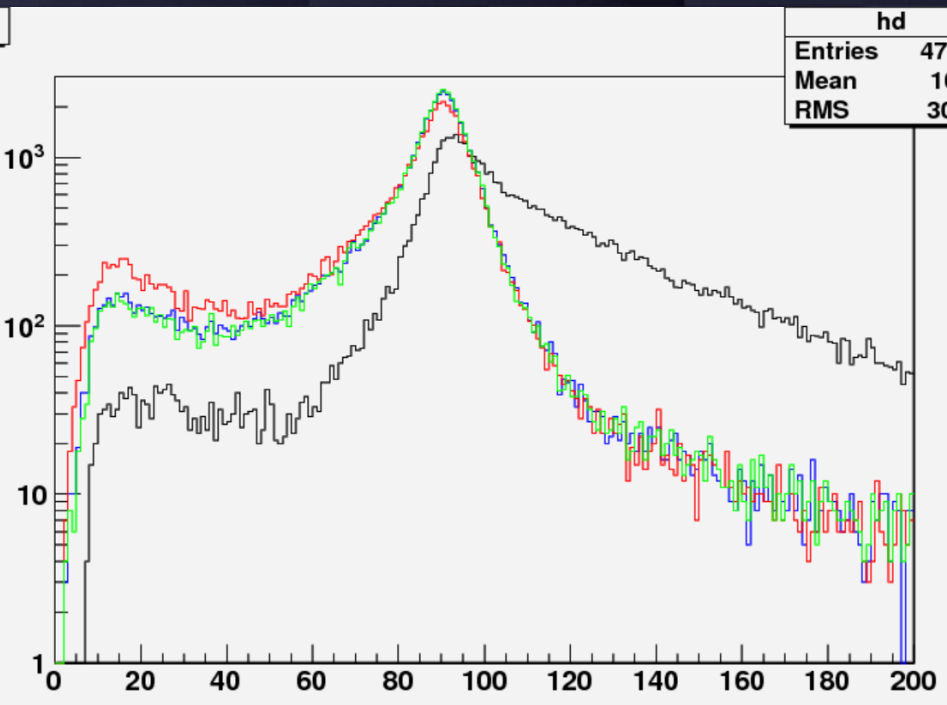
Re-training procedure

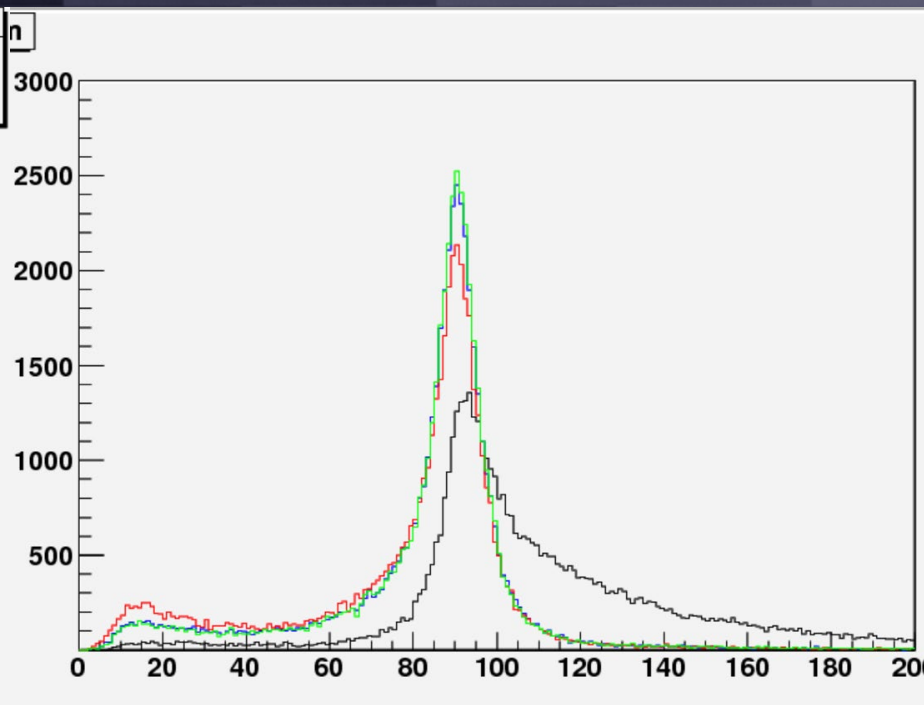
- MakeNtuple (makentuple.xml)
  - To produce training sample
  - Output of RefinedVertex needed
  - 10k of b/c/q sample minimum, 100k recommended
  - Can be 2q/4q/6q
    - For 4q/6q, bbbb(bb)/cccc(cc)/qqqq(qq) sample should be prepared
    - Categorization based on MC information has been prepared, but not well tested
- Output should just be “hadd”ed for training

Re-training procedure

- TrainMVA (train.xml)
  - Training BDT by TMVA
  - Confirm the “correct” ROOT version
    - Explained later
  - Input: bb/cc/qq ntuples (specified in steering)
  - Cannot be split over CPUs, time consuming if you use 100k samples
  - Output is a directory. ROOT files inside the directory is not needed (just for performance evaluation)
  - TMVA evaluation script can be used
    (but be careful that we use “multiclass” training)

Jet Clustering: performance





Invariant mass of Z
($Z\nu\nu \rightarrow qq\nu\nu$, 500 GeV, 2-jet clustering)

Kt: R=1.4
Durham: α=1.3
Valencia: R=1.1, β=1.0