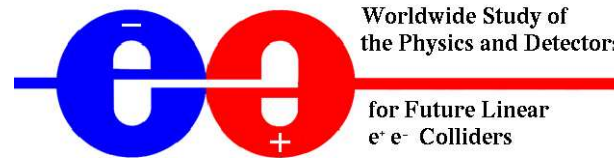


Skeleton of Particle–Flow Reconstruction Program

Vasiliy Morgunov

ITEP, Moscow



DESY, 9–10 December 2004

The copy of this talk one can find at the <http://www.desy.de/~morgunov>

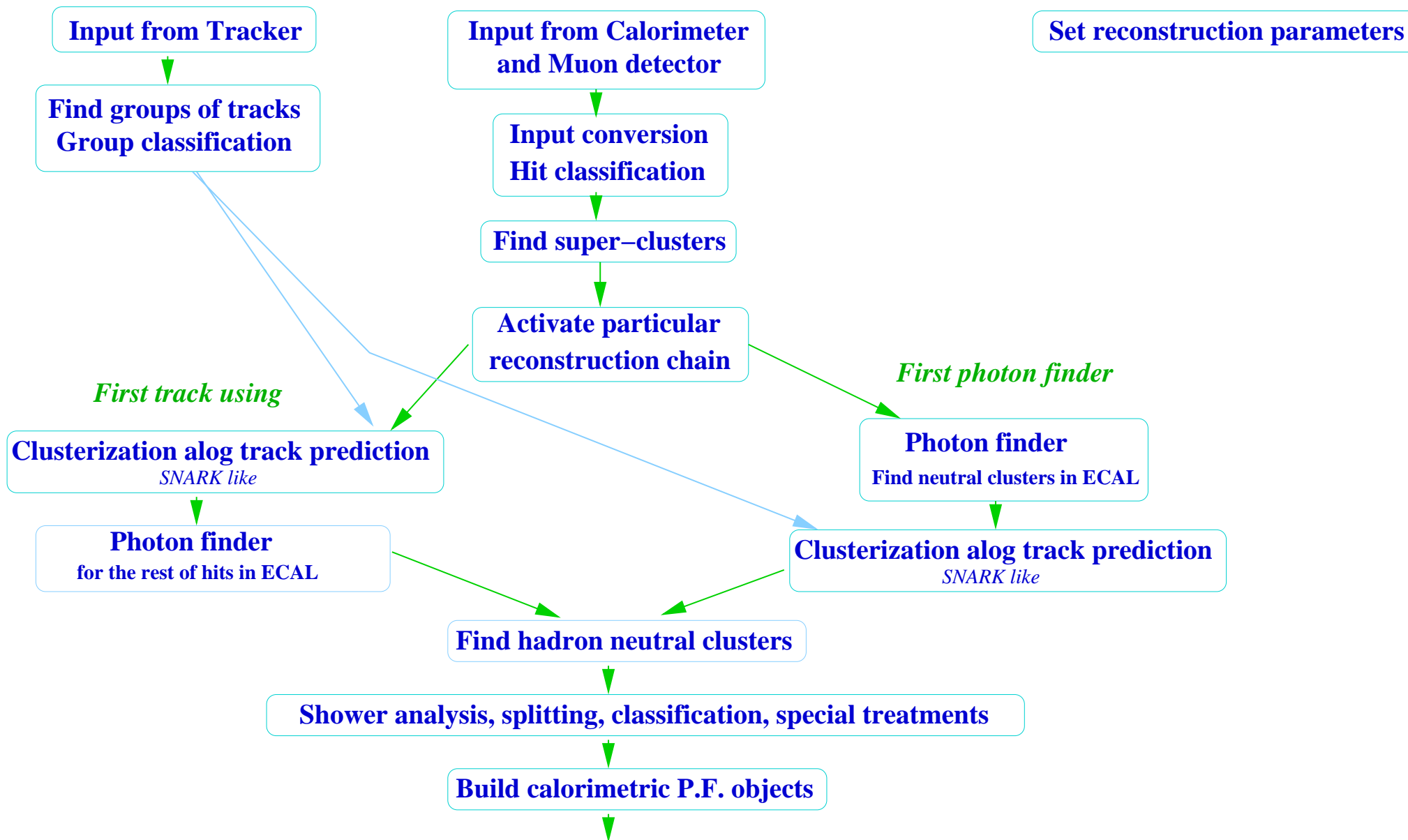
Plan

1. Aims of the program
2. Global view
3. Some details
4. Requirements

Particle–Flow Reconstruction Aim

Create the best estimation of parameters for each particle in event
or for all measured particles at the interaction point of the detector
using whole information read by detector.

Functional Skeleton



Functional Skeleton, continue

Track analysis

Find kinks, V0 and albedo tracks
Find gamma conversion points
Find bremsstrahlung points and photons

Build tracker P.F. objects

Calorimetric P.F. objects

Collect all of them into P.F. objects

Calculate final hypothesis, parameters and errors

Discard background and unused hits and tracks

Vertex analysis with final P.F. objects

Input conversion and Hit classification

Here is the latest connection between detector geometry and reconstruction program.

Latter on reconstruction procedure should no depends on any kind of detector.

All hits should be converted/transferred into unique physical coordinate frame and into one physical energy scale here.

Calorimeter hit classification includes:

- Energy cut-offs
- Hit density calculation depending on the absorber type
- Splitting hits into energy density classes including digital or semi-digital cases, in compare with MIP density
- Building of hit status words
- Some geometrical information can be included here into the hit data for further using

Find super-clusters

This procedure is more or less technical.

It will help to make less computer consuming time.

All hits should be separated in 3-D space into few super-clusters by the simplest cone algorithm, or histogramming technique in theta-phi spherical coordinates.

Clusterization along track prediction

- Build a prediction for track core in ECAL and HCAL using group of track information
- Build a predicted shower shape with spatial density distribution
- Collect hits around track core using predicted hit density distribution as criteria
- 3-D clusterization of found hadronic shower
 - Find/separate hadronic interaction points
 - Sub-cluster classification
- Calculate shower energy using energy correction, track momentum and shower properties
- Preliminary P.F. building for each shower

Photon finder

Can be based either on the 3-D clusterization either on H1 clustering algorithm or any others.

It should use HCAL hits at the few first layers if cluster energy more than 10 GeV.

- Find e-m clusters in super-cluster – by any technique
- Cluster analysis using e-m shower model (energy in the first order is known for each found cluster)
- Gamma shower splitting using $\pi^0 \rightarrow \gamma\gamma$ angular cone prediction; force for clusters with energy more than 10 GeV
 - Calculate transversal eccentricity, if it is less than some threshold – suppose it is π^0 case
 - Build a hypothesis for π^0 and calculate angle between gammas
 - Build two centers of clusters and recollect hits
 - Hits at the overlapped region separated to the clusters proportionally to the none-overlapped regions energies
 - Analyze of two showers separately and backward check hypothesis
- Find nearest and look for π^0 mass
- Find resulting clusters
- Calculate e-m shower parameters and errors
- Preliminary build P.F. objects

Find neutral hadron clusters

- Find clusters inside super-cluster by any simple procedure
- 3-D clusterization of found clusters
 - Find/separate hadronic interaction points
 - Sub-cluster classification
- Calculate shower properties
- Preliminary P.F. building for each shower

Shower analysis, splitting, classification, special treatment

Secondary shower analysis is to find shower overlaps and split shower if it is found.

Shower classification – an attempt to estimate a hadron type that creates a shower. It also includes muon case.

Special treatment

- μ -tracks can be treated specially to estimate μ -hypothesis probability.
- for low energy hadrons that did not go through the calorimeter with helix curve, it should be proceed with more complicated track prediction using energy losses and multiple scattering estimation.
- for albedo tracks.
- for tracks and showers near by the accelerator tube; take into account a possible leakage.
- for neutron hits; attempt to collect it into the initial shower.

3-D clusterization

Input: list of hits with $x, y, z, r, \theta, \phi, E, id$ and *keywords*

Step1: r-numeration, i.e. calculate spherical shell number for each hit (here is significant the thickness of the shell).

Step2: Collect hits at one spherical shell.

Step3: 2-D clusterization at this shell by θ - ϕ histogramming technique.

Step4: Calculate distance for joining – most significant step of algorithm; none-Euclidian metrics can be used here; and adaptive self tuning for particular object.

Step5: Start a tree collection. (tree is 3-D object with distances between neighbors less than chosen distance).

Step7: Start new tree when chain is broken.

Step6: repeat step 5 to the end of hit list.

Output: List of list of hits.

About OO Classes

LCIO classes are the base for reconstruction classes.

Class LCIO-hit should be extended to keep some additional information about hit which is once calculated to keep small computer consuming time.

Class Shower can be easy inherit from LCIO cluster class; it will carry additional data and links.

New classes for reconstruction especially can be called **Builder...** and **Analyzer...** of group, cluster, shower, P.F. object... .

Or **Processor...** as it is proposed for **MARLIN** framework.

Builders are actually make lists or lists of lists of hits and nothing more.

Analyzers are actually calculate the properties of that lists, it will keep that properties inside particular object, once calculated and then it can be extracted by the simplest function *Get.property(...)*.

These classes may keep all active part of program including all algorithms and calculation procedures in its member functions to separate actual calculations from data and to keep a flexibility for the different reconstruction algorithms that can be easily changed at any time.

Time requirement, man power

1. Detailed description of program structure	0.5 man*year
2. Writing the main Processors	1.0 man*year
3. Development of visual debugging and tuning tools	0.5 man*year
4. Writing utilities	0.5 man*year
5. Tuning and Debugging	0.5 man*year
6. Documentation	0.5 man*year