1 Detector Simulation and Detector Performance

This section covers most of the issues related to the software simulations performed by the 4th Concept. We will describe our software framework, ILCRoot, pinpointing some of its most imortant featires. Then, a detailed description detectors is done, along with the algorithms we used to simul;ate the vaious steps from the hits produtions in the sensitive materials trhough the digitization, the pattern recognition and the reconstruction. Finally, we will present the results of those studies we have done that are related to the performance of the detectors either with single particles or in topologically simple cases like di-jets events.

1.1 Software strategy

History is teaching that a non-optimal choise of the software systems at adopted in the initial phase of a new project has, very often, carried some serious consequences later in the life-span of the experiment. Infact, in general, when the details of the apparatous increases and the complexity of the algorithm used in the simulation-reconstruction, most of the flows of the software architecture are exposed nad the patches usually applied in the course of the simulations do not hold any more. Furthermore, in the majority of the cases, the software apparatus used in the design phase of the project is also adopted in the offline systems of the experiment. In this case, a famework not designed from the beginning for migrating into an offline systems, also presents major problems at some point during the life of the experiment. In both of the above cases, the outcome is a major overhoul of the software framework, with non negligeable overhed involved. Recent examples of software systems that had such fate involve some of the LHC experiments.

In the light of the above considerations and of the experience gained in the past, we have set a number of guidelines for our software and offline systems:

1. Use of public domain common tools. This way we can rely on the huge amount of tools the HEP community has developed in the past.

2. Impose a single framework. The simulations, reconstruction, Offline systems and analysis will all be made within the same framework.

3. Adopt ROOT as software infrastructure. ROOT is probabEP community. All needed functionalities are present (from data taking to final plots). Reconstruction and analysis are naturally developing in the same framework. Furthermore, it has extensive support from the largest laboratories as CER, FNAL

and KEK. It has an unprecedented large contributing HEP Community. It is an Open Source, mutplatforms project, supporting multi-threading and asynchronous I/O. Optimised for different access granularity (Raw data, DST's, NTuple analysis)

We also impose a number of requirements on the software framework, to be built over the ROOT infrastructure. Such requirements are needed to garantee the easy of use by several groups, possibly located in distant places and also to garantee that the system can evolve smoothly into a full Offline systems to be used when the experiment is running. The main requirements on the software framework are the following:

1. Scalability. This is essential to guarantee that, increasing the number of processing nodes, we also increasi the troughput from the offline systems.

2. Portability. When this feature is satisfied, the same code is able to run on different computing systems.

3. Multiplataform. This requirement is essential when computing farms are built using different computers or operating systems.

4. High level of modularity. This is a guarantees for easy of maintenance and development. We want to pursue modularity of the framework by imposing the absence of code dependencies between different detector modules and by designing the structure of every detector package in a way that static parameters (i.e. geometry and detector response parameters) are stored in distinct objects.

5. The data structure has to be built up as ROOT TTree-objects. The access to the data will occure efficiently either to the full set of correlated data (i.e., the event) or to only one or more sub-sample (one or more sub-detectors).

All the above requirements have conducted us to the implementation of ILCroot, the software and offline framework of the 4th Concept. An analogous choice has been made besides Alece and the 4th Concept, by other collaborations in High Energy Physics, most notably CMB, Panda, Opera and (initially) MEG. The growing number of collaborations adopting this framewor guarantess an expanding number of detector modules being developed and available to other for reuse. A detailed description of the 4th Concept Offline systems can be found in reference [**?**] of the bibliography.

1.2 The ILCroot Framework

1.2.1 Overview of ILCroot framework.

ILCroot derives from AliRoot, Alice's Offline framework[**?**]. The entire architecture and the structure of the classes have been inherited by the original with few, minor changes, needed for the simulations of an experiment to an electron-positron collider. The detector modules, containing the decription and the functionnalities of the 4yh Concept sub-detectors, have been iplemented according to the designs described in the previous sections of this document. It is evident that this choise for the software framework has granted to us a tremendous compatibility with the very extensive code developed by the Alice Offline Group and by other experimental groups who have also adopted an Aliroot-derived framework. The modularity of such framework easily allows replacing existing detector modules with new ones specific to the detector that is being simulated. Furthermore, the robustness of the persistent data loader makes it a perfect tool for handling large amount of data at very high dataflow (1.8 GBytes/sec). The architecture of ILCroot is briefly descibed here.

A STEER module provides steering, run management, interface classes, and base classes to the entire framework. The detectors are independent modules that contain the code for simulation and reconstruction while the analysis code is progressively added. Detector response simulation is performed via the Virtual Montecarlo (VMC) technology[**?**]. The user code is all in C++, including the geometry definition. All the user−defined parameters (including the event generator and the kinematics of the event) are included in a plain C macro−file ("*Config.C*") which is interpreted at execution time by the ROOT interpreter (*CINT*).

1.2.2 The Detector Module

The Detector Class is ILCroot building block; each detector is responsible for its code and data. Cross-modules calls are not allowed. Both sensitive modules (detectors) and non-sensitive ones are described by this base class. The Detector Class must support:

- Geometry description of the active and non-active volumes. The geometry can be specified using several ways. At present, the following descriptions are supported: Root (TGeo), Geant3, Geant4, Fluka, GDML, Oracle and CAD (altough the latter only in semi-automatic manner).
- Event display
- Simulation by the MC by providing the appropriate behaviour of the StepManager

- Digitization
- Pattern recognition
- Local reconstruction
- Local PiD
- Calibration
- QA
- Data from the above tasks

The structure of every detector package is designed so that static parameters (like geometry and detector response parameters) are stored in distinct objects. This detector-centric approach (compared to the more traditional processor-centric approach used in the majority of the existing software frameworks) as several advantages. The two most notable are that it is easier to work for groups across many countries, as in the case of collaborations like at ILC that are spreaded over several continent, and that it allows for several versions of the same detector or of several detector of the same kind (ex. TPC and DCH) to co-exist at the same time. The user will select the desired one via a switch in the config file.

1.2.3 The Virtual Montecarlo

The Virtual MC provides a virtual interface to the Monte Carlo. It enables the user to build a Monte Carlo application independent of any actual underlying Monte Carlo implementation. The greatest advantage, from the developer point of view, is that it decouples the dependence of the user code on a concrete MC as it allows to run the same user application with all supported Monte Carlo packages. Detector response simulation can be performed via different transport codes like GEANT3[2], GEANT4[3], and FLUKA[**?**]. The concrete Monte Carlo is selected and loaded at run time. This kind of approach makes the VMC an ideal tool when switching from a fast to a full simulation or when it is necessary to compare the predictions of different Montecarlo's systems (for example Geant3 vs Geant4), with the same detector geometry and data structure. In other words, the VMC allows to run different Monte Carlo simulations from the same user code. Two other advantages of the VMC are the following:

- It helps in smoothing the transition from Geant3 to Geant4 with maximum reuse of Geant3 based simulation code (user code)
- It makes possible the integration of tracking codes developed in the future.

With the adoption of a software framework based on the Virtual Montecarlo the user can decide which concrete Montecarlo to load and thanks to the the abstract interface, the transition from GEANT3 to GEANT4 and to FLUKA is seamless, since no user code has to be changed; only a different shared library has to be loaded. Another advantage of this approach is that one can generate and simulate different events with different Montecarlo's and merge the corresponding digits in digitization phase. A tipical example would be to use Fluka for signal events (in case one is partucularly sensitive to the simulation of hadronic showers) and Geant4 for beam background (as this Montecalo handle more realisically many soft tracks produced by the photons in the tacking devices). Another important application of the VMC approach is from the Quality Assurance point of view as it allows the comparison between Geant3 Geant4 and Fluka using the same geometry and data structure.

From what concerns the Offline architecture, the VMC is a separate package running on top of ROOT along with the real Montecarlo's packages and any other external package. It interacts with ILCroot main reconstruction program through the Run Manager which sends calls to the VMC whenever needed. From the programmer point of view, the development of the VMC package is basically independent from the development of the main program as it shares with it only the output format and the Geometry Database.

1.2.4 Event generators

The interface with the event generator is implemented through a special class called TGenerator. TGenerator is an abstract base class that defines the interface of ROOT and the various event generators (thanks to inheritance). It allows easy mixing of signal and background and, furthermore, it provides the user with an easy and coherent way to study a variety of Physics signals. From the data analysis point of view it makes it easier to test new generators and to perform background studies. It is, also, possible to study:

- Full events (event by event)
- Single processes
- Mixture of both ("Cocktail events")
- Cocktail of different processes

The TGenerator virtual class allso allows rate and weighting control of different event types and easy mixing of signal and background events. At present, more than than a dozen generators are built-in in ILCroot. An appropriate module to read events from

an external file in various format (txt, stdhep, etc.) has been added to interface event generators not natively supported by ILCroot. As a last remark, another advantage of using the TGenerator class is that we can clone an existing event generator and changing it to fit our specific needs, modifying, for example, some of the parameters related to the production of the jets in a e^+e^- collisions.

1.2.5 Processing flow in ILCroot

The way ILCroot implements a modular system is by the use of ROOT folders. They can contain basically any kind of object, including other folders, collections of objects or multiple collections of objects. A folder and its content can be accessed either by program or trough a ROOT browser. Folders are organized into two main categories:

- Data folders
- Task folders

The objects contained in the data folder represent basically constants and events while the objects in the task folders correspond to actions to be performed by the Detector modules (see below). Tasks can be organized into a hierarchical tree of tasks and displayed in the browser. A Task is an abstraction with standard functions to Begin,Execute,Finish. Each Task derived class may contain other Tasks that can be executed recursively. However data and task folders interoperate:

- Data Folders are filled by Tasks (producers)
- Data Folders are used by Tasks (consumers)

Posting the objects to a white board performs transient data exchange at run time: each sub-detector is responsible for posting its own data. The tasks access the transient data trough the whiteboard. Whenever several detectors need to cooperate, they do it through the whiteboard.

Task folders There will be six different task folders per each subdetector:

- Reconstruction: it performs the reconstruction of the subevent in the specific subdetector
- Data Quality Monitor: it checks the quality of the subevent by a fast reconstruction and it fills histograms that are part of the event and can be accessed at any stage of the dataflow for quality assurance
- Calibration

- Alignement: computes the alignment constant for that particular subdetector
- Vertex finding
- Trigger

Data folders The data folders appear within the ROOT object browser like any file browser. Objects can be accessed online and the histograms can be visualized through the ROOT internal 2-D and 3-D graphic package. Schematics of the data folder is shown in figure 3. The structure of the subfolders is outlined below. There will be:

- One main folder with the constants data
- One folder per run with the header information
- One folder per run with the condition information
- One folder per run with the configuration constants
- Within each run folder, there is one event folder per each event collected. It shall contain:
- One subfolder with the raw data
- One subfolder with the reconstructed data
- In the case of a Montecarlo event, one extra subfolder with the kinematics and another with the track reference.

Each subdetector will have its own subfolder in either the raw and the reconstructed folder. The way the data objects are stored on disk, however, is totally different from the way they look in the object browser. ROOT takes care of the interface between the disks and the user. There will be, for the raw data:

- One common TFile with all the constants
- One TFile per subdetector with the raw data (the hits) organized in several TTree's

During the reconstruction phase, each task produces one digits TFile and one reco TFile per subdetector. The events are organized in TTree's as well. An example of data structures used to store events is shown in fig. **??**.

Tasks and data coordination The coordination of the various tasks and data is performed by the Run Manager. It executes the tasks objects in the order they are listed in the configuration script. Global trigger, simulation and reconstruction are special services also controlled by the Run Manager class. The base class for each subdetector module is the Detector Class . Both sensitive modules (detectors) and non-sensitive ones are described by this base class. The Detector class supports the hit and digit trees produced by the simulation and the the objects produced by the reconstruction. This class is also responsible for building the geometry of the detectors.

The tasks objects belonging to the Detector Class are classified into two categories, depending if they need data object provided by other tasks:

- Detector stand alone (Detector Objects)
    1. Each detector executes a list of detector actions/tasks
    2. On demand actions are possible but not the default
    3. Detector level trigger, simulation and reconstruction are implemented as clients of the detector classes
- Detectors collaborate (Global Objects)
    1. One or more Global objects execute a list of actions involving objects from several detectors

The Run Manager executes the detector objects in the order they are listed. Each detector executes a list of Detector stand-alone tasks in such a way they would provide the data for the detector collaborate tasks. Finally, One or more Global Objects execute a list of tasks and data involving objects from several detectors.

Data Access The data will be stored within ROOT files and in a RDBMS database. The ROOT files constitute the events store. They will contain: " The TTree's with the event objects " The event related histograms " The geometry related informations.

The RDBMS will contain: " The Calibration Constants " The Run Catalog " The File Catalog

Basically, all the HEP experiment are now adopting the hybrid solution proposed here over monolithic solutions where everything is stored in one data system (i.e. Babar). ROOT offers integrated interface toward several RDBMS, commercial and open-source. The two RDBMS most used in the HEP community along with ROOT are Oracle and MySql. The former offers the advantage of a commercial support, while the latter is free and open source. As the ROOT interface is transparent to

the underlying RDBMS, the final choice can be made almost anytime during the development of the Offline.

A schematic view of ILCroot architecture is shown in fig. **??**

1.3 The Simulation Process

The simulation and reconstruction processes occur through the following passes:

- **Event generation.** The collision is simulated by a physics generator code or a parametrization and the final-state particles are fed to the transport program.
- **Particle tracking**. The particles emerging from the interaction of the beam particles are transported in the material of the detector, simulating their interaction with it, and the energy deposition that generates the detector response (hits).
- **Signal generation and detector response.** During this phase the detector response is generated from the energy deposition of the particles traversing it.
- **Digitization.** The detector response is digitized and formatted according to the output of the front-end electronics and the data acquisition system. The digitization process is split into two parts: the production of summable digits (*SDigitization*) and the final *Digitization*. In the first step, the response of the electronics to the passage of one individual particle is simulated in the assumption that no pile-up occurs. In the second step, the *SDigits* from several tracks corresponding to the same read−out channel are summed togheter.
- **Clusterization.** This pass is also called *pattern recognition*. In this phase the *Digits* are searched trough and separated in list associated with potential reconstructible particles. If the shape of one *Cluster* is such that it contains the contribution of two particles, an unfolding algorithm is called were we attempt to recover the signal belonging to each subcluster.
- **Reconstruction.** In this pass, the *Clusters* found are analized and the kinematic informations of the track are evaluated.
- **Particle Identification.** Finally, the informations from all the subdetectors are combined to extract a probability for tath particle to belong to a particular species.

A schematic view of ILCroot processing flow is shown in fig. **??**

The Data Model This section describes the data model proposed for the 4th Concept and includes the flow of data processing from the raw data delivered by the data acquisition system up to the final physics analysis. The software implementation of the data processing is done within the ROOT framework and will be described in the next section. Prior to data reduction, every data channel is calibrated on a detector-by-detector basis and the resulting calibration parameters are stored into a database common to all sub-detectors. The following step is the reconstruction. As suggested by the MONARC model, the reconstruction procedure will produce four objects that differ by their content and size (ESD, AOD, TAG, and DPD).

ESD Raw data, collected by the set of individual sub-detectors of the 4th Concept, are calibrated and objectified into Digits which will contain, at least in principle, the same information carried by the real raw data. Digits are then processed in a few consecutive steps; some specific to each sub-detector (clusterization) and others that require different detectors to collaborate. Each sub-detector produces a list of objects. Digits are then processed in a few consecutive steps; some specific to each sub-detector (clusterization) and others that requires different detectors to collaborate. For example, the Vertex Detector and Drift Chamber collaborate to reconstruct track segments, the DREAM and the Muon System can collaborate to identify charged particles. Each step produces its own kind of reconstructed object (RecPoints from the clusterization and TrackSegments from the tracking). Some sub-detectors provide already at this stage a best guess of the particle identification, stored in RecParticle objects (for example the EMCAL can already identify photons at this stage). Each sub-detector produces a list of each one of these objects. The collection of these reconstructed objects constitutes the Event Summary Data object, whose estimated size is on average 50 kByte (see later).

AOD Starting from the ESD object, the reconstruction process combines the information (segments and particles) reconstructed individually by the sub-detectors to create the physics object that is a list of reconstructed particles. This task requires the cooperation of several sub-detectors. The Analysis Object Data object contains all the information to perform the physics analysis (in most cases), typically the list of particles and vertices. Its size is estimated not to exceed 10-20 kBytes.

Tag The Tag objects collect the information that characterizes an event. The parameters will be computed from the ESD and AOD objects or directly collected from the L3 trigger information. Global information includes, among other things, primary vertex position, total charged multiplicity, total electromagnetic energy, and so on. The information indicating the presence of rare signals

should be included as well. The size of the Tag object is estimated to be of the order of few kBytes.

DPD The Derived Physics Data objects are constructed from the physics analysis of AOD and Tag objects. They will be specific to the selected type of Physics analysis and will typically consist of histograms or ntuple-like objects. These objects will in general be stored locally on the workstation performing the analysis, thus it does not add any constraint to the overall data-storage resources.

The output of the reconstruction is the Event Summary Data (ESD) containing the reconstructed charged particle tracks (together with the particle identification information), decays with the V0 (like $K_{short} \rightarrow p\pi$ and $\lambda \rightarrow \pi^+\pi^-$), kink (like charged $K \rightarrow \mu\nu$) and cascade topologies and some neutral particles reconstructed in the calorimeters.

Calibration Database The calibration algorithms will feed the calibration database that can be subsequently accessed through a ROOT interface. The calibration parameters will be stored in database tables as plain variables, deferring to the reconstruction program, which will collect the information, the task to structure them into objects.

1.4 Detector Simulations

General remarks The detector simulations presented in this document are the results of a full simulation performed in ILCroot. The geometry description of each sub-detectror is quite detailed and the detector response to the effect of the passage of the particles through their volumes is also fully simulated. Except for the case of the Muon Spectrometer, the production of hits occurs trough appropriate algorhitms and the reconstructed point along with the measurement error is estimated from a clusterization process. No smearing of the energy deposit nor of the reconstructed cluster is used (with the exception of the Muon Spectrometer). A more accurate description of the algorhitms mentioned above is done in the following sections. The Montecarlo used in the simulations depends on the specific study or reaction. In general, when jets or calorimetric measurements are relevant to the study, Fluka is the Montecarlo of choise, becouse of its superior treatement of hadronic interactions and shoer developments. On the other side, when tracking measurements are relevant or low momentum tracks are involved, we use Geant4 because it is more reliable in such envisronment. In some cases we repeated the study with Geant3 for cross chek and Quality Control. The eletronic noise is implemented in the simulation of each sub-detector with a gaussian shaped distribution in the digitization step. Thresholds of the readout electronics have been included as well.

Simulation of the Vertex Detector

Geometry Description As mentioned in the previous sections of this document, the layout of the Vertex Detector has been derived by the SiD design. The tanserse dimensions are scaled to the into account the different magnetic fields proposed by the two collaborations. Consequetly, the geometry description file, sidmay06, has been kindly provided by the SiD collaboration. The dimensions have been rescaled according to the observation above. The layout in the simulation is realistically described trhough the use of ladders. Each ladder consists of a silicon sensor with thickness of 100 μm and a readout chip on top of it. The latter is simulated with a thin layer of carbon fiber, of thickness about 100 μm. No description of supporting structures nor the utilities (cables, cooling, etc.) is teken into account for the studies presented in this document.

Detector response and digitization The simulation response of the Vertex Detector to the energy deposited in the silicon by any particle occurs trough several steps. First, the algorithms looks at the energy deposited along the path of the particle in the Silicon in steps of 1 μm. The corresponding energy dE is converted into a charge dC with the following formula:

$$dC = \frac{dE \times dt}{3.6e^{-9}}$$

The charge is spreaded in 2-dimensions across several pixels according to the following formula:

$$dQ = dC \times Erfc(x, z, \sigma_x, \sigma_z)$$

where:

$$\sigma_x = \sqrt{2k/e \times T^o \times dV \times L} \;\; ; \;\; \sigma_z = fda \times \sigma_x$$

where

$$T^o = 300^oK \;\; fda = 85\% \;\; dV = si\ thickness/bias\ \ voltage$$

σ_z is receive a scaling factor to take into account the effect of the magnetic field in the diffusion of the charge toward the pixels. One advantage of using the algorithm adopted is that charge pile-up is automatically taken into account in case of two tracks hitting the sensor with a distance comparable to σ_z. An extra single-hit inefficiency of 0.5% is simulated. Events merging (like for example signal with beam background) occurs at this stage. Finally, a charge couplig effect between nearby pixels is added row-wise and column-swise with constant probability. Electronic noise and threshold is also taken into account.

Pixel clusterization By pixel clusterization we mean the process of finding clusters of pixels, potentially produced by a single particle and extractig the informations relevant for track reconstruction (position, error, energy lost, etc.). We create an initial cluster from adiacent pixels (sidewise only, no diagonal). The, an iterative process starts where the previous cluster is subdivided in smaller NxN clusters (in the current implementation we use a 25x25 cluster range). All the sub-clusters are stored in the event and the Kalman filter in the global reconstruction will pick up the cluster that better fits the track being reconstructed.

Drift Chamber

Geometry Description The geometry used for these studies reflect the latest version of the Drift Chamber with spherical end plates, introduced at LCWS08. The description of the volumes and materials is quite accurate, including each single wire (both sense and field) to take into account the effect of kinks on tracks hitting the wires. The electronis is also simulated by adding an external layer of carbon fiber and copper to eah end plate. However, no supporting structure are simulated for the present studies.

Detector response and digitization The implementation of the detector response to the passage of a track uses a fast algorithm to simulate the Cluster Couning readout. This has been chosen in order to speed up the calculations, otherwise overwhelmingly slow given the size od the Drift Chamber. However, all the features of the Cluster Counting rechnique have been implemented, the most notable being the loss of resolution and efficiency when more than one track is present in the same cell. In this case, we associate single hit resolution of 55 μm to the first arriving track and of 120 μm to the second track. Any other track is assumed lost. The correct implementation of such behaviour is very important when events with many tracks are considered (like events with 6 jets) or when the beam pair background is taken into account.

The Electromagnetic Calorimeter

Geometry Description The crystal forming the electromagnetic calorimeter are simulated as individual volumes. In front of each crystal we assumed a layer of carbon fiber 3mm thick and a layer of silicon also 3mm thick. They are intended to simulate front end electronics (boards, SiPM, FADC, etc.). The support structure is on the back of the calorimeter and correspond to 5mm of carbon fiber.

Detector Simulation

The Hadronic Calorimeter

Geometry Description The geometry desciption of the hadronic calorimeter is estremely detailed. Each single fiber is individually simulate as an independent volume in order to improve the simulation of the detetor response to showers. However, for the studies presented in this document we have to eliminate the cladding of the fiber in order to speead up the computing time. We observed that the reulults were totally unaffected by this approximation. The structural support of the calorimeters are not simulated nor the electrinics and services on the bac of the towers. We believe that this approximation also minimally affect the results as the missing volumes occupy areas of the detector boyond the sensitive modules.

Detector response and digitization

Shower reconstruction The energy reconstruction of the calorimeter needs to go through a two-step-calibration. The first step is required to calibrate the raw signal from scintillating and Čerenkov fibers. The second step is required to keep the hadronic shower energy independent from the electromagnetic energy fraction, f_{em}, and the neutron component, f_N, whose fluctuations degrade the performance of the hadronic calorimeters (as explained in Section **??**).

In the first step, single 45 GeV electrons are used in order to evaluate the raw signals (in terms of *pe*) of the scintillating and the Čerenkov fibers. Then the amount of energy associated with each *pe* is evaluated in accordance with the request that the total reconstructed energy form each event must be 45 GeV for both signals.

In the second calibration step, to take into account the electromagnetic energy fraction, f_{em}, and the neutron energy fraction, f_N, single 45 GeV π^- are used. Starting with the equation (see [**?**]):

$$R(f_{em}) = f_{em} + \frac{1}{\eta}(1 - f_{em}) \tag{1}$$

for scintillation and Čerenkov energy one gets:

$$\frac{E_C}{E} = f_{em} + \frac{1}{\eta_C}(1 - f_{em}) \tag{2}$$

$$\frac{E_S}{E} = f_{em} + \frac{1}{\eta_S}(1 - f_{em}) \tag{3}$$

where E_S and E_C are the raw scintillation energy without the neutron contribution and Čerenkov energy respectively, taken from the first step of the calibration and evaluated event-by-event. Evaluating f_{em} from the equations (2) and (3) and equating the results the following equation is obtained:

$$E = \frac{\eta_S \cdot E_S \cdot (\eta_C - 1) - \eta_C \cdot E_C \cdot (\eta_S - 1)}{\eta_C - \eta_S} \tag{4}$$

where E is the π^- energy (in this case 45 GeV). This equation needs to be corrected for the neutron contribution. This contribution is characterized by an exponential tail in the pulse shape, due to the contribution of the low energy neutrons, as shown in Figure **??**. The energy component of the neutrons (E_N) is evaluated by a fit to the tail of the pulse shape extrapolated to the peak.
The equation 4 then becomes:

$$E = \frac{\eta_S \cdot E_S \cdot (\eta_C - 1) - \eta_C \cdot E_C \cdot (\eta_S - 1)}{\eta_C - \eta_S} + \eta_N \cdot E_N \tag{5}$$

Minimizing the $RMS(\eta_C, \eta_S, \eta_N)$ of the distribution 5, η_C, η_S and η_N are evaluated.

Once the HCAL is calibrated the equation:

$$E_{HCAL} = \frac{\eta_S \cdot E_S \cdot (\eta_C - 1) - \eta_C \cdot E_C \cdot (\eta_S - 1)}{\eta_C - \eta_S} + \eta_N \cdot E_N \tag{6}$$

is used to estimate the calorimeter energy. Figure **??** shows the reconstructed energy distribution of 45 GeV π^- (d) after correction for the electromagnetic fraction, f_{em}, and neutron fraction, f_n, obtained through the measure of the scintillation signal without the neutron contribution (a), the Čerenkov signal (b) and the neutron component in the scintillation signal (c). We can observe that, the reconstructed energy distribution is Gaussian and centered at 45 GeV, while the three raw signals are not.

The Muon Spectrometer

Geometry Description As for the other detector, the simulations of the volumes involved is quite accurate, including the thin walls of the straw tubes, the sense wires and the electronics. The support structures, however, are have not been included in these studies.

Detector Simulation The simulation of the hits produced by the charged tracks in the Muon Spectrometer is handled by the VMC. The *SDigitization* and *Digitization* steps are not implemented yet. The hits from the VMC are smeared according to a

gaussian distribution with $\sigma_{r\phi} = 200\mu$m and $\sigma_z = 3$mm, with the intent of simulating the readout of a proportional tube with charge division in the longitudinal direction. Random noise is, finally, added to the whole subdetector to simulate spurious hits.

1.5 Reconstruction

The reconstruction of the particles in the subdetectors proceeds trough two phases: *Local* and *Global Reconstruction*. The *Local Reconstruction* is perormed by the specific detector module and it handle both the reconstruction algorithms (tasks) and the corresponding data. As stressed in 1.1, ILCroot modularity imposes that the local reconstruction in one detector module is totally unrelated to any other module, with no exchange of data of any sort. STherefore, several version of the of a certain detector can coexist with no interference. The desired version is called at runtime by setting the corresponding switch in the *Config.C* file.On the other side, the *Global Reconstruction* integrates toghether the data posted by the several detector modules partecipating (for example, the three tracking detectors: Vertex Detector, Drift Chamber and the Muon Spectrometer) and perform the final reconstruction.

1.5.1 Local Reconstruction

Each particle going through the detectors leaves a number of hits at the position in space where it has traversed the sensitive material. These simulated space−points signals needs to be associated to the same particle we are trying to reconstruct. This process is called *clusterization* and it is equivalent of the traditional pattern recognition. The *clusterization* is very detector specific; therefore it is handled by a dedicated class per each detector module simulated. The clusterization algorithms implemented in ILCRoot at his stage are quite simple: a cluster is defined as a collection of nearby *digits* (see section 1.1). For each space point we also calculate the uncertainty on the estimate of the position of the *cluster*. As mentioned above, only the Muon Spectrometer only undergo a gaussian smearing of the generated hits, while the other subdetectors also simulate the Physics involved in the production of the signal.

1.5.2 Global Reconstruction

The *cluster* points along with the position uncertainties are then passed to the particle reconstruction. This process will involve several subdetectors and, in this sense, is called *Global*. The *Global Reconstruction* is quite different depending if we are trying to reconstruct a charged track or a calorimetric object.

1.5.3 Global Reconstruction

The space points found in the Local Reconstruction along with the position uncertainties are passed to the track reconstruction.

The reconstruction algorithm of the charged tracks is performed by a Parallel Kalman Filter algorithm[**?**]. The implementation of such method has been imported by the Alice experiment (Aliroot) which has a very good success in reconstructing tracks in their large multiplicity environment (up to 40% occupancy and more than 20,000 tracks)[1].

The reconstruction algorithm is able to cope with non-Gaussian noise and ambiguous measurements. To gain almost optimal results, so called Maximum Information Approach (MIA) is applied. An incremental approach to combined reconstruction was chosen. Algorithms and data structures are optimized for fast access and usage of all relevant information. To allow the use of the optimal combination of local and global information on tracks and clusters, a parallel hypotheses Kalman Filter tracking method is used: several hypotheses are kept and investigated in parallel. The advantage of a parallel vs a conventional (i.e. a simple Kalman Fit) approach rely on the fact that the algorithm performs 4 tasks at the same time:

1. Patter Recognition. New clusters are added or old ones are discarded in the attempt of reconstructing each track
2. Kink finding. Discontinuities in the tracks are treated as kinks and reconstructed accordingly
3. V0 reconstruction. Tracks originating from a common detached vertex are added to a V0 bank
4. Track fitting.

The detector specific implementations of the track reconstruction algorithm use a set of common base classes, which makes it easy to pass tracks from one detector to another and test various parts of the reconstruction chain. For example, we can easily switch between different implementations of the clustering algorithm or the track seeding procedure. This also allows us to use smeared positions of the simulated hits instead of the ones reconstructed from the simulated detector response, which is very useful for testing purposes.

The event reconstruction starts with the determination of the position of the primary vertex. This can be done prior to track finding by a simple correlation of the space points reconstructed at the first two layers of the Vertex Detector. The information about the primary vertex position and its position uncertainty is then

used during the track finding (seeding and applying the vertex constraint) and for the secondary vertex reconstruction. The combined track finding−fitting in the 4th Concept detectors consists of three passes:

1. Initial inward reconstruction pass. The overall track finding starts with the track seeding in the outermost layers of the drift chamber. Then, for each track reconstructed in the drift chamber, its prolongation in the Vertex Detector is obtained.

2. Outward reconstruction pass and matching with the outer detectors. From the innermost Vertex Detector layer the Kalman Filter proceeds in the outward direction. During this second propagation the space points with large χ^2 contributions are removed from the track. Finally, the Kalman filter continues into the Muon Spectrometer where the clusters found are matched to the prolonged track.

3. Final reconstruction pass. In this pass the primary track is refitted back to the primary vertex or, in the case of the secondary tracks, as close to the vertex as possible. During this pass the secondary vertices are reconstructed (V0s, cascade decays and kinks) and the information is stored in the Event Summary Data (ESD).

4. VXD Standalone Tracking. The clusters of the VXD which have not been used in the previous steps are fed to the Standalone VXD tracker.

5. Muon Spectrometer Standalone Tracking. The clusters of the Muon Spectrometer which have not been used in the previous steps are fed to the Standalone Muon Spectrometer tracker.

VXD Standalone Tracking This strategy adopted for the reconstruction of tracks that have an insufficient number hits in the VXD+DCH combined. In this case, the track finder looks for seeds in the VXD alone, taking the hit in the VXD closer to the IP and making a prolongation line from the beam point to the next layer. The track seed is constructed using either three *RecPoints* inside a narrow road along the prolongation line or using two such *RecPoints* plus the beampoint. The cluster finding continues by prolongating to next layers each helix constructed from a seed. After finding clusters, all different combination of clusters are refitted with the Kalman Filter and the tracks with lowest χ^2 are selected. Finally, in order to increase the reconstruction efficiency and the purity of the procedure, the process is repeated attempting to find tracks on an enlarged road constructed looping on the first point on different layers and all the subsequent layers.

1.6 Particle Identification

The Particle Identification method implemented in ILCRoot is described in more detail in Ref. [**?**]. The PID informations obtained from each individual detector are combined in a Bayesian way. This approach has the advantage that PID signals of a different nature (e.g. dE/dx and shower shapes) can be combined together even in case of signals distributed according to very different probability density functions. The method proceeds as follows:

- We first compute the conditional probability density function $r(s\,|i)$ to observe in some detector a PID signal s if a particle of type i ($i = \mathrm{e};\mu;\pi;\mathrm{K};\mathrm{p}$) is detected. The probability to be a particle of type i if the signal s is observed, $w(i|s)$, depends on $r(s|i)$ and on the frequency C_i for that particles to be registered in the considered experiment.

- Once $w(i|s)$ is evaluated with the above method, we compute the Global PID combining the PID measurements from all the detectors in a Bayesian way. This is done by multiplying the $w(i\,|s)$ for each detector and normalizing the results to the unity.

1.6.1 Reconstruction of Jets

The quality of jet reconstruction is determined by two effects. The first is related to the response of the measuring devices, the second to the algorithm used for clustering the particles into a jet. The topology and the stucture of jets are also important, but we can do nothing to improve them. In this section I will show how calorimetric and tracking information can be combined in order to reconstruct two-jet invariant masses with higher precision than the traditional methods.

As already mentioned in Section **??**, reconstructing jets means to reconstruct their energy and their direction, but while the reconstruction of the energy is essentially related to the energy performance of the calorimeter, the attempt to reconstruct the direction of the jet is attended also by the presence of the magnetic field and by the granularity of the calorimeter. Therefore a traditional jet-finder may lead to a wrong result. The compensation achieved by the triple readout calorimeter, described in Section **??**, helps us in obtaining the correct measurement for the energy without any further correction. The projective geometry of the calorimeter and its longitudinal segmentation allow us also to perform directional measurements, as well. However, the direction will be correct only for neutral clusters, while results obtained in the case of charged clusters will be wrong because of the bending effect of the magnetic field. This effect can be seen with a simple example by comparing the measurement of theta and phi angles performed with the calorimeter for a 5 GeV charged pion