RecPack

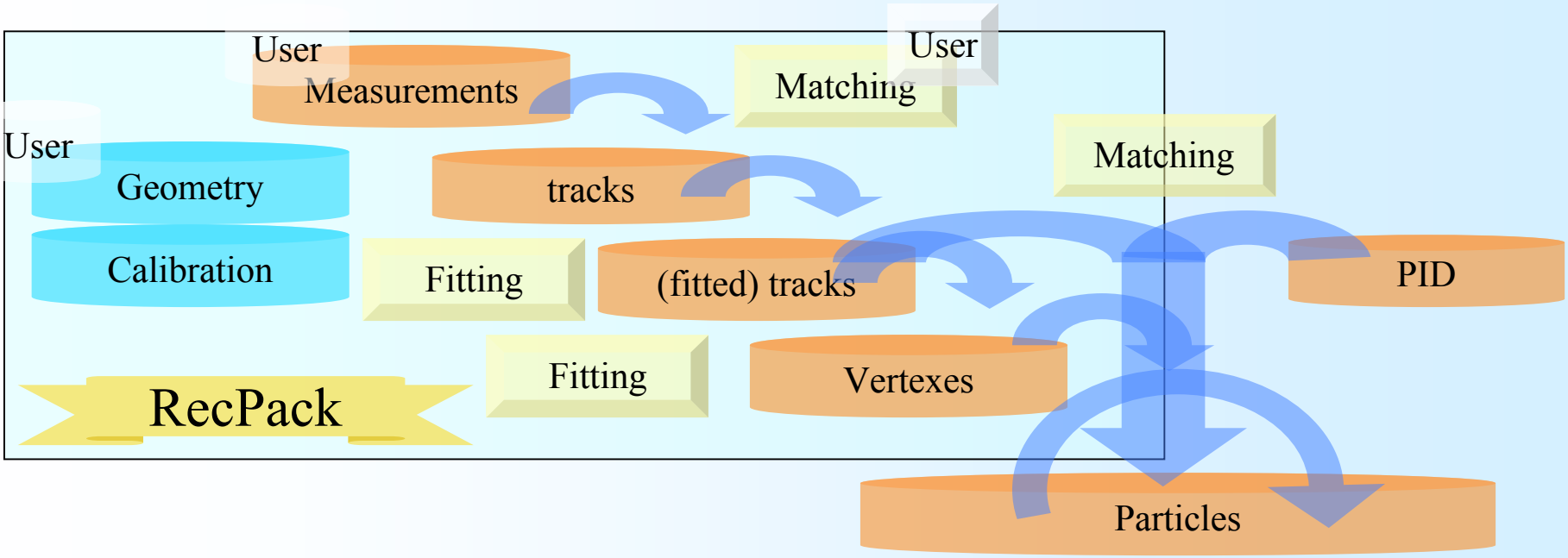
A Reconstruction Toolkit

Jose Angel Hernando
(CERN, Switzerland)

In collaboration with:
Anselmo Cervera Villanueva (Geneve, Switzerland)
Juan José Gómez Cadenas (Valencia, Spain)

ACAT03, @ KEK, 2003/12/02

the HEP Montjuïc data fountain



- **Montjuic data fountain:** ***data & tools*** that we use in any reconstruction program in HEP:
  - **Data:** information classes
  - **Tools:** operate on the data classes

Data

Tool

- These classes admit an **interface**
- Common tools of fitting & matching **are general and run in these interfaces**
- **Users** can implement (derive) their **own classes**
  - I.e, data: geometry & measurements, matching tools

Montjuic is a mountain in Barcelona with a beautiful and always changing fall and fountain

What is RecPack?

- **Idea:**
  - Most of the ***tracking reconstruction*** programs (pattern & fitting) done *in HEP* ***use common algorithms***.
    - ***I.e Kalman Filter***
    - ***Helix Model***
  - Code the common algorithms in a general package

- **RecPack is a C++ toolkit :**
  - ***To reconstruct & fit trajectories***.
  - Fit trajectories to a model and estimate model parameters and errors
    - Ie. Using the ***Kalman Filter***
  - Match measurements & trajectories
  - Navigate states in a n-dimensional space

- **Modular, extendible, friendly**
  - **Different modules** light connected
    - ***Fitting, Model, Geometry & Navigation, Matching***
  - **Extendible:** "developer" user can implement its own data classes or tools from interfaces.
  - **Friendly:** "client" user interacts via an unique Manager

- **and general…**
  - It can be apply to any dynamic system:
    - *Evolution of a state in a space according with a model*
    - *Fitting a trajectory to a model*
  - Ballistic problems, stock market,…

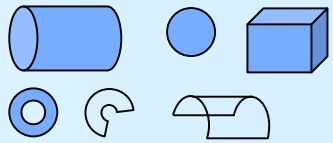
RecPack Manager & Services

The Manager:

User access to the services

The services:

1. store of data & tools
2. provide the package functionality

service name	*methods*	*elements*
Geometry	**Access to geometry** Volume & surfaces Properties of volumes and surfaces <T>	
Model	**Access to models** **Access to model tools that operate on states** equation , propagator surface intersector, projectors noisers	**Models:** Sraight line, Helix in B field **Noisers: MS**
Navigation	**Access to Navigators** **propagate states to any surface and length** **Access to Inspectors**	**Navigators** **Inspectors (Helix, MS noiser, counters)**
Fitting	**Track fitting,** **Vertex fitting**	**Least squares** **Kalman Filter**
Matching	**matching trajectory-trajectory** **matching trajectory-measurement** **pattern recognition methods**	
Simulation	**Simulate a trajectory & measurements**	**RecPack Simulator**

Manager
Access to services

Extensions (more services)

Extensions (more methods)

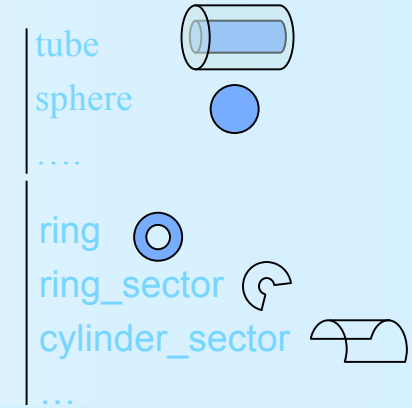
Extensions (more elements)

Geometry service

Geometry

Access to geometrical setups:

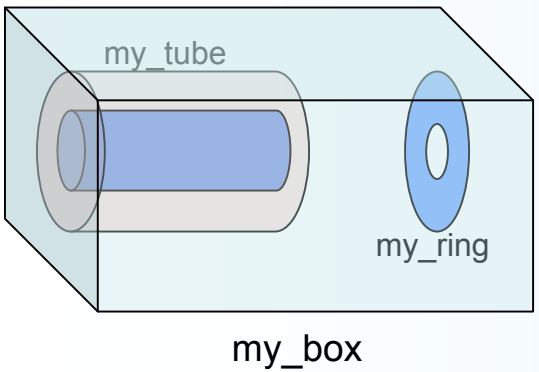
- Volumes & surfaces into a mother volume
- Associated properties (template) any volume or surface

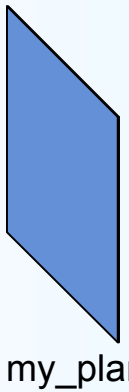


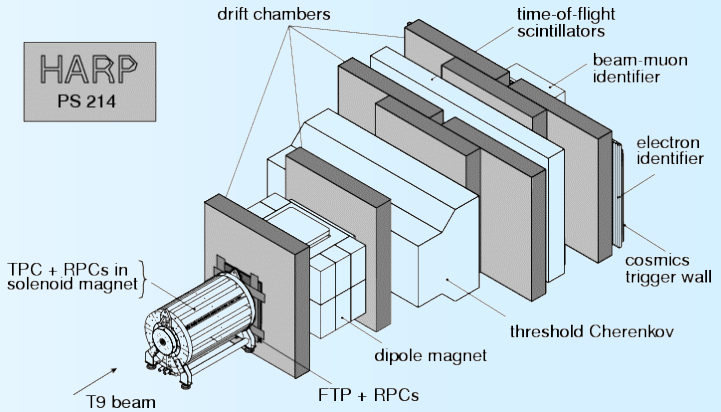
add_volume("my_box", "box3D", pos, axes, size);
add_surface("my_plane", "rectangle", pos, axes, size);

volumes may have any dimension
$d_{surface} = d_{volume} - 1$

add_volume_to_volume("my_box", "my_tube", "tube", pos, axes, size);
add_surface_to_volume("my_box", "my_ring", "ring", pos, axes, size);







Navigation service

Navigation

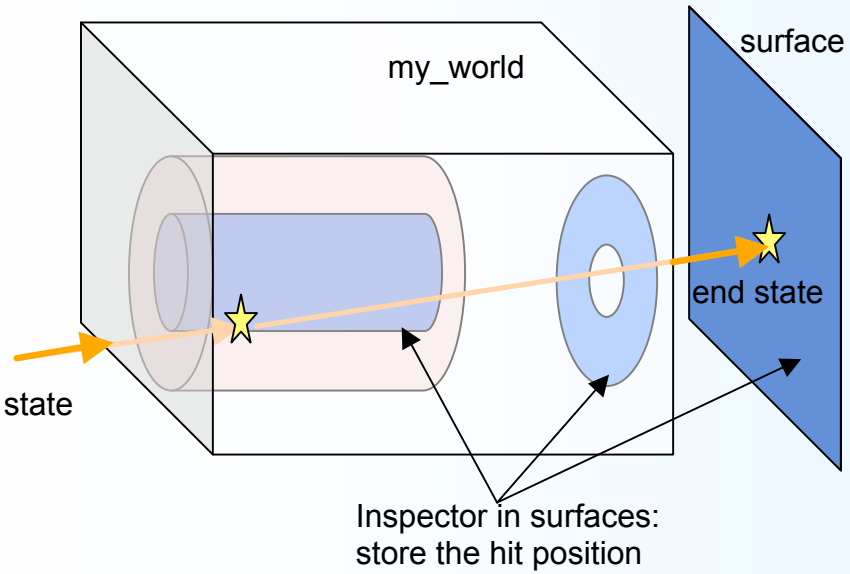
Navigator:

- propagate an state in a setup **via steps**
- At each step inspectors are called

Inspectors:

- They do external operations at each step:
  - *User counters,*
  - *Modify propagation (looking at material of the volume),…*
- Can be associated to any surface or volume

propagate(state, surface);



User can:

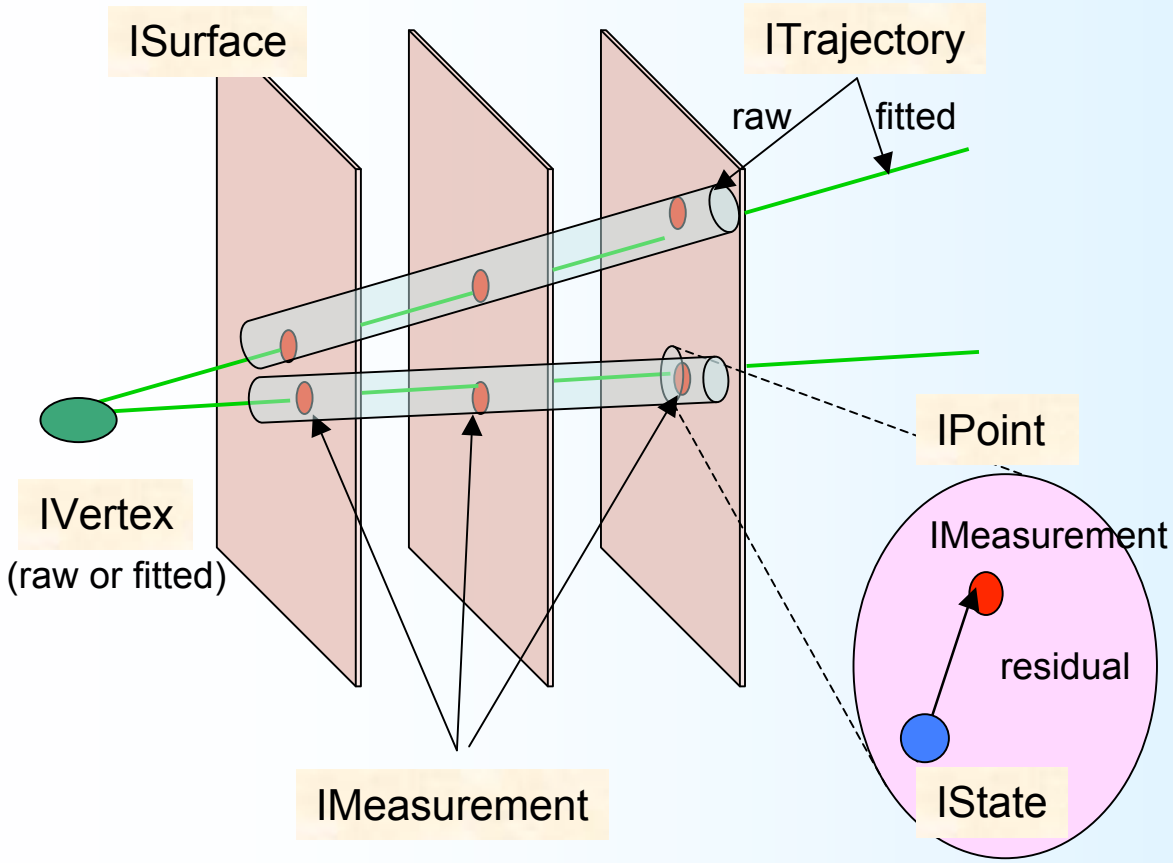
- implement analytic intersection for a given:
  - *model and surface.*
- Establish a sequence of surfaces and volumes to intersect!

User can navigate in parallel setups:

- *Material (X0),*
- *Physical (B field)*
- *User setup(counters)*

Data Interfaces Classes

This classes are interfaces (generic)



IMeasurement

- Vector of measures
- Resolution matrix

ie: (x,y) measurement

IState

- Vector of parameters
- Covariance matrix

ie: straight line (x,y,x',y')

ITrajectory

- A collection of states
- A collection of measurements
- The agreement between both

ie: (LSQ fit to a straight line)

Model service

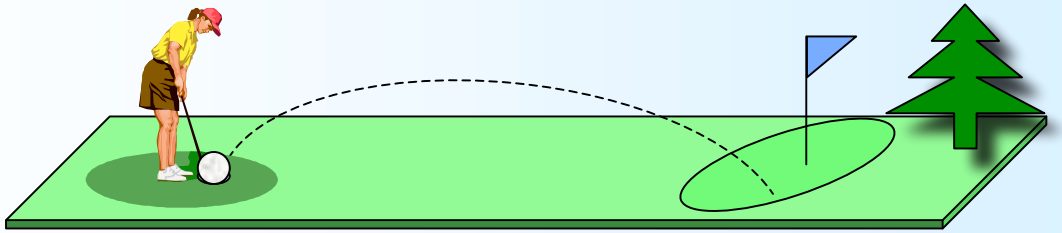
Model

Access to model tools:

- **Equation**
- Projectors (for fitting & matching)
- Propagator, Surface intersectors (to help navigate)
- conversion

Automatically updates the model dependent services

select_model("helix");



Equation

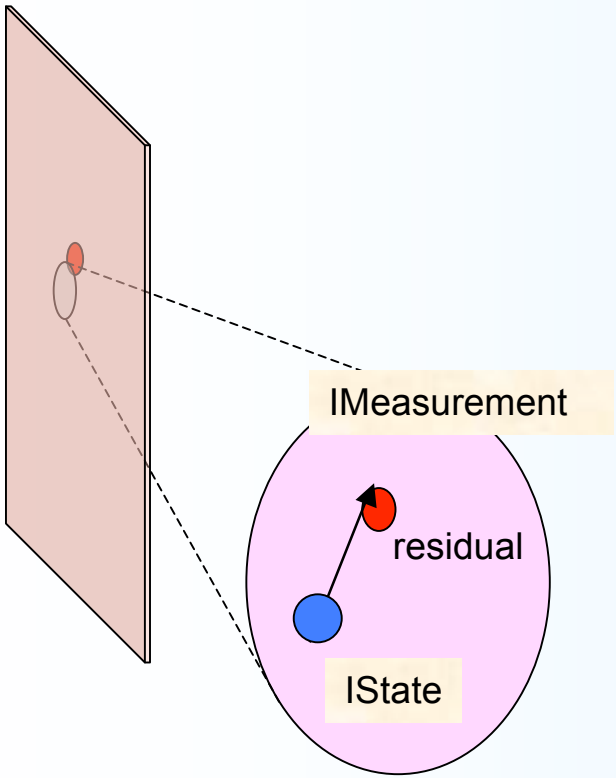
The equation defines the model!

- Evolution of the state vector
  (vector&) vector(double length);
- A "ray" in the geometrical space
  (vector&) position(double length);
  (vector&) direction(double length);

Evolution of the state in the parameter space

Evolution of the state in the geometrical space
(all what we need to navigate!)

Model Tools: projectors



Projector

- A tool that depends on model & measurement type
- Project an state into a measurement:

- State is in the model parameters space
  - *Helix: (x,y,x',y',q/p)*
- Measurement is in an internal space
  - *(u,v) rotated with respect (x,y) an angle φ*
- The projection "reduces" the information of the state vector to be compared with the measurement
  - *In the linear is a matrix **H***

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ x' \\ y' \\ q/p \end{pmatrix}$$

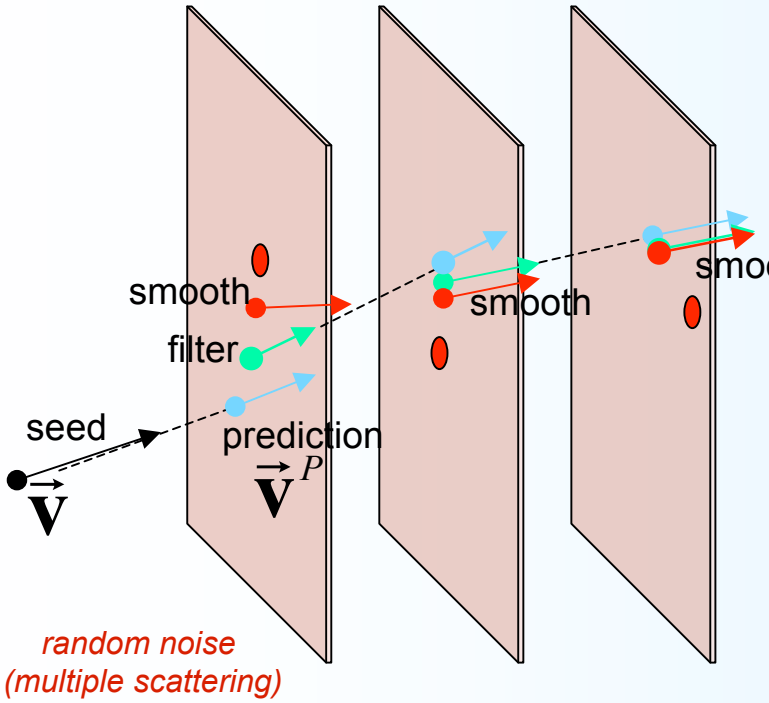
Projectors deals with alignment & calibration

- **The projection converts "global" to "local"**

Fitter: Kalman Filter

- **Kalman Filter:**
  - Used for track fitting by most of HEP experiments
  - Easy to include random noise processes (ms) and systematic effects (eloss)
  - It is a local and incremental fit (dynamic states)

    We can do simultaneously fitting & patter recognition



IPropagator

$$\vec{v}^{\,P} = f(\vec{v}) \xrightarrow[linear]{} \vec{v}^{\,P} = F\vec{v}$$

transport matrix

$$C^P = FCF^T + Q$$

noise matrix (ms)

IProjector

$$\vec{w} = h(\vec{v}) \xrightarrow[linear]{} \vec{w} = H\vec{v}$$

Projection matrix

$$\vec{r} = \vec{w} - \vec{m}$$

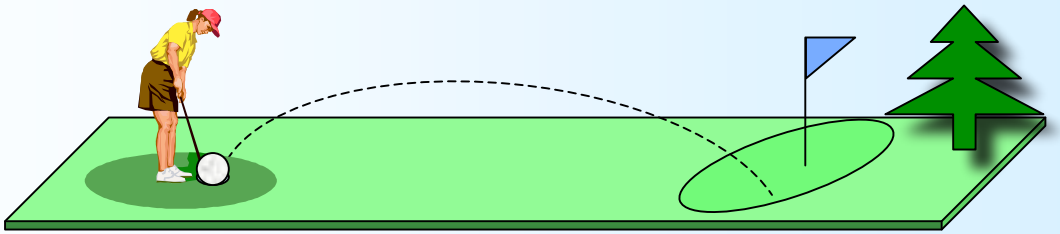
residual = projection - measurement

Example of model tools

Equation	Noise estimators	Systematic effect estimators	surface intersectors	finite surfaces	Projectors
straight line in any dimension	multiple scattering	energy loss	plane	rectangle ring	2D
helix in variable B field	Energy loss		cylinder	cylinder cylinder_sector	3D
			sphere	sphere_sector	rϕ

Adding your model is straight forward!

Model



parabola	wind fluctuations	wind	earth surface	green	2D

Matching and simulation service

We can construct new services:

Matching & Simulation

Using:

- **Navigation & Fitting & Model** services
- Model: propagator & projectors

Matching

match(trajectory, trajectory);
match(trajectory, measurement);
match(state,measurement);
match(trajectory, state);

Use for pattern recognition

Match using the projectors

Future plans: implement pattern recognition "logics"

Simulation

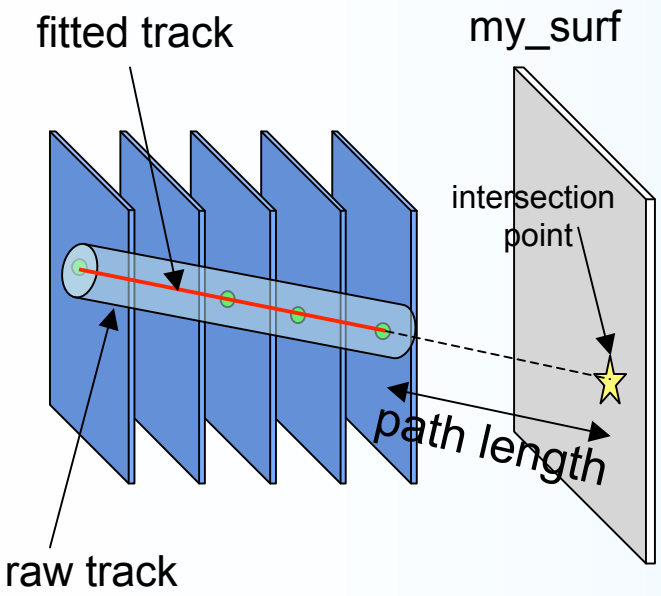
simulate_trajectory(trajectory, seed_state);

simulate measurements along a trajectory given a seed state

Future plans: interface with Geant4

Example 1

- **Fit a single track in a single volume and compute the path length to a given surface**



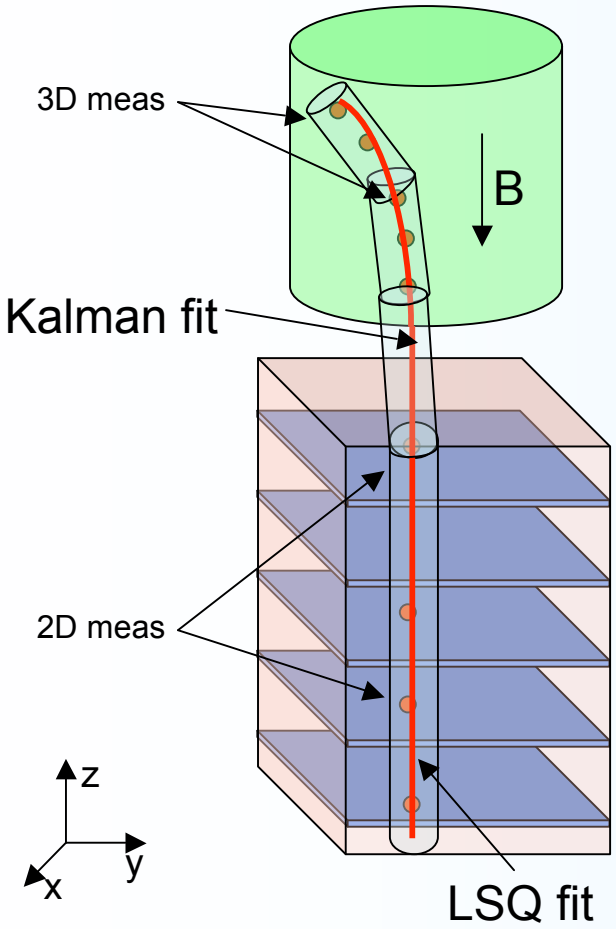
path length = 28 cm

c++ code

```
// Create a track and fill it with measurements
BITrajectory track;
for (i=0; i<4 ; i++){
   IMeasurement& meas[i] = BIMeasurement( pos, pos_error, “xy”)
   track.add_measurement( meas[i] );
}
// Fit the track by Kalman
fitting_svc().fit( “Kalman”, track, seed_state );

// Retrieve a previously defined surface
ISurface& surf = geometry_svc().surface(“my_surf”);

// Computes the path length to the specified surface
navigation_svc().path_legth( track, surf, length);

// Print out the path lenght
std::cout << “path length = “ << legth << std::endl;
```

Example 2

- **Fit a single track in several volumes with different models and different measurement types**



c++ code

```
// Create a track and fill it with 3D measurements
BITrajectory  track1;
for (i=0; i<5 ; i++){
   IMeasurement& meas[i] = BIMeasurement( pos, pos_error, "xyz")
   track1.add_measurement( meas[i] );
}
// Create a track and fill it with 2D measurements
BITrajectory  track2;
for (i=0; i<4 ; i++){
   IMeasurement& meas[i] = BIMeasurement( pos, pos_error, "xy")
   track2.add_measurement( meas[i] );
}
// Fit  the second track by Least squares
fitting_svc().fit( "LSQ",  track2);

// Merge both tracks
track1.add_segment( track2);

// Fit the whole track by Kalman using the previous fit as seed
fitting_svc().fit( "Kalman", track1,  track2.first_state() );
```

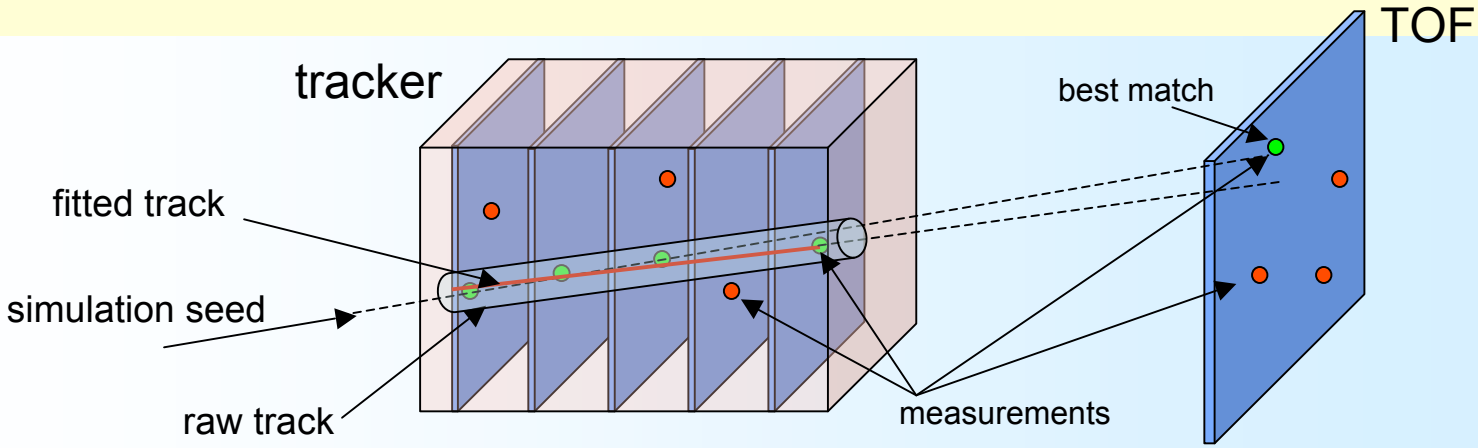
Example 3

- **Simulate a particle traversing several volumes with Geant4, reconstruct tracks in "tracker" and match with "TOF"**

c++ code

```
// Set the Geant4 simulator
simulation_svc().set_simulator("Geant4");

// Simulate a track
simulation_svc().simulate_measurements( simul_seed );

// Find tracks in "tracker" applying predefined PR logic
matching_svc().set_property( "tracker", "PRLogic", "planar" );
matching_svc().find_trajectories( "tracker" , track_vector );

// Fit the first track by Kalman
fitting_svc().fit( "Kalman", track_vector[0], fit_seed );

// Look for the best matching hit in the TOF
matching_svc().best_matching_measurement( "TOF", track_vector[0], meas );
```

For future plans



Clients

RecPack-1

- *RecPack* **was born in *HARP* (CERN)**
- ***MICE* (RAL)**

RecPack0

- ***SciBar* detector, which is part of *K2K* (Japan)**
- **Design of future neutrino experiments: HERO**
- **Trigger studies on *LHCb* (CERN)**
- **Open vertex detector at *LHCb (CERN)***

RecPack-0

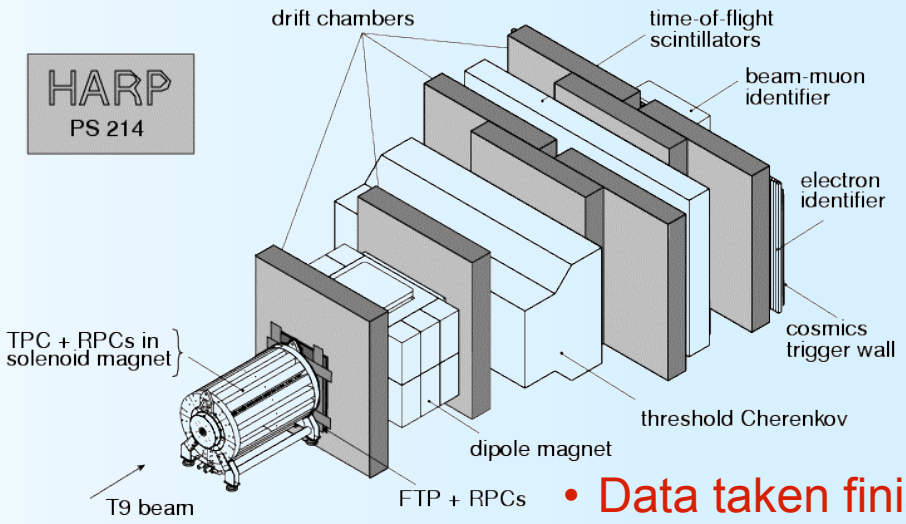
unique library: requires only *CLHEP*

source in: http//evalu29.ific.uv.es

compile: automake, or *CMT*

Linux gcc2.95.2, gcc3.2

some examples with *GAUDI*



- Data taken finished
- On going analysis

Conclusions

RecPack

- **RecPack is a toolkit to built a reconstruction program:**
  - **Does: Navigation, Matching & Fitting**
- **Its modular structure allows extensions in any direction**

  data types volumes, surfaces, measurements, …
  tools models, navigators, simulators, …

- **It is setup independent**
- **It is being successfully used by four HEP experiments**
- **If you want to play, please contact us:**

Jose.Angel.Hernando@cern.ch
Anselmo.Cervera@cern.ch
Juan.Jose.Gomez.Cadenas@cern.ch