RecPack

reconstruction toolkit

Anselmo Cervera Villanueva

University of Geneva

(Switzerland)

with:

Juan José Gómez Cadenas

José Ángel Hernando Morata

NSS 2003

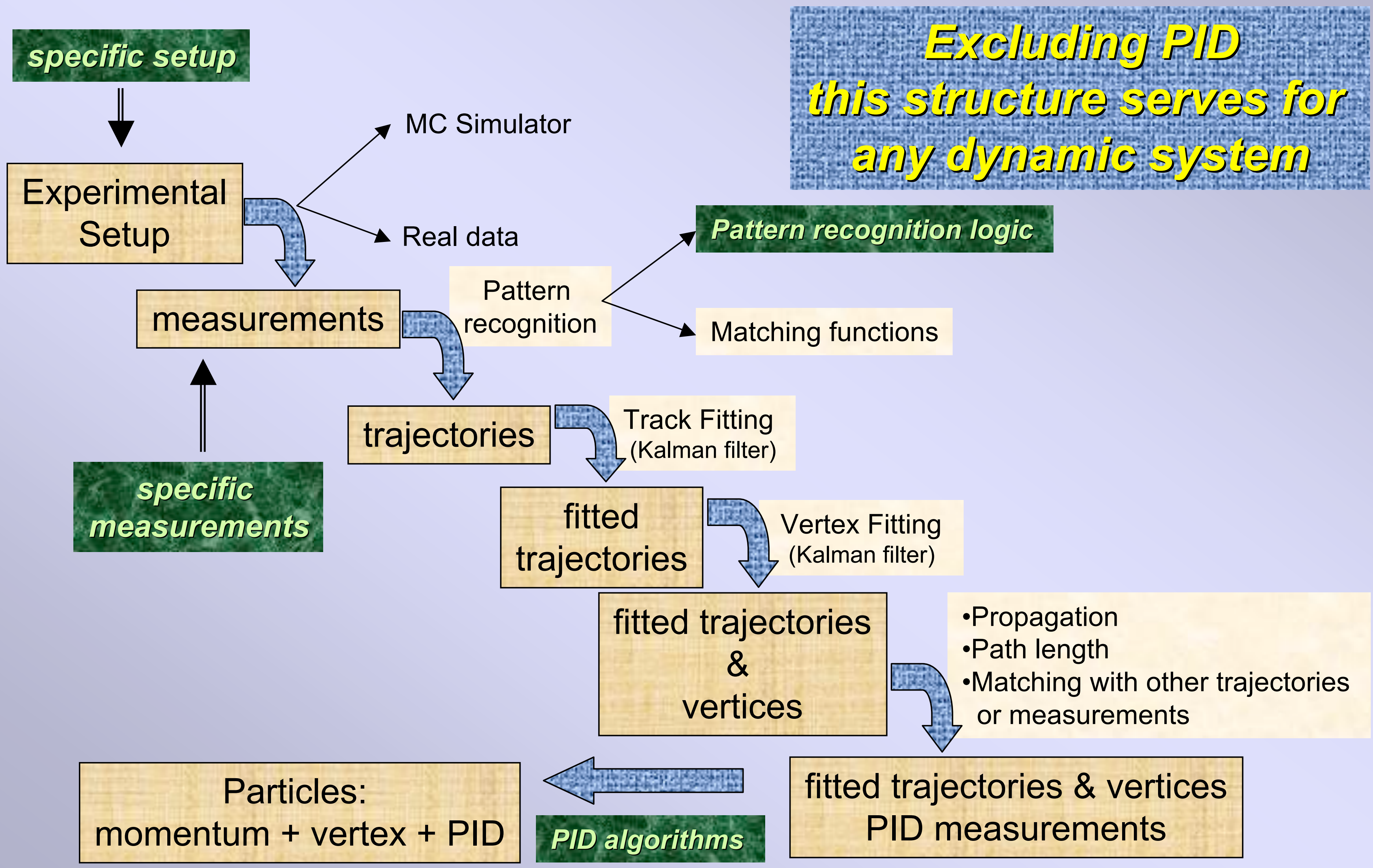
24 October 2003

Introduction

- ***RecPack*** is a reconstruction toolkit:
  - fitting, propagation, matching, toy simulation …
- which follows a c++ object-oriented approach

  $\Longrightarrow$ modular, extendible
- It is setup independent, and therefore
- it can be applied to any dynamic system
  - HEP detectors
  - Medical imaging
  - Space
  - Meteorology
  - Stock market
  - ….

Reconstruction in HEP

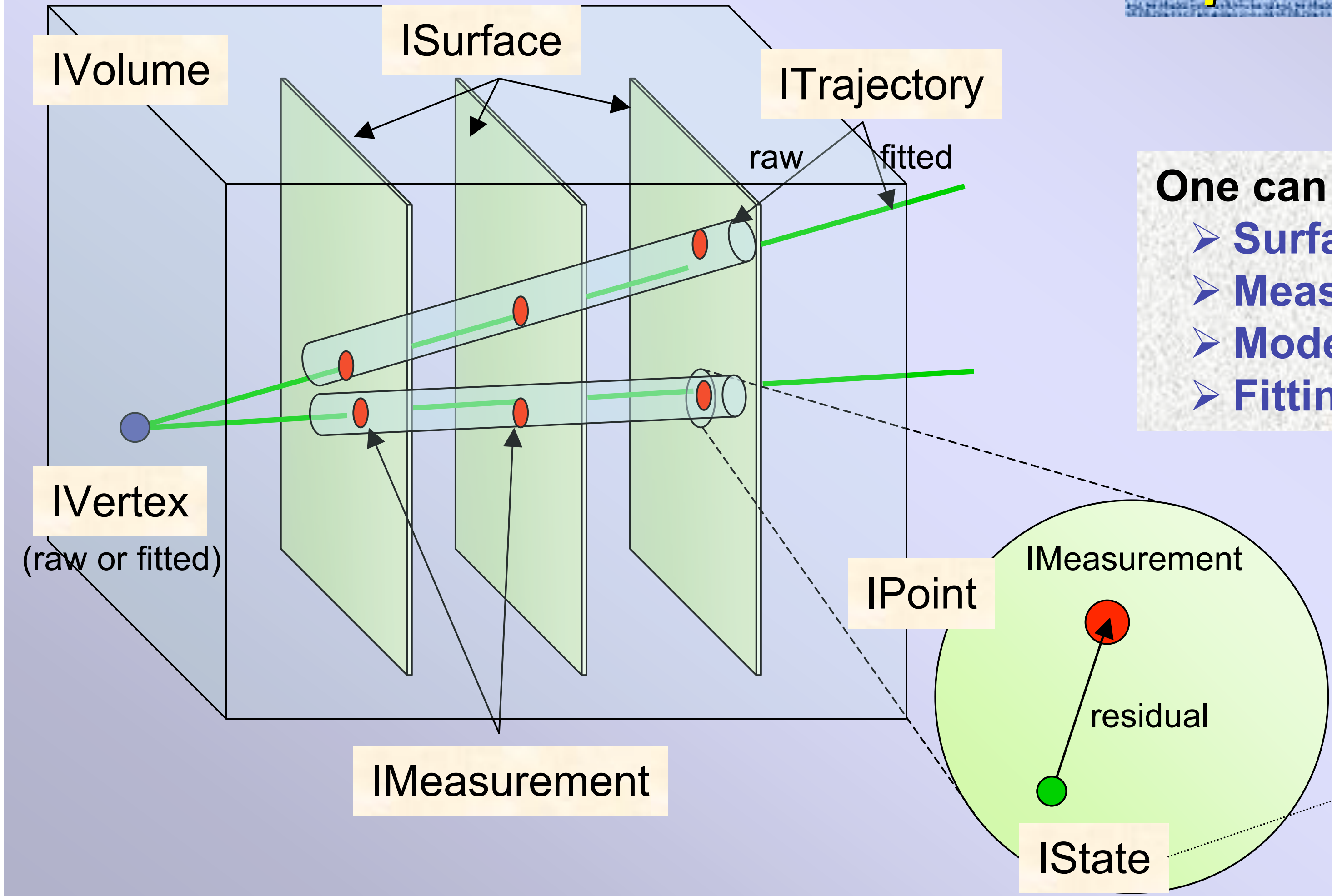


The data classes

This structure exists in any dynamic system

All data classes are pure interfaces

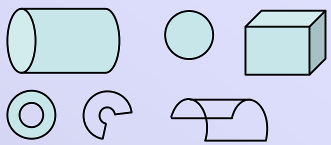
One can use any kind of :
- **Surface/volume**
- **Measurement**
- **Model**
- **Fitting method**



Particle example

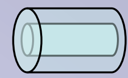
- position
- momentum
- cov matrices
- charge

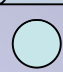
RecPack services

service name	***services***	***available tools***	***extensions***
Simulation	Simulating trajectories and measurements	RecPackSimulator	Geant4 my_simulator
Matching	matching functions pattern recognition methods	trajectory-trajectory trajectory-measurement trajectory-state state-measurement	my_pattern_recognition
Fitting	track fitting vertex fitting	Least squares Kalman Filter	my_fit
Navigation	propagate to any surface compute path lengths	RecPackNavigator	Geant4 my_navigator
Model	mathematical equations for: propagation, intersection with surfaces, random noise (ms) and systematic effect estimators (eloss), ...	straight line, helix, ms, eloss, etc	my_model my_noiser my_intersector
Geometry	volumes surfaces		my_volume_type my_surface_type
Property	volume properties model properties, etc	X0, density, magnetic fields, stepping, etc	my_property

Geometry

add_volume("my_box", "box3D", pos, axes, size);

tube 

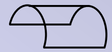
sphere 

….

add_surface("my_plane", "rectangle", pos, axes, size);

ring 

ring_sector 

cylinder_sector 

…

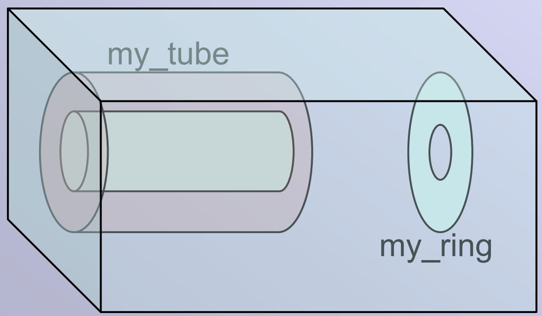
volumes may have any dimension

$$d_{surface} = d_{volume} - 1$$

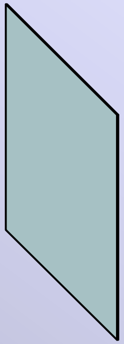
---

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

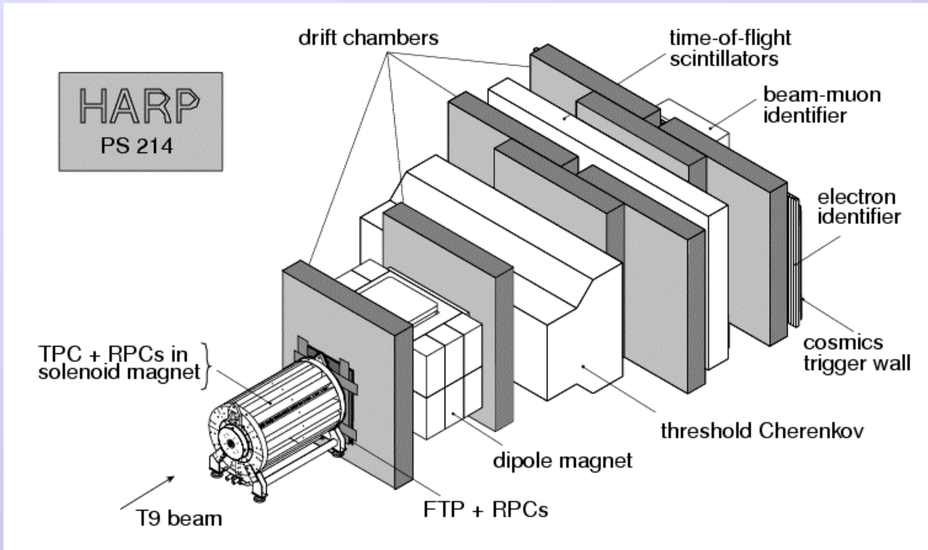
add_surface_to_volume("my_box", "my_ring", "ring", pos, axes, size);



my_box



my_plane

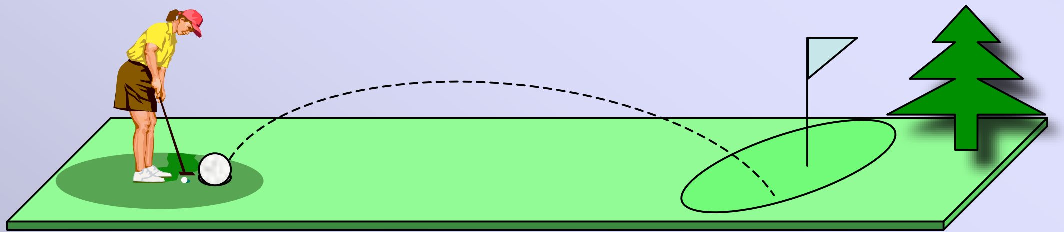


The *RecPack* mathematics

Models	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

All these pieces live in independent containers & ***implement the corresponding interface***

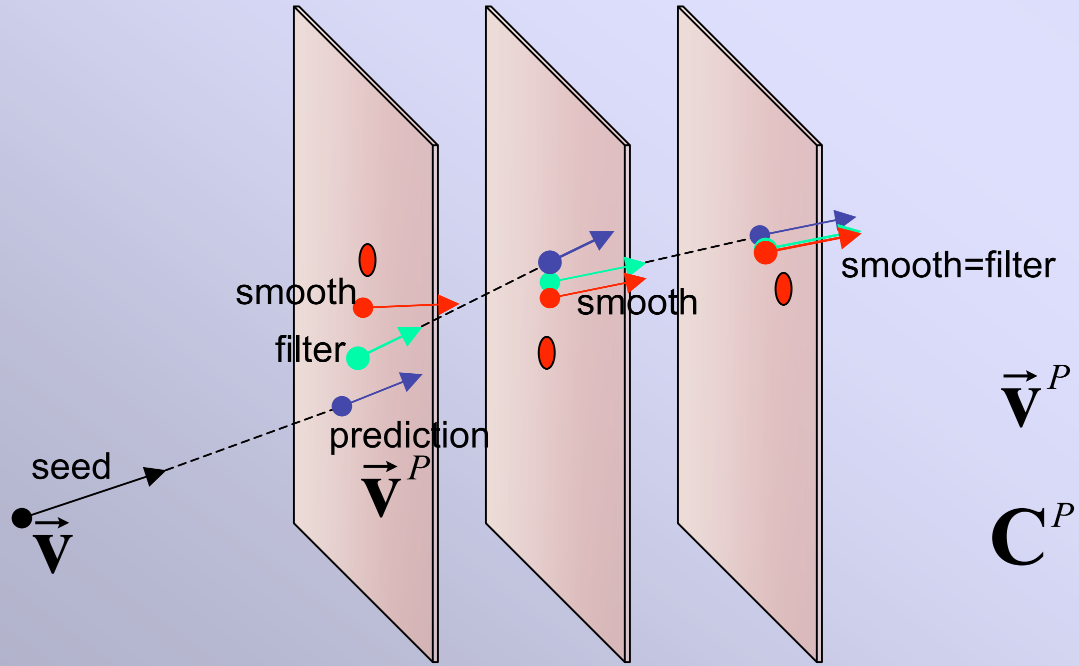
→ ***Extensions are straight forward***



parabola	wind fluctuations	wind	earth surface	green	2D

Fitting: Kalman Filter

- Used for track and vertex fitting by most of HEP experiments
- Easy to include random noise processes (*multiple scatt.*) and systematic effects (*eloss*)
- It is a local and incremental fit ⟶
  - simultaneous pattern recognition and fitting
  - detection of outliers



Any other fitting algorithm can be plugged in

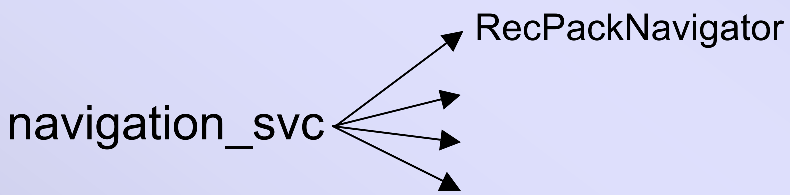
$$\vec{\mathbf{v}}^P = f(\vec{\mathbf{v}}) \xrightarrow[linear]{} \vec{\mathbf{v}}^P = \mathbf{F}\vec{\mathbf{v}}$$

$$\mathbf{C}^P = \mathbf{F}\mathbf{C}\mathbf{F}^{\mathrm{T}} \;\blacksquare\; \blacksquare$$

noise matrix (ms)

Navigation

- A ***INavigator*** propagates states and trajectories through different volumes



RecPackNavigator

- Two types of propagation steps:
  - To a surface
  - A given length

volume properties	propagation type
homogeneous	surface-surface
inhomogeneous	dynamic stepping

- ***Surfaces are intersected analytically***

- After each step a list of ***IInspector***'s is called:
  - Change volume properties
  - Model conversion
  - Sum up intermediate path lengths
  - Set length of the next step
  - Counters
  - …
- Inspectors actuate over:
  - volumes (after each internal step)
  - surfaces (after intersection)
- User defined inspectors can be added to any surface or volume

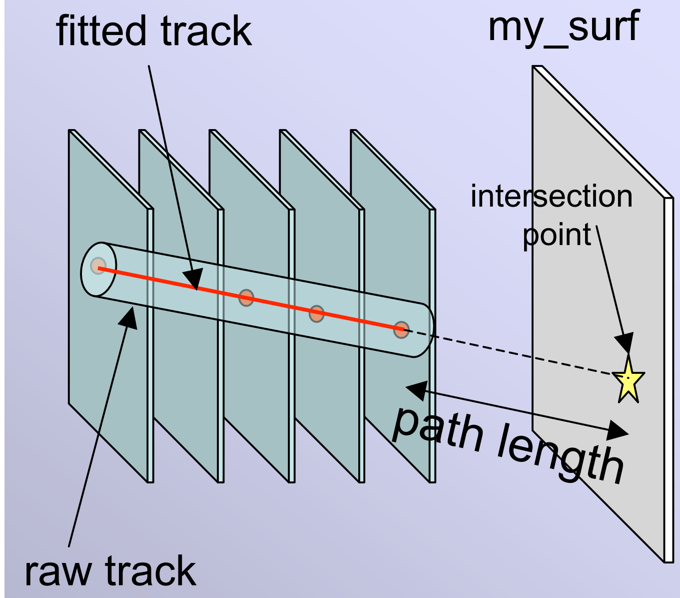
Any other navigator can be plugged in

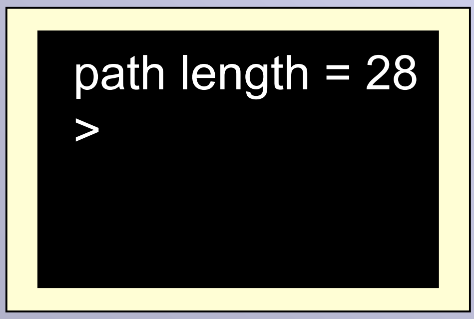
class *Geant4Navigator*: public INavigator

- bool propagate_to_surface(…);
- bool propagate_length(…);

Exercise 1

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



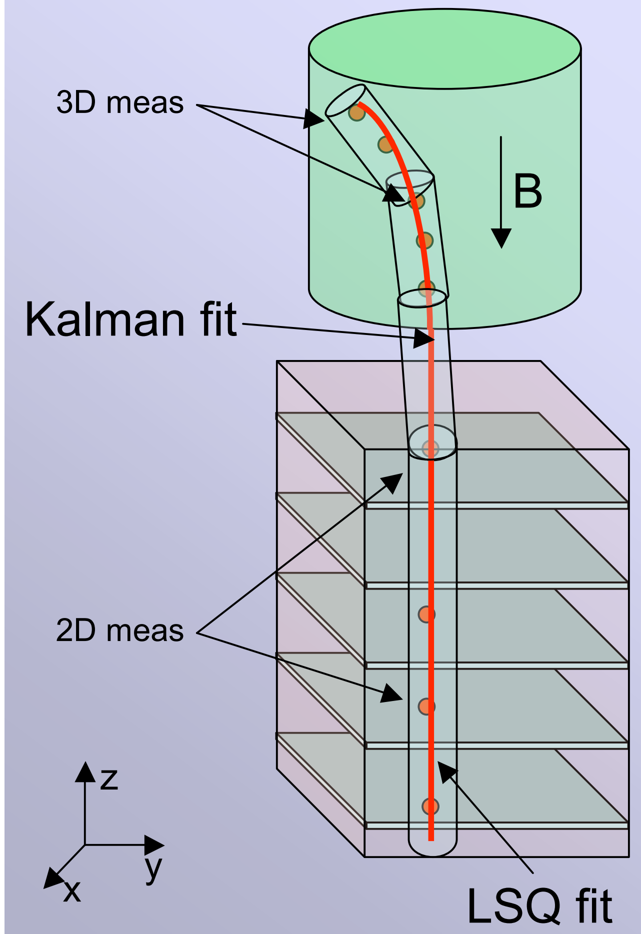


c++ code

```
// 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;
```

Exercise 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() );
```

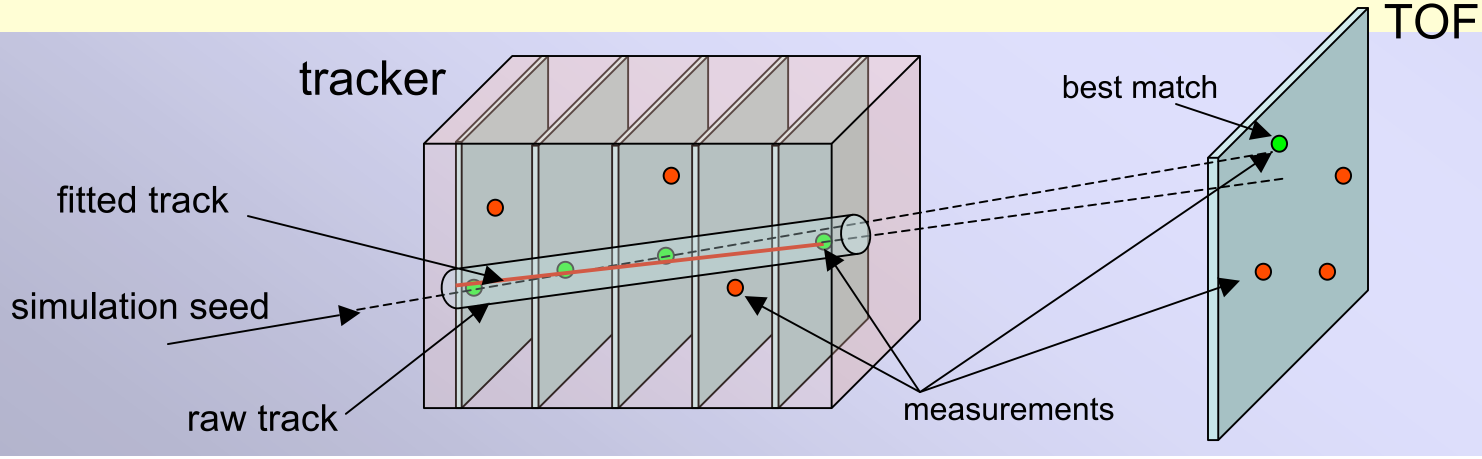
Exercise 3

- 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], best_meas );
```

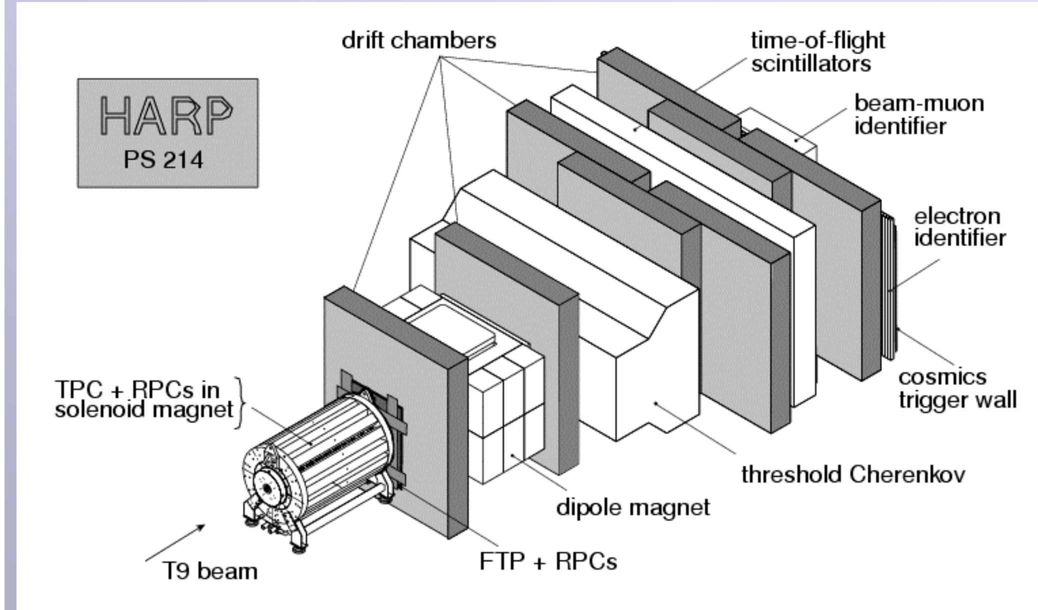
Still science fiction
but
Reality in few months



Clients

RecPack-1
- ***RecPack*** was born in ***HARP*** (CERN)
- ***MICE*** (RAL)

RecPack0
- ***SciBar*** detector, which is part of ***K2K*** (Japan)
- Trigger studies on ***LHCb*** (CERN)
- Design of future neutrino experiments: HERO



- Data taken finished
- On going analysis

RecPack in HARP

- Kalman and LSQ track fitters
- Kalman vertex fitter
- Straight line and Helix models
- Inhomogeneous fields
- Multiple scattering
- Propagation to any surface
- Path length functions
- Matching functions

Future extensions for HEP

- Common pattern recognition logics:
  - 3D: TPC, …
  - 2D: planar, cylindrical, spherical, …

- Common PID algorithms:
  - Cerenkov, rich, …
  - Tof
  - Calorimeter
  - dE/dx

- GUI: (HEP and non-HEP)
  - Detector design and geometry maker
  - Event display
  - Visual debugger

Conclusions

- ***RecPack*** provides the common tools of any reconstruction program → avoids reinventing the wheel !!!
- Its modular structure allows extensions in any direction

  data types → volumes, surfaces, measurements, …
  machines → models, navigators, simulators, fitters, …
- It is setup independent
- It is being successfully used by several HEP experiments
- Interested people may contact:

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