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

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
    - I.e. 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

## Geometry

<table>
<thead>
<tr>
<th>service name</th>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Access to geometry</td>
<td>Volume &amp; surfaces Properties of volumes and surfaces &lt;T&gt;</td>
</tr>
</tbody>
</table>

## Model

<table>
<thead>
<tr>
<th>service name</th>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Access to models</td>
<td></td>
</tr>
<tr>
<td>2. Access to model tools that operate on states equation, propagator surface intersectors, projectors noisers</td>
<td></td>
</tr>
</tbody>
</table>

## Navigation

<table>
<thead>
<tr>
<th>service name</th>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Access to Navigators</td>
<td></td>
</tr>
<tr>
<td>2. Propagate states to any surface and length</td>
<td></td>
</tr>
<tr>
<td>3. Access to Inspectors</td>
<td></td>
</tr>
</tbody>
</table>

## Fitting

<table>
<thead>
<tr>
<th>service name</th>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Track fitting, Vertex fitting</td>
<td></td>
</tr>
</tbody>
</table>

## Matching

<table>
<thead>
<tr>
<th>service name</th>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. matching trajectory-trajectory</td>
<td></td>
</tr>
<tr>
<td>2. matching trajectory-measurement</td>
<td></td>
</tr>
<tr>
<td>3. pattern recognition methods</td>
<td></td>
</tr>
</tbody>
</table>

## Simulation

<table>
<thead>
<tr>
<th>service name</th>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Simulate a trajectory &amp; measurements</td>
<td></td>
</tr>
</tbody>
</table>

## Extensions

- (more services)
- (more methods)
- (more elements)
Geometry service

Access to geometrical setups:
- Volumes & surfaces into a mother volume
- Associated properties (template) any volume or surface

\[
\text{volumes may have any dimension} \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \\
\text{ } \

\text{add_volume("my_box", "box3D", pos, axes, size);} \\
\text{add_surface("my_plane", "rectangle", pos, axes, size);} \\
\text{add_volume_to_volume("my_box", "my_tube", "tube", pos, axes, size);} \\
\text{add_surface_to_volume("my_box", "my_ring", "ring", pos, axes, size);}

Navigation service

**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

Access to model tools:

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

select_model("helix");

**Equation**

- Evolution of the state vector
  \( \text{vector}\&(\text{vector\&}) \text{vector}(\text{double}\ \text{length}); \)
- A “ray” in the geometrical space
  \( \text{vector}\&(\text{vector\&}) \text{position}(\text{double}\ \text{length}); \)
  \( \text{vector}\&(\text{vector\&}) \text{direction}(\text{double}\ \text{length}); \)

*The equation defines the model!*

*Automatically updates the model dependent services*

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 \(\varphi\)
  - The projection “reduces” the information of the state vector to be compared with the measurement
    - *In the linear is a matrix \(H\)*

Projectors deals with alignment & calibration

- The projection converts “global” to “local”

\[
\begin{pmatrix}
  u \\
  v
\end{pmatrix} =
\begin{pmatrix}
  \cos \varphi & \sin \varphi & 0 & 0 & 0 \\
  -\sin \varphi & \cos \varphi & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  x' \\
  y' \\
  q/p
\end{pmatrix}
\]
**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 & pattern recognition*
### Example of model tools

<table>
<thead>
<tr>
<th>Equation</th>
<th>Noise estimators</th>
<th>Systematic effect estimators</th>
<th>surface intersectors</th>
<th>finite surfaces</th>
<th>Projectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>straight line in any dimension</td>
<td>multiple scattering</td>
<td>energy loss</td>
<td>plane</td>
<td>rectangle ring</td>
<td>2D</td>
</tr>
<tr>
<td>helix in variable B field</td>
<td>Energy loss</td>
<td>cylinder</td>
<td>cylinder cylinder_sector</td>
<td>3D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sphere</td>
<td>sphere_sector</td>
<td>rφ</td>
</tr>
</tbody>
</table>

Adding your model is straight forward!

<table>
<thead>
<tr>
<th>parabola</th>
<th>wind fluctuations</th>
<th>wind</th>
<th>earth surface</th>
<th>green</th>
<th>2D</th>
</tr>
</thead>
</table>
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

```cpp
// 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_length(track, surf, length);

// Print out the path length
std::cout << "path length = " << length << std::endl;
```

path length = 28 cm
Example 2

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

```cpp
// 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”

```cpp
// 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 );
```
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