Available online at www.sciencedirect.com
SCIENCE DIRECT®
Nuclear Instruments and Methods in Physics Research A 534 (2004) 180–183
NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A
www.elsevier.com/locate/nima

"RecPack" a reconstruction toolkit

A. Cervera-Villanueva[a], J.J. Gómez-Cadenas[b], J.A. Hernando[a,c,*]

[a]*CERN, Université the Genève, CH-1211 Geneva 23, Switzerland*
[b]*Universitat de València, Spain*
[c]*Universidade de Santiago de Compostela, Spain*

Available online 2 August 2004

Abstract

We present a C++ toolkit to do tracking and vertex reconstruction. The toolkit incorporates common fitting methods, as the Kalman Filter, a framework to define a detector setup, a general navigation and a simple simulation. Furthermore, the toolkit provides a collection of interfaces which facilitates the addition of new fitting methods, trajectory models, geometrical objects, pattern recognition logic, etc. Although the toolkit was originally developed to be used in High Energy Physics, it could be applied to other fields.

PACS: 29.40.Gx; 29.85.+c

Keywords: C++ toolkit; Reconstruction; Kalman filter

1. Introduction

In high energy physics (HEP), one frequently faces the problem of reconstructing (modeling) the evolution of a dynamic system (states) from a set of measurements. Three different levels of abstraction show up addressing this problem: a purely abstract level, which contains the fitting algorithms, such as the Kalman Filter [1,2], a second level which admits a basic structure (the definition of a setup and the procedure of navigation), and a third level which is completely setup dependent (for example, pattern recognition algorithms).

In "RecPack" we provide the basements for these three levels. The package has a core module with the most common fitting algorithms. It has modules to do generic tasks: definition of a setup, navigation and simulation. And finally, it provides a collection of interfaces, which allow the user to extend the package classes adding new geometrical objects, trajectory models, etc.

In the next section we will give an overview of the classes of the package, and in the sections that follow, we will concentrate on describing the functionality of the different modules (or services).

*Corresponding author. CERN, Université the Genève, Geneva 23211, Switzerland.
E-mail address: jose.hernando@cern.ch (J.A. Hernando).

0168-9002/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.nima.2004.07.074

2. Structure of the package

The classes of "RecPack" are divided into two main groups: data and tools. The data are pure holders of information (they are passive), while the tools operate and modify the data (they are active). With this separation, data do not know about the tools that operate on it, and new tools could be easily added.

The basic types of the package are vectors and matrices. "RecPack" uses HepVector and HepMatrix classes from the CLHEP package. Nevertheless the user has the possibility of replacing them by other vector and matrix classes.

There is a third basic type, HyperVector, made from the aggregation of a vector and a covariance matrix. The two main data classes of "RecPack", measurement and state, are essentially HyperVectors. A measurement is a vector of measured values and their resolution matrix. A state is a vector with the parameters, which define the model (for example, the parameters of a straight line model in 2 dimensions are an offset and a slope), and its covariance matrix.

A trajectory is a two-status data class: initially (before the fit) it contains a collection of measurements. After the fit it holds the resulting fitting parameters (states), and a set of global variables holding the quality of the fit (i.e. the total χ^2, the number of degrees of freedom, etc.). In the most general case, the fitting parameters are local. For this reason we introduced an intermediate object, a point, to accommodate a measurement, its associated state, and the quantities that relate both objects (the residual HyperVector and the local χ^2 of the fit). After the fit, a trajectory can also be seen as a collection of points.

The active classes, called tools, manipulate the data. Most of them work on states.

Tools are stored in services which also manage them. Services have friendly methods to do most of the common operations; they are the main interface of the package functionality to the user.

Finally, the user gains to access all the services via a unique class the "RecPackManager". From this master manager, the user has complete control of the package.

In the following sections, each of the services is treated individually.

3. Geometry

The geometry service provides methods to define a setup, which is built locating surfaces and volumes into a mother volume, and linking them to properties such as materials or fields (i.e. magnetic fields).

The geometrical classes are implemented from the interfaces of surface and volume. Volumes are logical classes contained inside other volumes and surfaces. Both classes admit the relevant *is_inside()* method, that given a point in the space will answer if the point is inside the geometrical object or not, which is used by navigation (see later) to move states in and out volumes.

Volumes and surfaces can have any dimensions. "RecPack" provides a set of pre-defined surfaces for a 3D setup: plane (rectangle and ring), cylinder and sphere (and sectors of both); and three 3D volumes: box, tube[1] and sphere. It is straightforward to add new geometrical classes.

The following line of code shows as example how to add a box volume "tracker" into the setup.

geometry_svc().add_volume("tracker", "box", pos, axis1, axis2, size), where pos, axis1, axis2 and size are vectors.

In this way, complicated setups can be built. (Fig. 1 shows the setup of the HARP detector [3] the first experiment where "RecPack" was applied.)

4. Model

The model service holds and manages tools related with propagation models, such as propagators, intersectors of surfaces, projectors, etc. In "RecPack" we have pre-defined three models: straight line, polynomial and the Helix model (charged particle in a *B*-uniform magnetic field).

A model will always be in a particular representation, which is defined by the parameters of

[1] A tube is defined by two concentric cylinders.

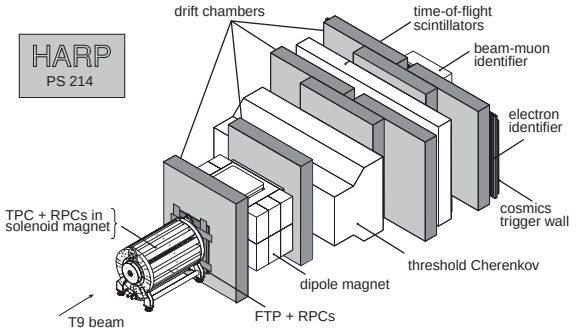

Fig. 1. Setup of the HARP detector at CERN-PS.

the state vector $\vec{\mathbf{v}}$. For example, a 3D straight line could be in a cartesian representation: $\vec{\mathbf{v}}_1 = (x, y, z, \mathrm{d}x/\mathrm{d}z, \mathrm{d}y/\mathrm{d}z)$, or in a spherical one $\vec{\mathbf{v}}_2 = (r, \theta, \phi, \mathrm{d}r/\mathrm{d}z, \mathrm{d}\theta/\mathrm{d}z)$.

The *model equation* is the fundamental tool of the model. It describes the evolution of the state vector as a function of the path length in a geometrical space (needed by the fitter). In addition it provides position and direction at that path length (needed to navigate within the setup).

Most of the model tools could be recreated numerically starting from the model equation. Nevertheless, to save execution time, the user should better provide the analytical tools when possible.

Two of the major tools associated with any model and the propagator and the projector.

4.1. Propagator

The propagator propagates a state (its vector and covariance matrix) to a given surface or a specific length. To do so, it delegates the intermediate calculations to smaller tools, which perform concrete actions: (a) surface intersector, (b) model equation and (c) noiser.

A surface intersector calculates the path length between the state's actual position and a given surface. In general, for each surface type, there is an intersector. Nevertheless, we provide a numeric intersector valid for any model and surface type.

The noiser computes the *random noise* covariance matrix produced after state has been propagated to a given length. For example, the multiple scattering in HEP.

4.2. Projector

The projector converts the state $(\vec{\mathbf{v}}, \mathbf{C}_v)$ into a "predicted" measurement $(\vec{\mathbf{m}}^{\mathrm{pred}}, \mathbf{C}_m^{\mathrm{pred}})$ to be compared with the "empirical" measurement $(\vec{\mathbf{m}}, \mathbf{C}_m)$.

$$\vec{\mathbf{m}}^{\mathrm{pred}} = \vec{\mathbf{h}}(\vec{\mathbf{v}}), \quad \mathbf{C}_m^{\mathrm{pred}} = \mathbf{H}\mathbf{C}_v\mathbf{H}^{\mathrm{T}} \tag{1}$$

where $\mathbf{H} = \partial\vec{\mathbf{h}}/\partial\vec{\mathbf{v}}$ is the projection matrix. In this way, the residual and its covariance matrix can be computed.

The projector depends on the measurement type. An HEP experiment is usually made of subdetectors which provide different type of measurements ("xy", "xyz","$r\phi$", etc.).

Projectors link measurements and states; they are a basic tool, used by other high-level tools as fitters, matching functions, etc.

5. Fitting

The fitting service provides an extensible collection of fitters for trajectories and vertexes. Two fitters (least squares and Kalman) and one vertex fitter (Kalman) are pre-defined in "RecPack".

The fitter takes a *raw trajectory* (which contains measurements) and transforms it into a *fitted trajectory*; that means, it computes the states and the quality of the fit. In the case of a Kalman Filter (for trajectories or vertexes) a seed state must be provided.

The fitter delegates the task of projection and propagation to the model tools described above.

Let us consider a collection of 2D measurements produced by a charged particle in a magnetic field. The measurements have already been introduced in a raw trajectory (*track*) and we now want to fit it. First by least squares and then use the result of the fit (*first_state*) as a seed for re-fitting the track but now using the Kalman filter, the C++ code would be:

fitting_svc().fit("Lsq", "Helix", track); fitting_svc().fit("Kalman", "Helix", track, track.first_state());

6. Navigation

The navigation service provides methods to propagate a state to a given length, surface or volume, inside a setup. The process of navigation includes entering and exiting volumes, changing the model, etc.

In general, the navigation will be done in steps. Before and after each step some actions can take place. The step size in which the surface or volume to stop at and the reason to do so are controlled by a tool named inspector. Inspectors are the "smart" elements of the navigation. They have a wide action; For example, they can set the radiation length on a multiple scattering noiser after the state has entered a volume with a different material, or do user specific tasks, such as simply counting how many times a surface has been reached.

The navigator can change its status dynamically in request of one inspector. In particular, this can be used to change models on the flight; For example, one can start propagating a "helix" state in a volume where a magnetic field was defined, and turn into a "straight line" state when exiting that volume.

7. Matching

The matching service provides methods to execute pattern recognition (PR) algorithms. The task of PR algorithms is to distribute measurements into trajectories, and these into vertexes.

There are two types of PR algorithms: matching functions, which serve to estimate the probability of two objects of being related to each other (i.e. measurement to trajectory), and pattern recognition logics, which define the logical sequence in which such a relations are established. The first in general, while the second may have a strong setup dependence.

Currently, the "RecPack" matching service provides the following matching functions: trajectory to a measurement, trajectory to trajectory and trajectory to vertex. PR logics are not implemented. In the future, we will try to identify common PR logics and include them in the service. For example, in HEP one deals with 2D measurements in parallel planes in similar ways, and this could be coded into a pattern recognition logic.

8. Simulation

The simulation service provides some methods to generate a trajectory and create measurements in a given active setup. This is not a full simulation (in the sense of GEANT). It is more of a debugging tool to check if models, fitters and other tools of "RecPack" are correct.

The simulation will be initialized with some active surface and volumes, the type the measurements they produce and their resolution. The simulation uses the navigation service to propagate the seed state through the setup, it will call a specific type of inspector that will project the state in each active surface (or volume) to produce measurements (using the correspondent projectors, according to Eq. (1)), and these will be smeared randomly (using noisers) according to the resolution.

External simulation toolkits could be imported. To do so, the external classes should fulfill the "RecPack" interfaces.

9. Conclusions

"RecPack" is a modular and extensible reconstruction toolkit. It provides a core service with common fitting methods. It also provides services to define a setup, to do navigation, matching and a simple simulation. The "RecPack" classes admit interfaces, which allow the user to implement their own derived classes to meet the requirements of his reconstruction. That makes "RecPack" a powerful toolkit to address any reconstruction problem.

References

[1] R.E. Kalman, J. Basic Eng. 82 (1960) 35;
R.E. Kalman, R.S. Bucy, J. Basic Eng. 83 (1961) 95.
[2] R. Fruhwirth, M. Regler, Nucl. Instr. and Meth. A 241 (1985) 115.
[3] The HARP Collaboration, ⟨http://harp.web.cern.ch/harp/⟩.