Computer Physics Communications 140 (2001) 21–30

Computer Physics Communications

www.elsevier.com/locate/cpc

# Implementation of an object oriented track reconstruction model into multiple LHC experiments ☆

Irwin Gaines [a], Saul Gonzalez [b], Sijin Qian [b,*]

[a] *Fermilab, Batavia, USA*

[b] *Univ. of Wisconsin, Madison, USA*

**Abstract**

An Object Oriented (OO) model (Gaines et al., 1996; 1997; Gaines and Qian, 1998; 1999) for track reconstruction by the Kalman filtering method has been designed for high energy physics experiments at high luminosity hadron colliders. The model has been coded in the C++ programming language and has been successfully implemented into the OO computing environments of both the CMS (1994) and ATLAS (1994) experiments at the future Large Hadron Collider (LHC) at CERN.

We shall report:

(1) how the OO model was adapted, with largely the same code, to different scenarios and serves the different reconstruction aims in different experiments (i.e. the level-2 trigger software for ATLAS and the offline software for CMS);
(2) how the OO model has been incorporated into different OO environments with a similar integration structure (demonstrating the ease of re-use of OO program);
(3) what are the OO model's performance, including execution time, memory usage, track finding efficiency and ghost rate, etc.; and
(4) additional physics performance based on use of the OO tracking model.

We shall also mention the experience and lessons learned from the implementation of the OO model into the general OO software framework of the experiments. In summary, our practice shows that the OO technology really makes the software development and the integration issues straightforward and convenient; this may be particularly beneficial for the general non-computer-professional physicists. 

*PACS:* 29.40.Gx; 29.85.+c; 29.90.+r

*Keywords:* OO model; Track reconstruction; Kalman filtering; CMS; ATLAS

## 1. Introduction

Nowadays, Object Oriented (OO) technology has become more and more popular in the computing of high energy and nuclear physics. Here we describe an OO model [1] for track reconstruction with simultaneous pattern recognition and track fitting by the Kalman filtering method. It was initially designed using the Booch methodology and coded in the C++ programming language in its first version in 1995 for the CMS experiment [2] at Large Hadron Collider (LHC) at CERN. After the C++ Standard Template Library (STL) became available, the model was re-designed

☆ Presented at the Computing in High Energy and Nuclear Physics (CHEP'2000), Padova, Italy, 2/2000.

* Speaker. Corresponding author. Mailing address: EP-Division, CERN, CH-1211, Geneva 23, Switzerland.

*E-mail address:* sijin.qian@cern.ch (S. Qian).

0010-4655/01/$ – see front matter 
PII: S0010-4655(01)00252-1

at the beginning of 1996 in order to take advantage of the container classes provided by STL. In the fall of 1997, another partial re-design in the OO model was undertaken by introducing an abstract class that clarified and simplified the code. Since 1998 the OO model and C++ code has been successfully re-used, with only minor modification, in ATLAS [3], another major LHC experiment. This demonstrates one of the claimed advantages of the OO technique: ease of re-use. Here, the "re-use" is not only for just some particular classes, but also for the entire model. The modularity enforced by the OO design of the model made this re-use much easier, as the experiment-specific code is very localized. The first 4 years of history of the OO model (up to 4/1999) has been documented in [1,4].

The original OO model was stand alone, as it predated the development of full reconstruction and analysis frameworks in both experiments. During 1999 CMS developed a powerful OO reconstruction framework known as ORCA (Object oriented Reconstruction for CMS Analysis). After all necessary functionality for input objects in ORCA became standardized in autumn of 1999, we integrated the OO tracker model into ORCA by inputting the ORCA reconstructed hits to the model and reconstructing tracks. ORCA is continuing to develop rapidly, and complete integration of the OO model will be done using the final ORCA detector and track classes. Also in 1999, the OO model has been implemented in the OO reference software framework of ATLAS level-2 trigger [5]. Again, the two implementations (into different OO environments) have shown a certain degree of similarity. As a consequence, any improvement in the code and any experience in the implementation to the general OO environment for one experiment immediately benefit other experiment.

In this report, the history of the OO model for track reconstruction and C++ coding up to 4/1999 are summarized in Section 2; the new developments of the OO model and its implementations in the CMS and ATLAS experiments are explained in Section 3; some preliminary performance results are shown in Section 4; the experience and lessons we have learned are list in Section 5; and a summary and future prospects are given in Section 6.

## 2. Review of the OO model and C++ coding

Following the trend in the High Energy Physics (HEP) community to gradually move from the procedure oriented to the Object Oriented Programming (OOP), we have tried to design an OO model by using well known HEP data concepts to form classes, then grouping the relevant services for those data into those classes (i.e. class abstraction). Meanwhile, we have also paid attention to implement some of the other main features of OOP (such as inheritance and polymorphism) in this model. During the model design stage, we found that some member functions in some classes have the corresponding FORTRAN subroutines in the already existing Kalman filtering track reconstruction package [6] of the then CMSIM (CMS simulation and reconstruction facility). Those subroutines are highly modular and do not have any common blocks interface with the outside world. We had successfully re-used those subroutines as member functions of various classes, which account for about 30% of total lines of code in the OO model and enable us to concentrate our effort on the OO and C++ aspects of model in the first 4 years of development. Until 3/1999, when the model was sufficiently mature, the highly modular FORTRAN code had been converted to C++ in a straightforward manner, with the aid of the UNIX utility "f2c". Since then, all further development in the model has been with the pure C++ version. Nevertheless, the temporary use of legacy FORTRAN in member functions of certain classes greatly increased the development speed of the initial prototype versions of the model.

The model design was started by formulating a "problem statement", which guides the design of the class diagram (Fig. 1) drawn here by the commercial product Rational/Rose using the Booch methodology [7]. A "function statement" serves as a guideline for drawing the object (Fig. 2) and the message trace diagrams. The details of the model and class description can be found in [1] and are downloadable from the Web [4].

The first few versions of the model were rather messy, partly due to the confusion engendered by mixing the implementation details of container classes with the class definitions. Since December 1995, we started to investigate the utilization of the Standard

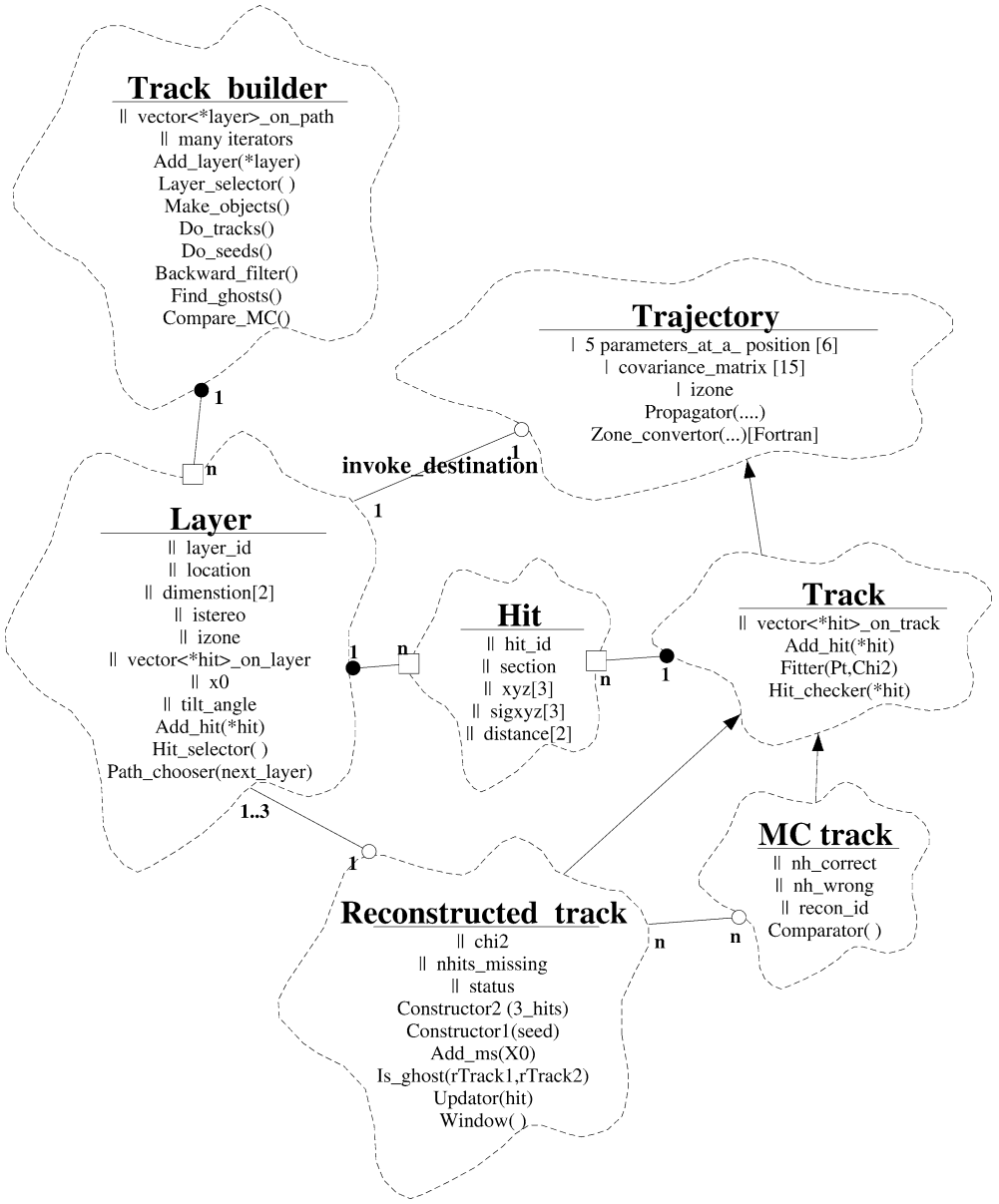


Fig. 1. Class diagram of the OO Tracker model.

Template Library (STL) [8]. It turned out that STL is indeed a powerful tool, not only for the C++ programming, but is also helpful for the OO model design. Some list-type classes representing vectors and linked lists (see Fig. 1 of [1]) in the first few versions of the OO model have now been absorbed into relevant classes by using STL containers, resulting in a much cleaner and simplified class diagram (Fig. 1).

In addition, the STL not only helped us to simplify the model design; it also helped us in the implementation of variable length arrays and linked lists in the main program. Various types of STL iterators were

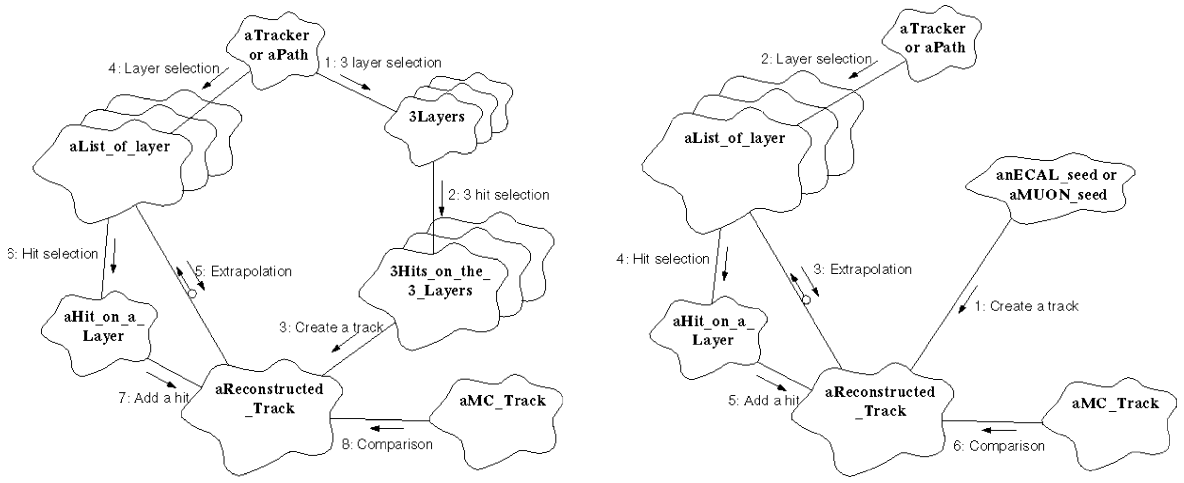


Fig. 2. Object diagram for scenario 1 (left) and scenario 2 (right).

particularly useful, as the STL iterator concept is well matched to looping through lists (vectors) of hits and track candidates. The flexibility and ease of use of STL proved a great asset to this OO model.

While the work evolved, the C++ code had become more and more complex since functionality was added to deal with various circumstances; the main program had grown to a length of a few hundred lines. After carefully examining the structure of the main program, we realized that the main program could be modularized as several separate pieces, and that each piece could form a member function of an abstract class. Then we re-named the "Tracker" class of Fig. 1a of the first article of Ref. [1] to "Track builder" class with those additional member functions. As a result, the main program for the OO model was reduced to a very terse one with merely dozens of essential lines. The successful modularization of those new member functions also made the debugging more effective by isolating bugs in different modules. This track builder class is perhaps unnatural to more traditional procedure oriented programmers (as opposed to more conventional track and hit classes that correspond directly to actual physical objects). Nevertheless, it proved an important design element in moving towards a truly object oriented model structure.

Since the end of 1997, the OO model has been ported into the environment of another major LHC experiment, ATLAS. Interestingly, the port did not involve merely the reuse of some of the classes originally designed for the CMS implementation. Rather, we found that the model, with a modular design enforced by use of the OO methodology, was able to be used almost in its entirety, with changes between the CMS and ATLAS implementations confined to a small number of well defined areas (e.g., the input and output interfaces).

The primary intention of this model in ATLAS is to reconstruct tracks in the SemiConductor Tracker (SCT, including the pixel detector) for the level-2 trigger software. During the low luminosity operation period of LHC the level-2 B-physics trigger will only have about milliseconds available for processing. Thus, the execution speed is a major factor of concern for any trigger track reconstruction model. Several measures have been taken to increase the execution speed in this OO model, e.g., to increase the hit selection efficiency by a "circular iterator", to reduce the number of initial triplet seeds used to instantiate candidate tracks, and to investigate various seeding methods.

## 3. Implementations of the OO model in CMS and ATLAS

The layouts of ATLAS and CMS inner trackers are shown in Figs. 3 and 4. Though the technical

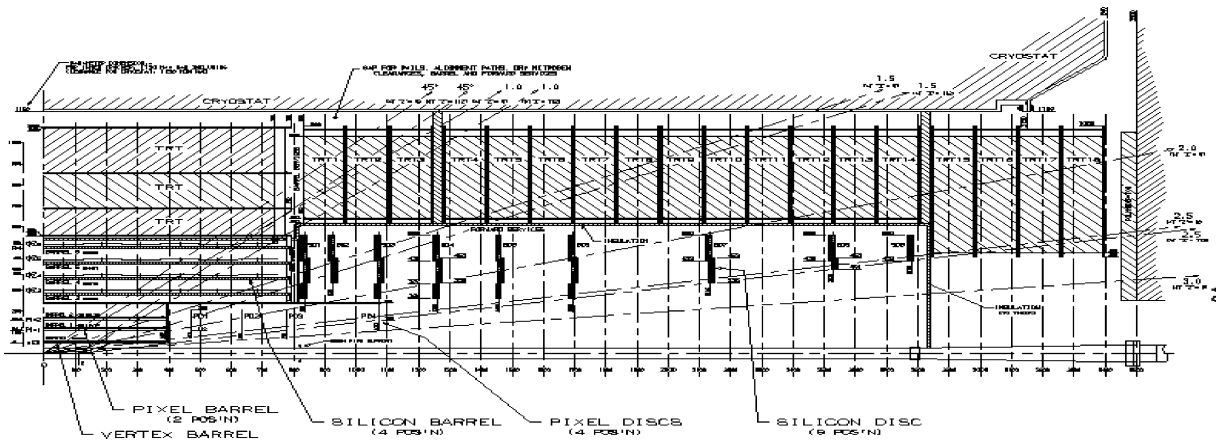


Fig. 3. Layout of the ATLAS Tracker.

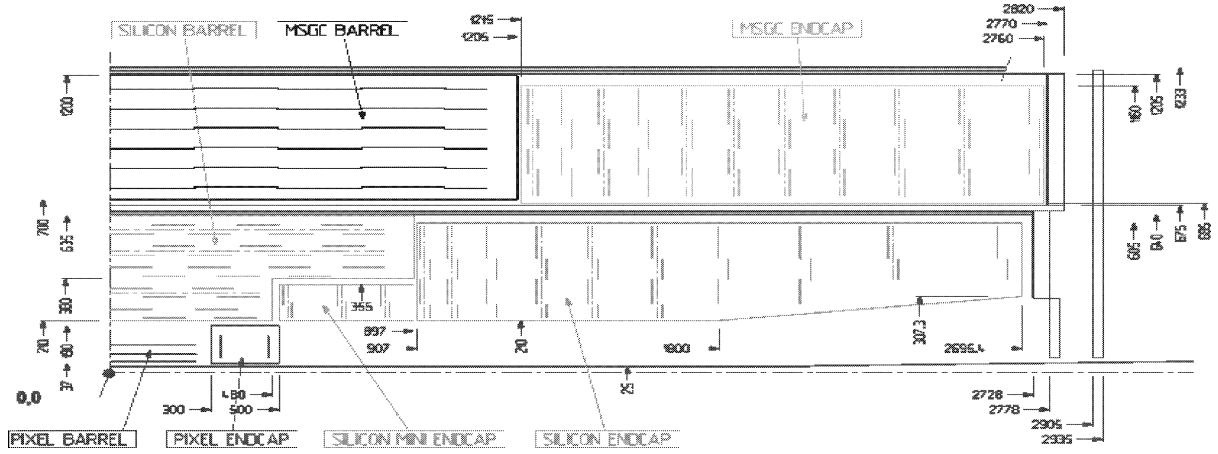


Fig. 4. Layout of the CMS Tracker (before it was changed to the full silicon version from the beginning of 2000).

details of the two trackers are quite different, they are essentially the same from point of view of the software implementation, i.e. they both consist of several layers of detector. This is one of the reasons that this OO model can be easily re-used in different experiments without major modification (i.e. by only instantiating different layer classes).

However, the implementation of the model into different experiments can be very different. This is mainly an I/O issue. For instance, in CMS, we need to read input hits from CMSIM output, then store the output back to CMSIM data system; whereas in ATLAS, we input hits by reading ASCII files from the ATLAS trigger simulation and store the reconstructed tracks into ATLAS database. For simplicity, at the early stage of the stand-alone OO model development, we temporarily dumped the CMSIM data into an ASCII file; then re-formatted it to feed the model. Now we are using the I/O scheme of ORCA. For the latest ATLAS implementation, we have to plug the model into the OO reference software framework of level-2 trigger [5] with yet another different I/O scheme.

Another major difference between the two implementations in CMS and ATLAS is due to the purpose of the application. In CMS, we intend to apply it in offline track reconstruction where the completeness,

accuracy and high efficiency are emphasized. Therefore, we use the first scenario (Fig. 2(a)) of this OO model's object diagram to search for tracks throughout the tracker without using outside seeds (although in principle, the outside seeds also can be used in future to find an initial set of tracks). Whereas in the ATLAS trigger application, the speed and efficiency are more important, so we use the Transition Radiation Tracker (TRT)'s output as the seeds and perform the track reconstruction within each Region of Interest (RoI) by invoking the second scenario (Fig. 2(b)). Since each event only has a limited number of RoI in the LHC low-luminosity operation period and since the most time-consuming part (i.e. triplet formation) of the model is eliminated, the OO model can perform closer to the speed requirement of level-2 B-physics trigger. Recently, another seeding method using the seeds produced by a pixel reconstruction package, and starting the Kalman filtering process from inner-most position outwards has been implemented. This results in a better performance of the OO model due to the preciseness of the pixel hits. For the 2nd scenario, the performance of the OO model depends critically on the quality of the seeds; therefore, more seeding methods will be tested in order to achieve the optimum performance.

Despite the above differences between the two experiments, we could put the I/O code in two encapsulated functions, then the implementation structures for both experiments are very similar.

## 4. Preliminary performance result

For the performance of the OO model, the ATLAS implementation has been tested extensively by using trigger simulation data as input. The standard data sets (for testing various algorithms) are single track ($\mu$'s, $\pi$'s and electrons at different energies) events and B-physics events at low luminosity. The computer used is a 300 MHz Pentium II under Linux. Here, we only show a few typical results about the efficiency (Figs. 5 and 6), momentum resolution (Fig. 7), energy loss correction for electrons (Fig. 8), execution speed (Table 1) and ghost reduction (Table 2). Fig. 9 is an example of B-physics study in the channel $B \to \pi\pi$. Many more results are described on Web site [9]. A very recent documentation "ATLAS High-level Triggers, DAQ and DCS Technical Proposal" [10] and one [11] of its backup notes include the most updated performance result by using the OO model.

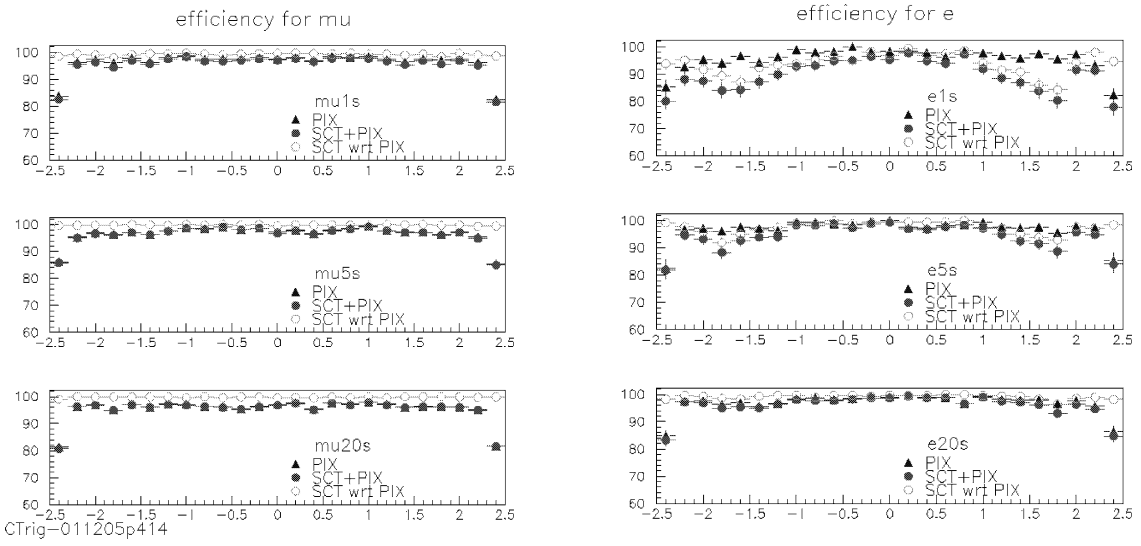


Fig. 5. Efficiencies vs. pseudo-rapidity for muons (left) and electrons (right) at different momentum of 1 (top), 5 (middle) and 20 (bottom) GeV/c.

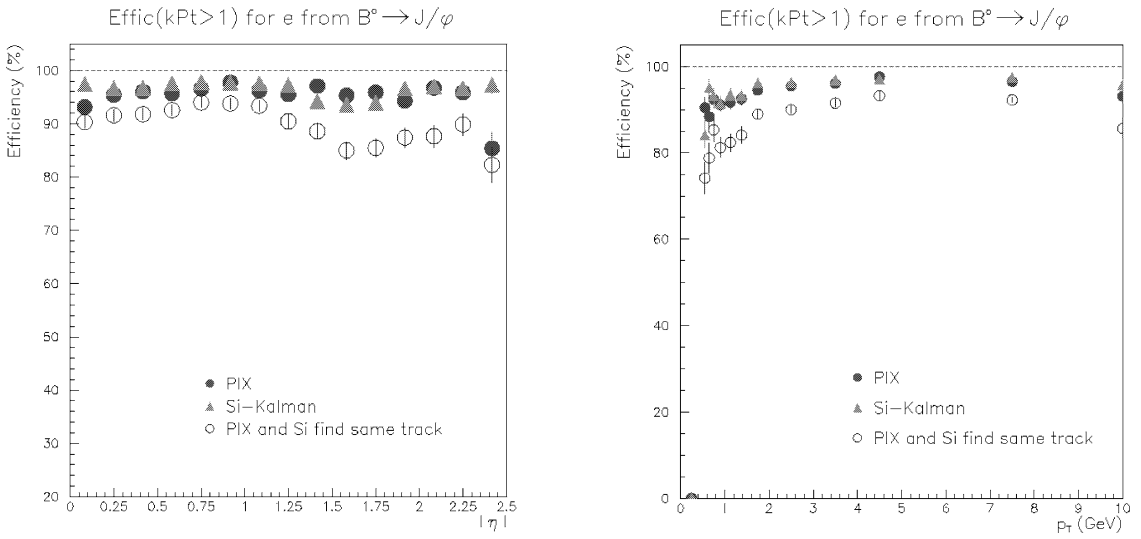


Fig. 6. Efficiencies vs. pseudo-rapidity (left) and vs. transverse momentum (right) for B-physics events.

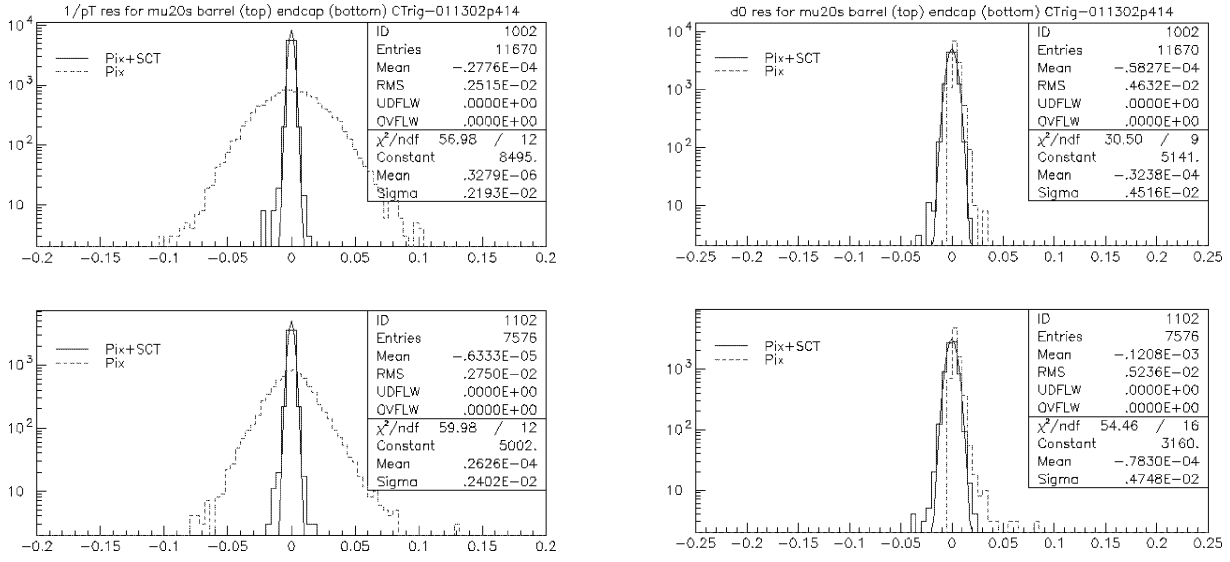


Fig. 7. Resolutions of 1/pT (left) and impact parameter d0 (right) for 20 GeV muons.

Table 1
Statistics of the benchmarking measurement

| Data set | Execution time per event (millisecond) | | | TRT seeds per event | | | Average execution time per TRT seed |
|---|---|---|---|---|---|---|---|
| | min. | max. | average | min. | max. | average | (millisecond) |
| Electron without pile-up | 1.2 | 5.6 | ∼⟨1.9⟩ | 1 | 3 | ∼⟨1.55⟩ | ∼⟨1.8⟩ |
| Dijet without pile-up | 0.8 | 109 | ∼⟨5.3⟩ | 1 | 14 | ∼⟨2.21⟩ | ∼⟨2.6⟩ |
| B-physics at low luminosity | 7.6 | 3544 | ∼⟨172⟩ | 2 | 493 | ∼⟨53.3⟩ | ∼⟨2.6⟩ |

Table 2
Statistical results for the standard data sets (for barrel region only, i.e. $|\eta| < 0.65$)

| Data set | Total number of events | Extra TRT tracks | Extra SCT/Pixel tracks* |
|---|---|---|---|
| 5 GeV $\mu$ | 4737 | 111 (2.3%) | 0 |
| 1 GeV $\mu$ | 4770 | 56 (1.2%) | 0 |
| 5 GeV e | 526 | 167 (32%) | 11 (2.1%) |
| 1 GeV e | 1293 | 324 (25%) | 1 (<0.1%) |
| 5 GeV $\pi$ | 4331 | 1254 (29%) | 86 (2.0%) |
| 1 GeV $\pi$ | 4720 | 429 (9.1%) | 6 (0.13%) |

* The right-most column is the result of this OO model applied in ATLAS inner SCT/Pixel tracker.

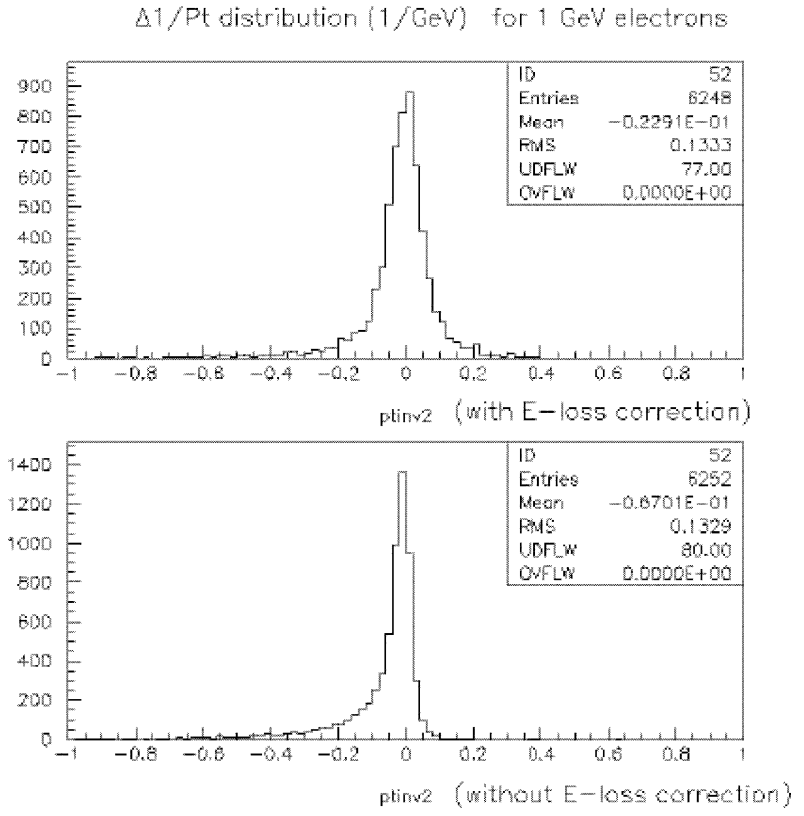


Fig. 8. Energy loss correction for electrons.

The memory usage of the model is in the order of 10 MByte, which also depends on the data volume of the event, with high luminosity events occupying more memory than the low luminosity ones.

## 5. Experience and lessons learned with this OO model

During the practice of OO analysis, OO design and C++ coding stages as well as several cycles of redesign in last 5 years, we feel that

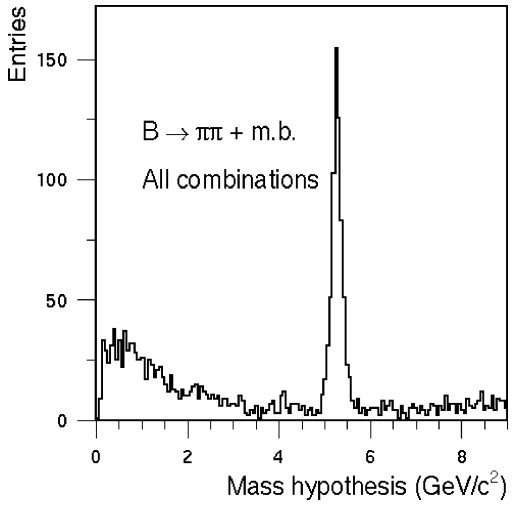


Fig. 9. Reconstruction of the B $\rightarrow \pi\pi$ mass hypothesis in the ATLAS Level-2 trigger by using the Kalman filtering method.

(a) The transition from procedure oriented programming to OOP is quite natural and straightforward, partially since the syntax of FORTRAN and C++ have some similarity and correspondence.
(b) The user-defined data types (to form the classes) are extremely powerful, encapsulating the behaviour of classes makes it easy to use other classes without interfering with each other.
(c) STL provides powerful tools well matched to HEP needs (e.g., iterators, sorting algorithms, etc.).

C++ has some new features (which were unfamiliar to us as previous FORTRAN programmers and somehow frustrated us), e.g., pointers, scoping rules, memory leaks, etc. But with some powerful debugging software tools that have gradually become available during the last few years, the new features can now be handled. For example, the memory leak problem now can be detected by INSURE++ (a product of ParaSoft); the bugs related with pointers can be detected by a graphic debugger DDD (Data Display Debugger, a product of Technical University of Braunschweig).

Regarding to the integration of the OO model into a general OO environment of the whole experiment, we feel that it is essential for the successful integration by having a stable general framework with a stable interface. As soon as all functionality of I/O objects needed by the OO model are available, the integration is very straightforward.

In summary, our experience has been:

- Object Orientation makes it easier to do cooperative development among widely separated collaborators, and promotes truly modular designs.
- Object Orientation makes re-use easier, i.e. the design and code can be shared between different experiments and have each almost immediate benefit from development in the other experiment.

## 6. Summary and prospects

We have designed an object oriented model for track reconstruction in HEP experiments, coded it in the C++ programming language and preliminarily implemented it into both the CMS and ATLAS experiments on LHC. The main features of this model are:

- Class design is according to the OO paradigm and is based on the proven data concepts in HEP track reconstruction, so that hopefully it can be easily adopted by the non-expert class users.
- The OO model is closely related to the Kalman filtering track reconstruction package of the previous pure FORTRAN version. Many FORTRAN subroutines of the package have been originally re-used as member functions of various classes, and later converted into C++ in a straightforward manner.
- The STL, a powerful tool in C++ programming, has been extensively used in the OO model design and the C++ coding.
- The OO model is flexible enough to be re-used in multiple HEP experiments, with only the implementation of layer class different.

The preliminary results show that its memory usage is moderate, its track finding efficiency is satisfactory and its execution speed is approaching the requirement of level-2 trigger. We now have some powerful C++ debugging tools on hand, which will be very helpful for future development.

Next, for the CMS implementation, we will complete its integration into ORCA by investigating more efficient I/O functions, use the standard CMS classes, and then tune the performance. For ATLAS, we can improve the performance by investigating new seeding methods and exploring new reconstruction strategies. More sophisticated corrections (e.g., non-uniform magnetic field correction, etc.) can be grad-

ually considered after a recently implemented energy loss correction for electrons.

More details of this OO model and its implementation (including the model design, the class and object diagrams, the performance results, the documentation and the presentations in various conferences and meetings, etc.) can be found on the Web site [4] which is updated regularly to include all new developments.

## Acknowledgements

We are very grateful to Drs. S. Wynhoff and T. Todorov for their assistance on the integration of the OO model into ORCA, and to Drs. M. Sessler and J. Baines for their great help on the ATLAS implementation of the model into the level-2 B-physics trigger software.

## References

[1] I. Gaines, P. LeBrun, S. Qian, Design and test of an OO model for CMS track reconstruction, CMS TN/96-122, 1996; http://cmsdoc.cern.ch/documents/96/tn96_122.pdf; I. Gaines, T. Huehn, S. Qian, Design and performance of an OO model for CMS track reconstruction, in: Proceedings of Computing in High Energy Physics (CHEP'97), Berlin, Germany, April 1997; also filed as CMS CR/1997-018, 1997; http://cmsdoc.cern.ch/documents/97/cr97_018.pdf; I. Gaines, S. Qian, Design and re-use of an OO track reconstruction package in multiple LHC experiments, in: Proceedings of Computing in High Energy Physics (CHEP'98), Chicago, USA, September 1998; also filed as CMS CR/1998-023, 1998; http://cmsdoc.cern.ch/documents/98/cr98_023.pdf; I. Gaines, S. Qian, An OO model for track reconstruction in multiple LHC experiments, in: Proceedings of 6th Advanced Computing Conference in Physics (AIHENP'99), Crete, Greece, April 1999.

[2] CMS Technical Proposal, CERN/LHCC/94-38, 1994.

[3] ATLAS Technical Proposal, CERN/LHCC/94-43, 1994.

[4] http://cmsdoc.cern.ch/∼sijin/oo.html; http://www-wisconsin.cern.ch/∼sijinat/oo.html.

[5] S. Qian, M. Sessler, Implementation of Kalman filtering algorithm in the SCT FEX system for the object-oriented LVL2 reference software, ATLAS testbed note 026, 1998; http://www.cern.ch/ATLAS/project/LVL2testbed/www/notes/026/tn026.ps.

[6] S. Qian, Simultaneous pattern recognition and track fitting by Kalman filtering method for CMS inner tracker, CMS TN/96-001, 1996; http://cmsdoc.cern.ch/documents/96/tn96_001.pdf.

[7] G. Booch, Object-Oriented Analysis and Design, Benjamin-Cummings, 1994.

[8] D.R. Mosser, A. Scini, STL Tutorial and Reference Guide, Addison-Wesley, Reading, MA, 1996.

[9] http://www-wisconsin.cern.ch/∼sijinat/lvl2/res.html; http://www-wisconsin.cern.ch/∼atsaul/results/sctkalman; http://hepunx.rl.ac.uk/atlasuk/simulation/level2/Bphys/plots.html.

[10] ATLAS HLT/DAQ/DCS Group, ATLAS high-level triggers, DAQ and DCS technical proposal, CERN/LHCC/2000-17, April 2000; http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/SG/TP/tp_doc.html.

[11] J. Baines et al., B-physics event selection for the ATLAS high level trigger, ATLAS-DAQ-2000-031, June 2000; http://documents.cern.ch/archive/electronic/cern/others/atlnot/Note/daq/daq_2000_031.pdf.