




# ILD Tracking Software for the DBD

Steve Aplin

LCWS10 Beijing  
29<sup>th</sup> March 2010

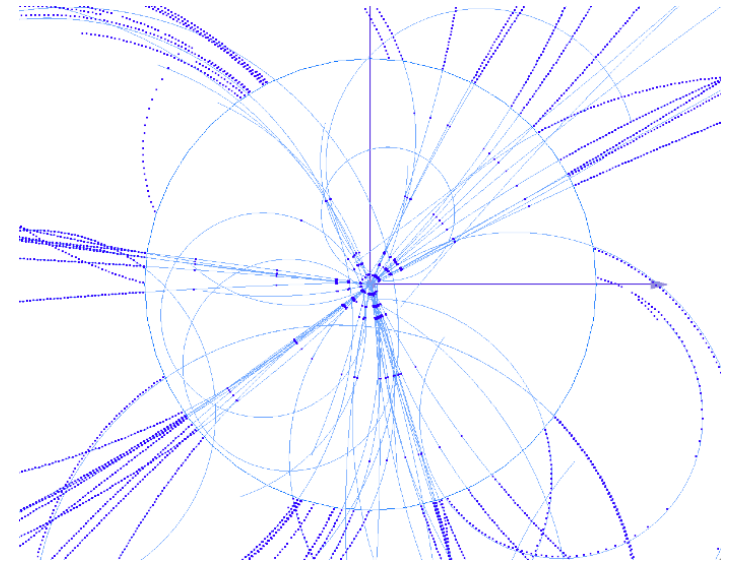


# Overview

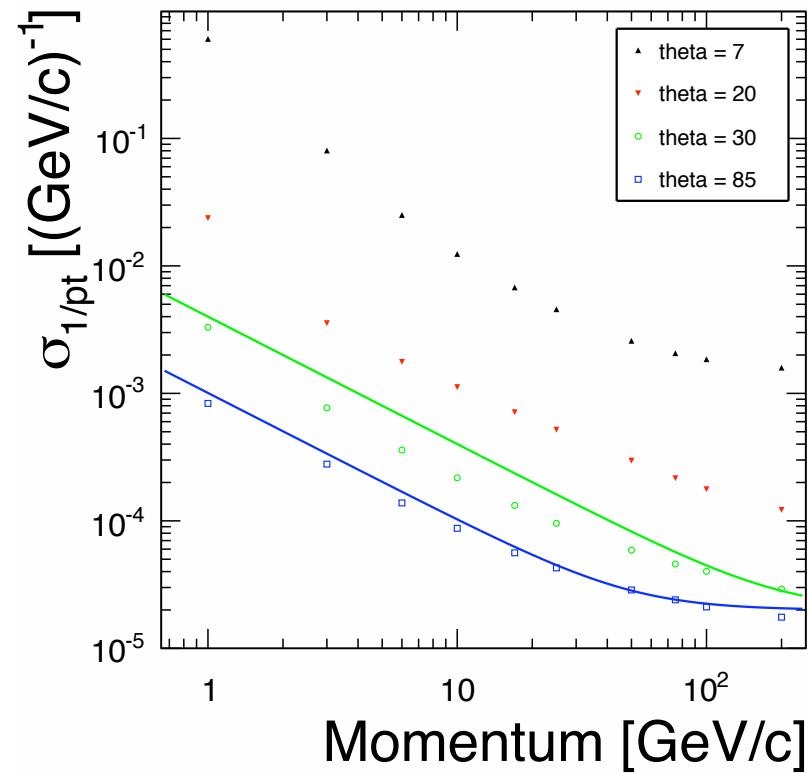
- Present Status
- Requirements for the DBD
- Current Activities
- Plans for Moving Ahead
- Technical Requirements

# ILD Tracking Software

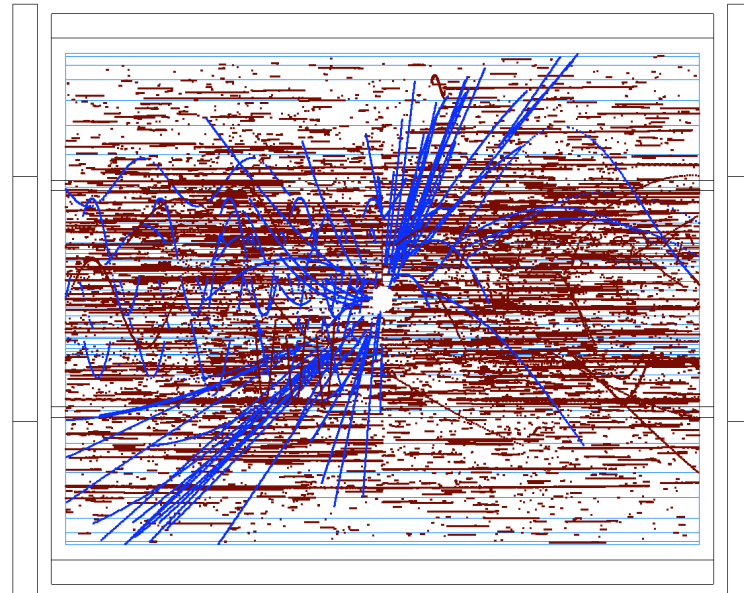
- Full Pattern Recognition
  - both Local and Global
  - kinks, V0's, background rejection
- Kalman Filter used for Fitting
- Well Validated for the Lol Studies



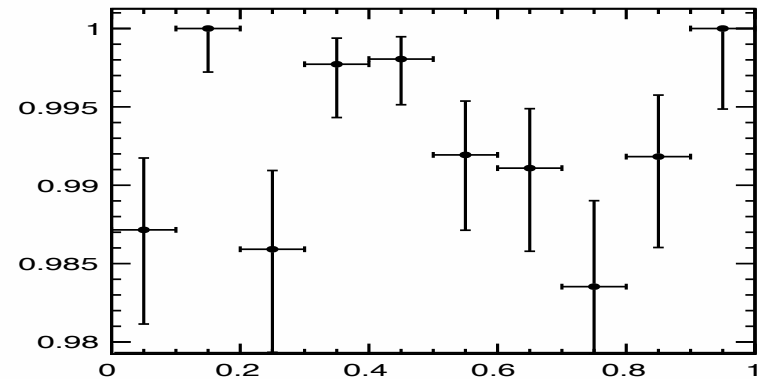
# ILD Tracking Software



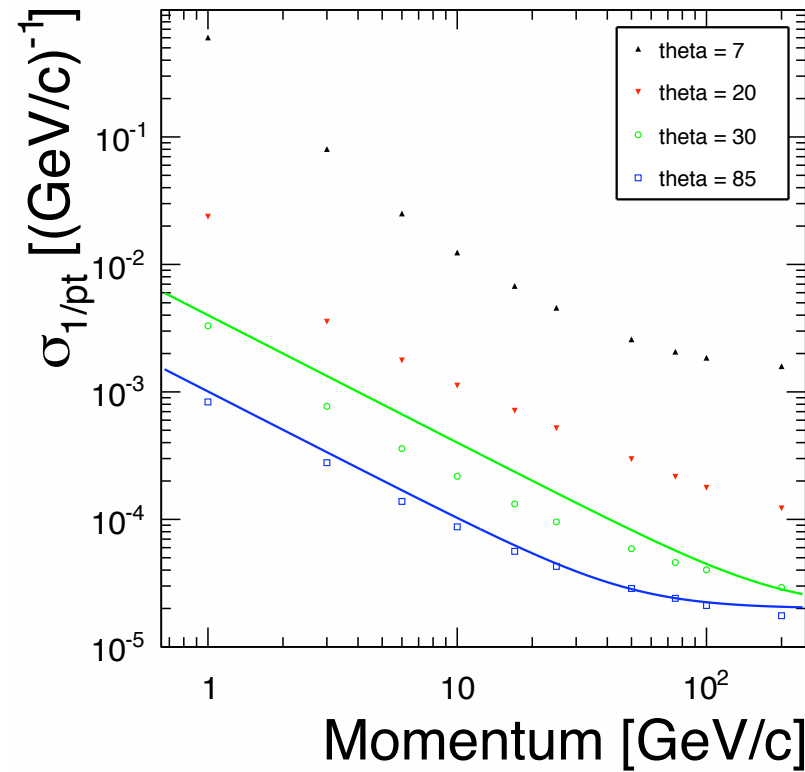
—  $\sigma_{1/pt} = 2 \times 10^{-5} \oplus 1 \times 10^{-3} / (pt \sin \theta)$



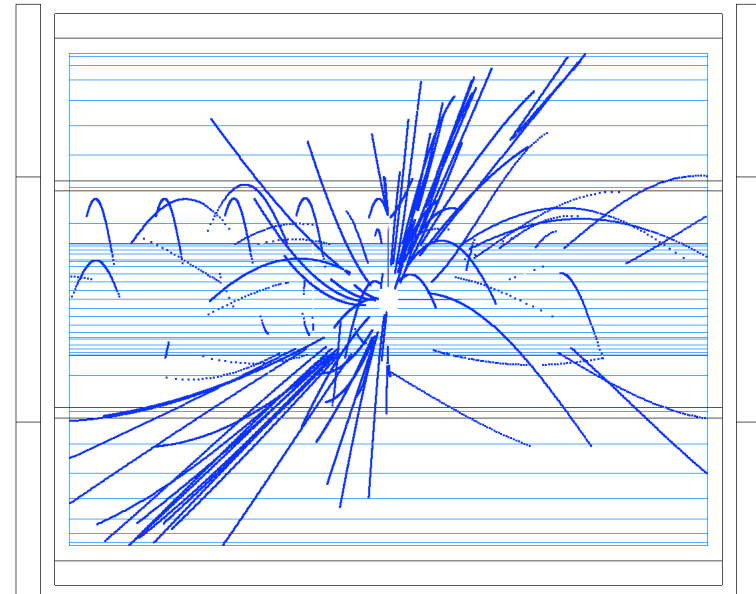
TPC Tracking Efficiency (Good Tracks) vs Cos $\theta$  ( $p > 1\text{GeV}$  and  $N_{\text{Hits}} > 30$ )



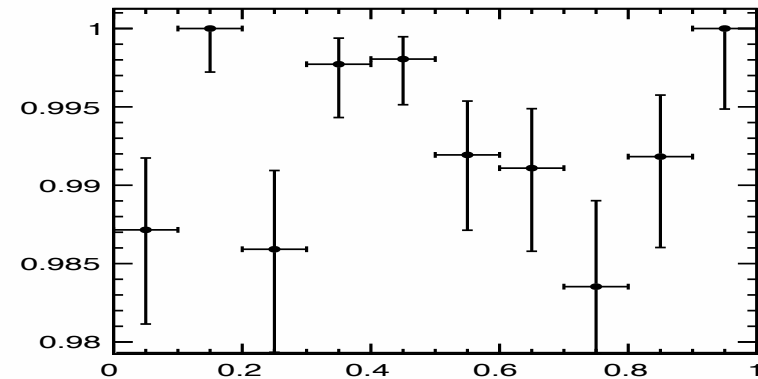
# ILD Tracking Software



—  $\sigma_{1/pt} = 2 \times 10^{-5} \oplus 1 \times 10^{-3} / (pt \sin \theta)$



TPC Tracking Efficiency (Good Tracks) vs Cos $\theta$  ( $p > 1\text{GeV}$  and  $N_{\text{Hits}} > 30$ )



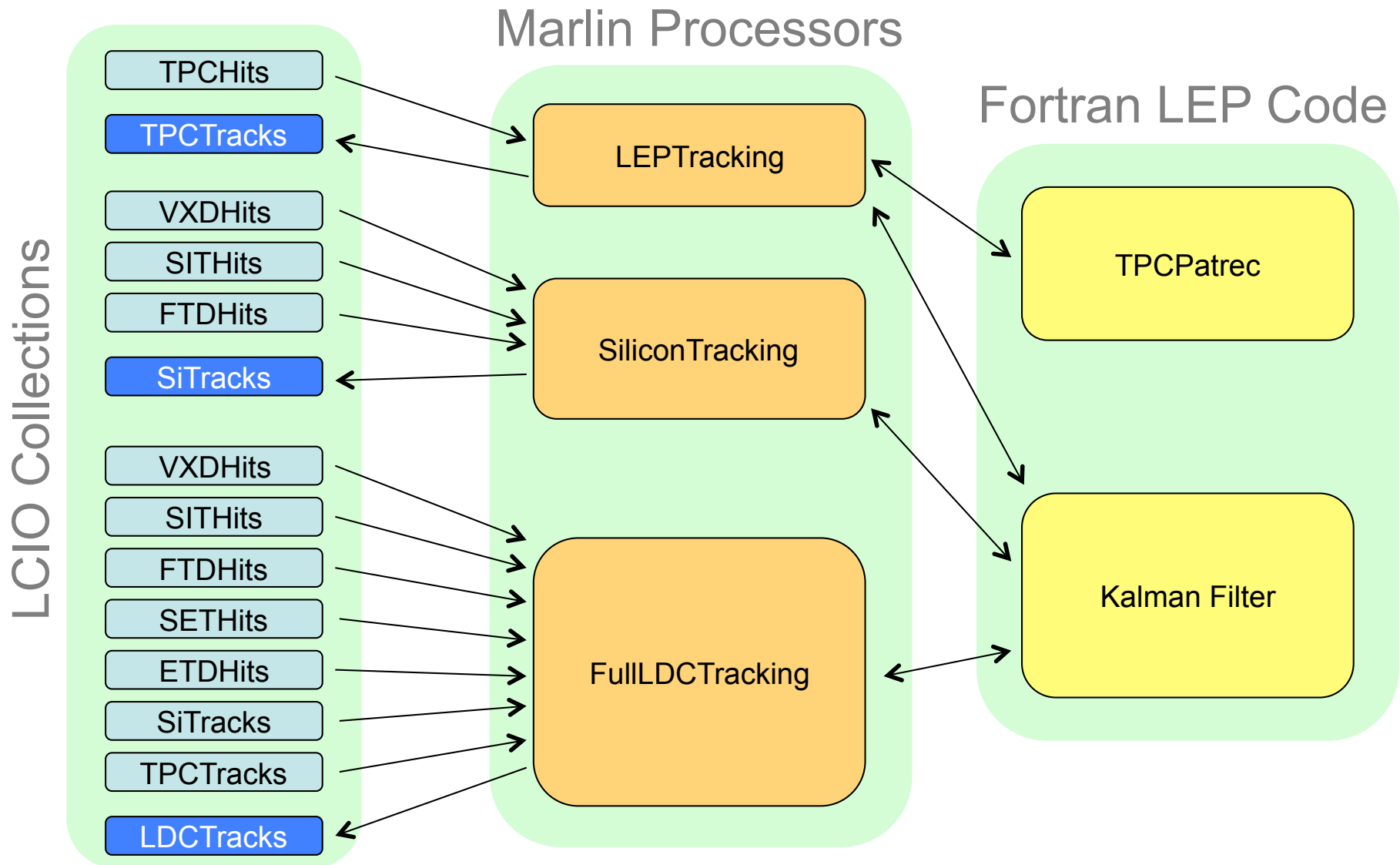
# Outstanding Issues

- The description and treatment of strip detectors
- Pattern recognition in the silicon trackers, esp. FTD
- TPC Pattern recognition is based on single BX's
  
- Code is generally messy and hard to maintain
- The Fortran code makes modification difficult, as well as presenting memory issues, esp. w.r.t. background studies

# Current use of Fortran and C++

- All Track fitting done is done in Fortran
- TPC Pattern recognition is done in Fortran
- VXD+SIT+FTD Pattern recognition is done in C++
- Final Track Building + Extra Hits + SET + ETD – C++
- Final Track fitting is done in Fortran

# Current use of Fortran and C++





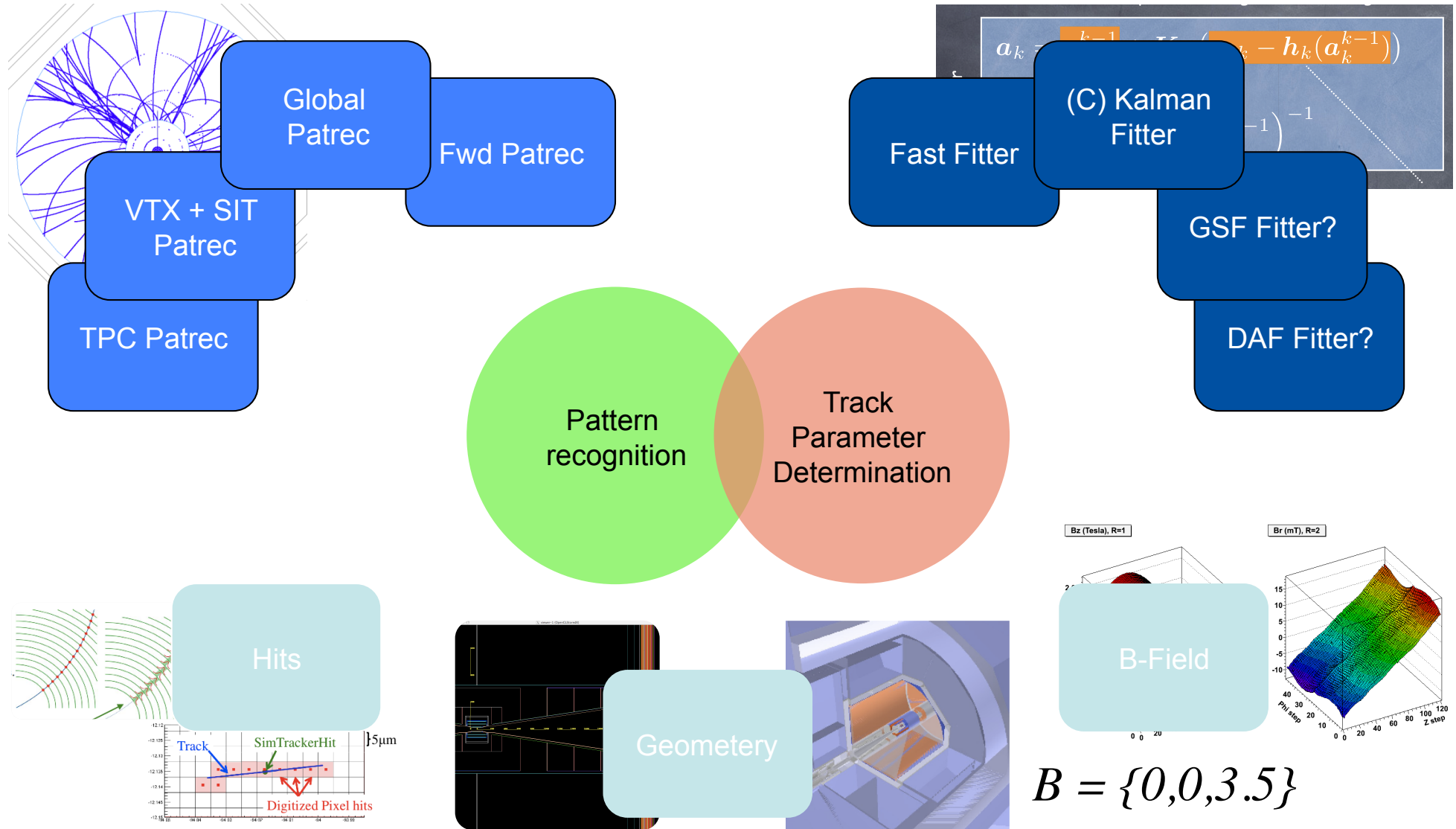
# Currently Discussed for the DBD

- More Realism
  - Material
  - Digitisation
  - Non Uniform Magnetic Field
  - Background
  - Alignment

# Requirements for a replacement

- Tracking Software needs to be used for mass production, this means being able to reconstruct millions of challenging events, whilst maintaining an efficiency of better than 99%, including up 1TeV
- Use more realistic measurements, i.e. move away from 3D space points and towards measurements on surfaces, e.g. pixels, strips.
- Patrec: able to survive the high background levels
- Fitting: able to cope with a non-uniform magnetic field

# Requirements for a replacement



# Use of Atlas Tracking Code

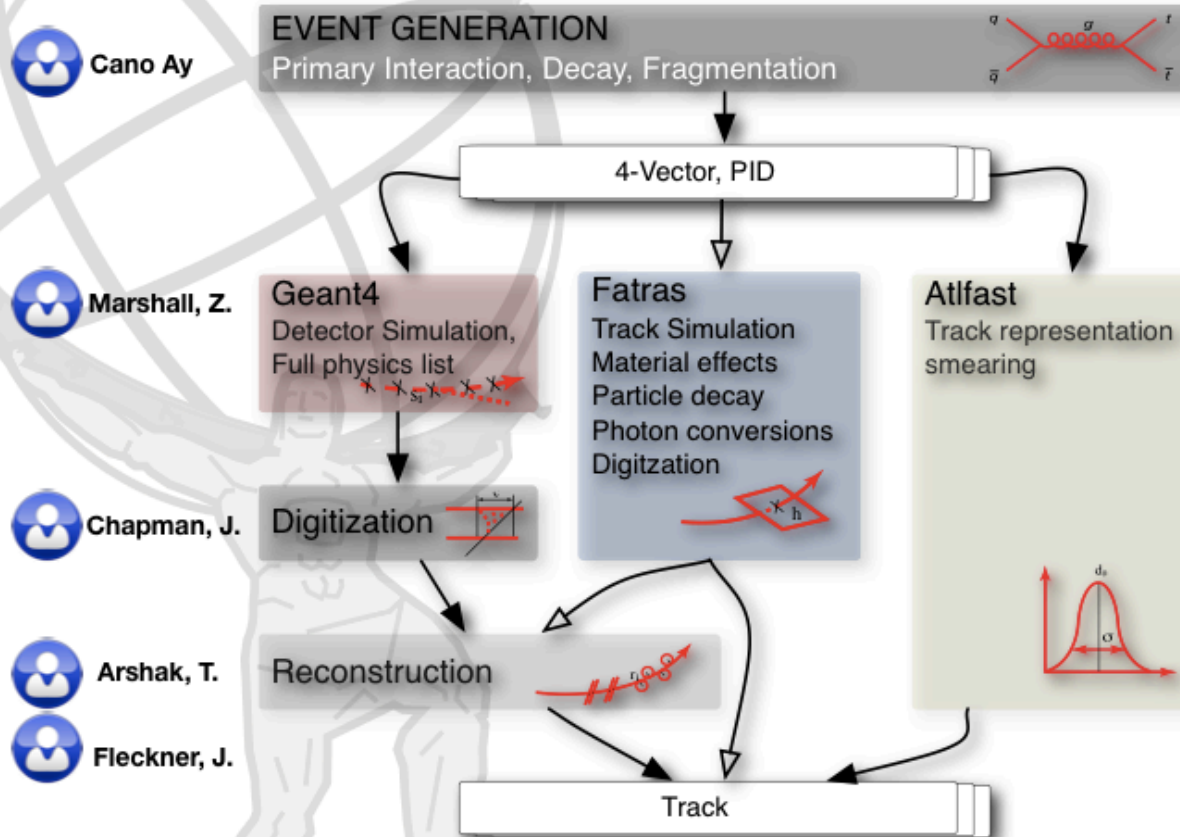
- Atlas Tracking Code contains a well defined Tracking and Geometry EDM, which are used by a set of Track Fitting Interfaces, with implementations of a Kalman Filter, CKF, DAF, together with a geometry navigator.
- We spoke to Atlas Tracking Authors at CHEP 09 where they initially expressed an interest in seeing their code reused.
- To evaluate the code the idea was to try to build a prototype of the Atlas tracking code for a TPC, outside of the Gaudi and Athena frameworks, ideally in Marlin

# Use of Atlas Tracking Code

- This was initially with the aim of not significantly modifying the codebase, e.g. use simplified header files, typedefs and #define statement wherever possible.
- The Code is well structured, well written and also very well documented.
- Unfortunately for us the dependence on Gaudi and Athena appears to be quite involved.
- Decided to change strategy and try to evaluate the code in situ, using the Atlas fast simulation tool FATRAS.

# FATRAS

## FATRAS - a fast full simulation (2)



A. Salzburger - ATLAS Upgrade Simulation using FATRAS - CHEP2009

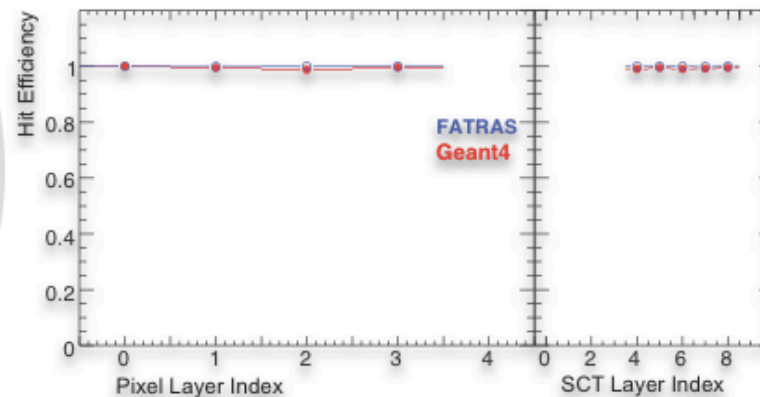
8

# FATRAS

## FATRAS - comparison with full simulation (2)

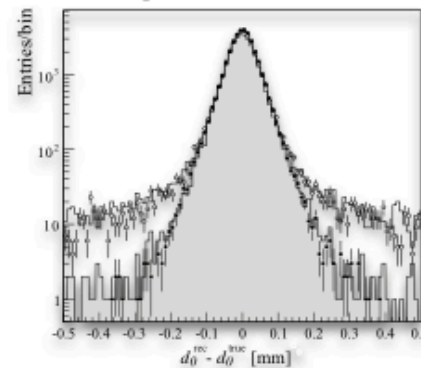
### ▶ Track reconstruction

- compatible results for FATRAS/Geant4
- strategy: use FATRAS simulation for reconstruction tuning



### ▶ Accomplished by extensive comparison on ATLAS layout

- track resolutions (plot: 5 GeV  $\mu$ /pion sample)
- reconstruction efficiencies

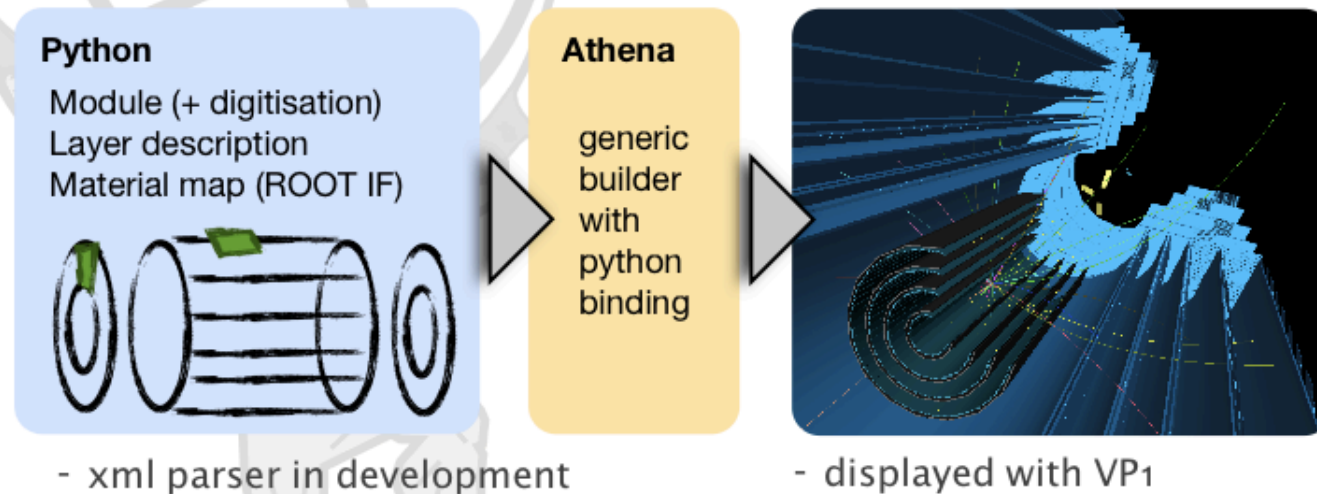


	FATRAS	offline/NEWT
Muons	†	†
Entries	33814	33707
RMS	0.0446 mm	0.0469 mm
Pions	†	†
Entries	37146	34539
RMS	0.070 mm	0.076 mm

# FATRAS

## FATRAS - custom geometry building

- ▶ **Reconstruction geometry is very flexible**
  - only set of cylinders and discs with sensitive modules attached
- ▶ **FATRAS provides a python interface for custom geometry**

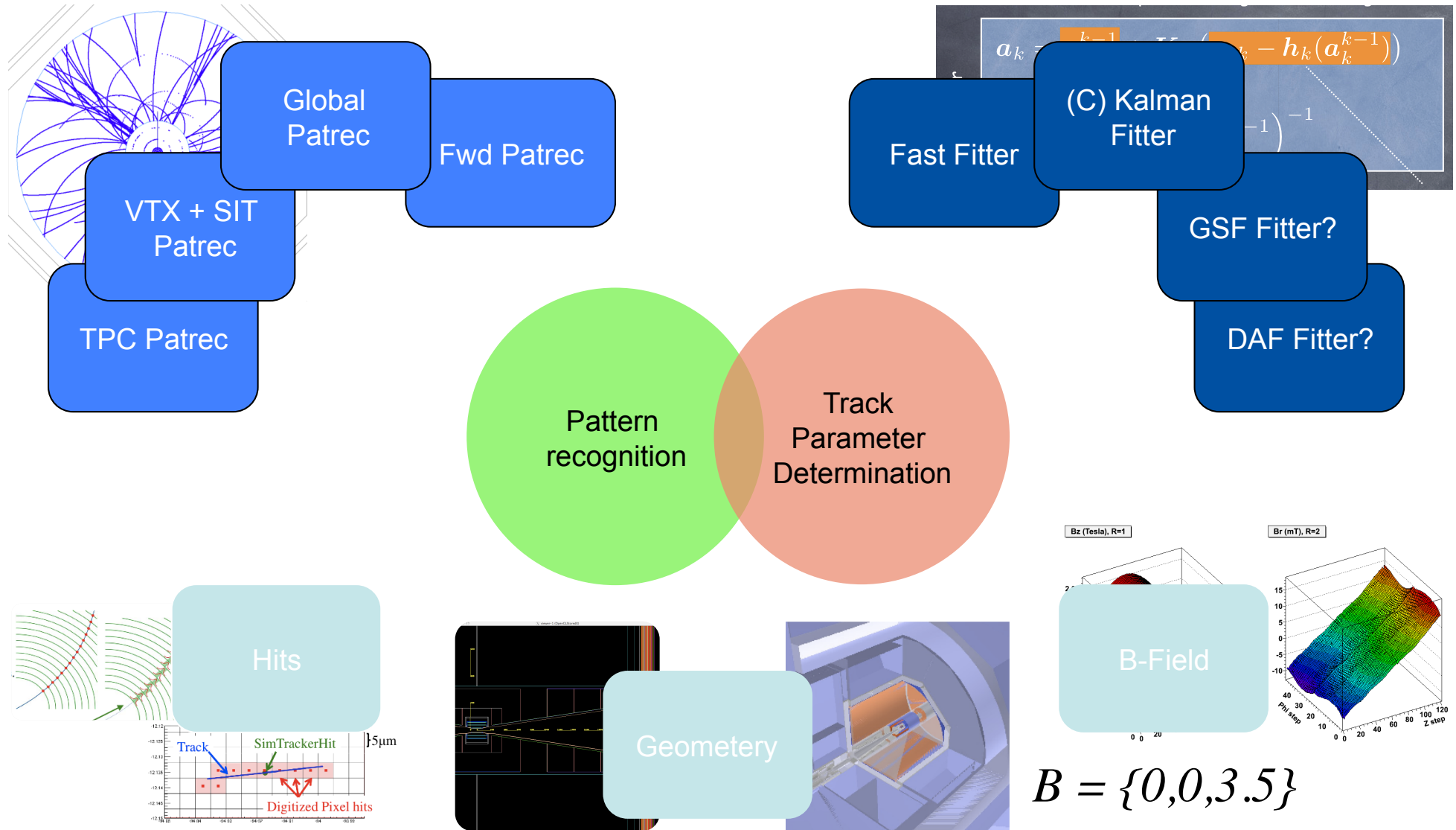




# Use of Atlas Tracking Code

- Our conclusion on porting the Atlas Tracking code for use in Marlin is that, without significant input from the authors, it is not feasible to use it for the DBD studies.
- The flexibility offered by FATRAS may provide us with a useful tool for evaluating the impact of more advanced features, e.g. GSF, DAF.
  - Presently in contact with the lead author of FATRAS to see what modifications are necessary to enable us to use it efficiently

# Towards a Replacement



# Towards a Replacement Fitter

- KalTest
  - ROOT based Kalman filter C++ library
  - Being brought into MarlinTPC by the LCTPC Group
  - Currently implementing a Combinatorial Kalman Filter
- Genfit
  - Generic track fitting package written for the Panda Experiment
  - ROOT based C++
  - Currently being introduced into Marlin for Belle II by A. Moll
- Broken Lines Fitting Algorithm
  - written by V. Blobel at DESY
  - C. Kleinwort is currently working on bringing this to C++ for CMS, and hopes to then be able to use it for LCTPC

# Technical Improvements Needed

- Provide a convenient method of calling a fitter.
- Update LCIO Track class or provide additional Trajectory class.
- Improve LCIO description of hits to include more than just 3D space-points.
- Improve GEAR description of Silicon Trackers.  
(Ongoing for SiLC by A.Charpy)
- Improve the way that the material description required for tracking is handled in Marlin.

# Summary

- ILD tracking software remains available and well suited to performing physics and optimisation studies.
- The search for a replacement fitter proceeding well.
- Material description and hit treatment improving.
- Pattern Recognition will require the greatest effort, especially in the presence of background.
- Within the AIDA EU Grant proposal, Tracking has been specified as a major part of the Software Work-package.