Summary of the first Common Track Reconstruction Software Forum

Johannes Rauch

Physik Department E18
Technische Universität München
Germany

December 4, 2015

The HEP Software Foundation

`hepsoftwarefoundation.org`
Resources, events, projects, etc.

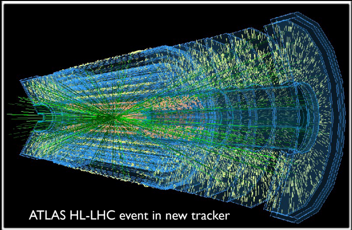
Introduction: Goals of the Common Tracking Software Forum

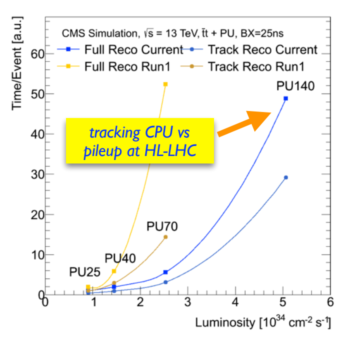
Frank Gaede, Benedikt Hegner, Markus Elsing

An introduction to creating a "forum" across experiments to discuss and promote the development of tracking software

The Experiments' Software Challenges

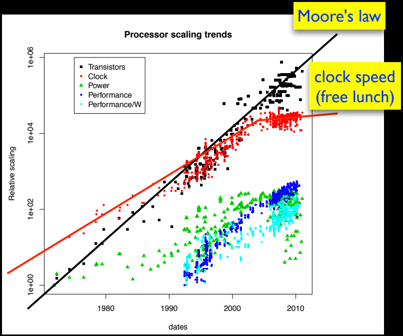
- ATLAS/CMS - million dollar question:
  - ➡ how to reconstruct HL-LHC events with 200 pileup
  - ➡ how to keep the physics performance up
  - ➡ and do it within the computing resources we'll have...
- tracking is reconstruction CPU driver
  - ➡ not new, we knew this would be the problem
  - ➡ will aim to improve on already highly optimised code
- LHCb and ALICE trigger-less readout
  - ➡ processing/filtering done in online trigger farms
  - ➡ offline quality reconstruction online to achieve needed data reductions
- Belle-II is about to start data taking
  - ➡ raw data volumes comparable to LHC
- Future Collider studies (ILC, CLIC, FCC)



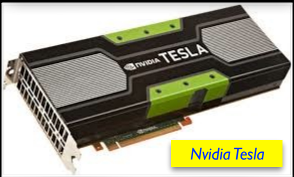


Technology Challenges

- Moore's law is still alive
  - ➡ number of transistors still doubles every 2 years
    - no free lunch, clock speed no longer increasing
  - ➡ lots of transistors looking for something to do:
    - vector registers
    - out of order execution
    - hyper threading
    - multiple cores
  - ➡ many-core processors, including GPGPUs
    - lots of cores with less memory
  - ➡ increase theoretical performance of processors



see G.Stewart, CHEP 2015

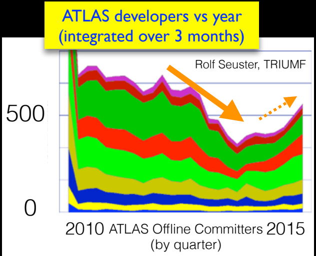


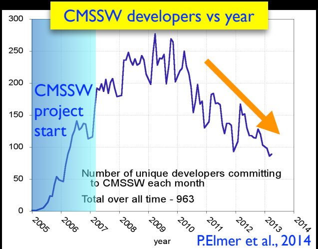
- challenge will be to adapt HEP software
  - ➡ hard to exploit theoretical processor performance
    - many of our algorithm strategies are sequential
  - ➡ need to parallelise applications (multi-threading)

    (GAUDI-HIVE and CMSSW multi-threading a step in this direction)
    - change memory model for objects, more vectorisation, ...

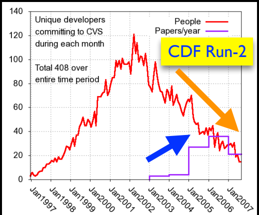
Software and Manpower

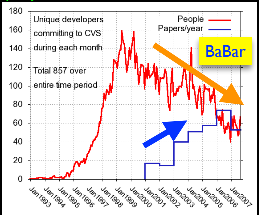
- software follows a natural life cycle
  - ➡ building up the software for an experiment
  - ➡ start of experiment operations and data taking
  - ➡ data analysis and detector upgrades
- loss of software manpower in ATLAS/CMS
  - ➡ (mostly) students and postdocs moved on to do physics
    - same trend like in previous experiments
  - ➡ like CDF/D0 Run-2, LHC upgrade program is ambitious
    - need to find sufficient manpower to develop the software for the upgrade (some positive trend in ATLAS)





P.Elmer, L.Sexton-Kennedy, C.Jones, ICHEP 2007

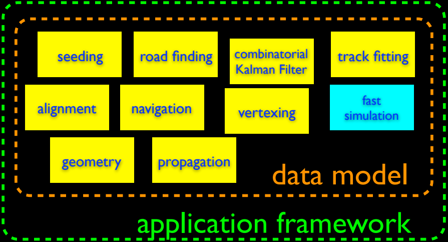




Common Tracking Software ?

- examples for common tracking software
  - ➡ AIDA tracking - primarily targeting ILC / FCC
  - ➡ GenFit - an implementation of standard track fitting techniques (Belle-II)
  - ➡ Millepede - track alignment page
  - ➡ CMS vertexing suite - package of standard vertexing codes (CMS, Belle-II,...)
  - ➡ VDT, SMatrix, Eigen - vector algebra and math libs
- current attempts for a common tracking implementation
  - ➡ AIDA is building one common solution
  - ➡ plan of ATLAS tracking group
    - make tracking/vertexing/fastsim suite public for FCC, builds on Gaudi/Athena
  - ➡ GenFit is aiming at a common solution
- are there obstacles ?
  - ➡ experiments already have a solution
  - ➡ integration means picking a data model
    - determines Jacobians in math formulars
  - ➡ integration means framework interfaces
  - ➡ best physics performance ?
    - pattern strategy depends on experiment

from Graeme Stewart



Why a "Tracking Software Forum" ?

- some observations:
  - ➡ major workshop like Connecting the Dots is filling a hole
    - not many other tracking oriented conferences and workshops
  - ➡ we probably should think about schools on tracking and reconstruction
    - we need to invest in future experts (and give them career perspectives)
- we felt we as well need a more regular forum to discuss developments in tracking software in HEP
  - ➡ complement major (yearly) workshops like CTD and technical Concurrency Forum
  - ➡ enable exchange of software ideas and concepts, share best practices
  - ➡ a by-product of this forum may be increased code re-usage
    - at last CTD not much enthusiasm expressed across all experiments (but FCC) to migrate to something like a common tracking software stack
    - but common software projects may grow naturally out of needs we may identify and we do have some common developments alreadystart by offering a common
  - ➡ provide repository, build system, etc., as an offer to the community to host candidate package that people like to share (see talk from Benedikt)

Linear Collider Track Reconstruction Tools

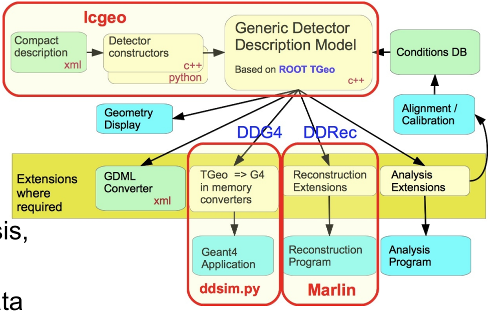
Frank Gaede, DESY
Common Track Reconstruction Software Forum
CERN, 2. Dec. 2015

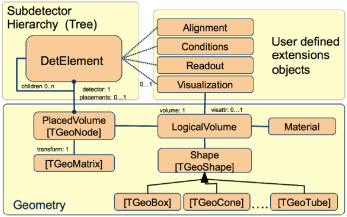
Introduction

- developed new C++ tracking tools for Linear Colliders
  - to replace old F77 (!) code from LEP
  - successfully used for ILD DB (2011), CLIC CDR (2012)
- partly done in context of AIDA-WP2 project
  - goal: eventually have a generic HEP tracking toolkit that could be shared by all LC detector concept groups (and possibly others)
  - allowing to transparently use different fitting algorithms
  - provide toolkit for pattern recognition
  - have well defined and easy to use interface to detector geometry
- code developed in context of ILD w/ generality in mind

DD4hep - DDRec for Tracking

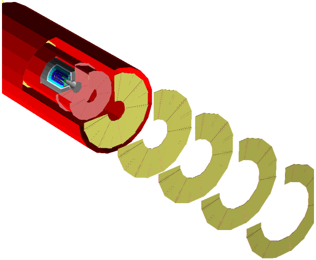
- **DD4hep**: detector geometry description for HEP
  - AIDA project (CERN/DESY)
- support full experiment life cycle
- one source of geometry for
  - simulation, reconstruction, analysis, event displays,...
  - extension mechanism for user data → **DDRec**
  - simple detector description classes: extend, layout, #layers,..
  - cellID ↔ position
  - material properties (point, line)
  - tracking Surfaces

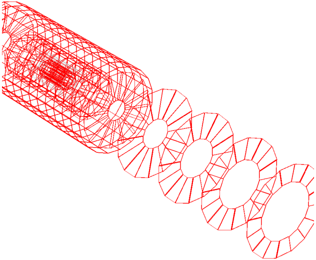




DDRec surfaces for tracking

- tracking needs special interface to geometry
- measurement and dead material surfaces (planar, cylindrical, conical)
- surfaces attached to volumes in detailed geometry model





- u,v, origin and normal
- inner and outer thicknesses and material properties
- local to global and global to local coordinate transforms:
- (x,y,z) ↔ (u,v)

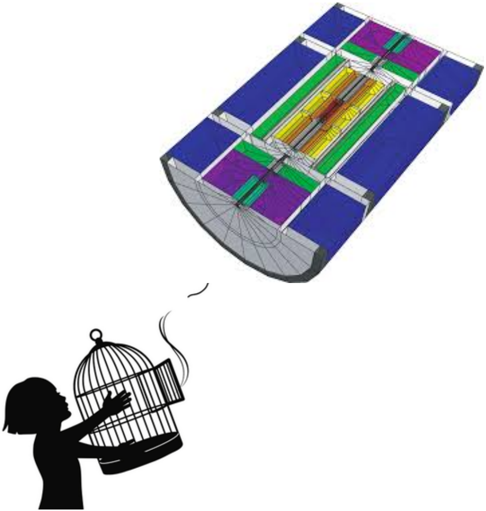
Summary - what we can offer

- the linear collider community has a complete set of tracking tools that are
  - lightweight compared to LHC
  - many pattern recognition algorithms based on
    - topological clustering, Cellular Automatons, conformal mapping
    - partly depending on LCIO and DD4hep - partly standalone
  - track fitting tools that
    - use DD4hep Surfaces as geometry model
    - simple interfaces for tracker hits and tracks
    - framework independent
  - used currently by three detector concepts (ILD, CLICdp, SiD)
  - flexible for adaptation to new detector models
    - in particular if they are described in DD4hep

Outlook - what we like to get

- we are continuously trying to improve our tracking code, e.g.
  - currently navigation is somewhat simplistic and brut-force
    - → would like to benefit from ATLAS code for geometry navigation
  - treatment for non-homogeneous B-fields is not yet optimal
    - → eventually we need a Runge-Kutta solver for arbitrary B-fields
  - implement parallelization where possible
    - → want to benefit from work done for LHC
  - use this forum for exchange of ideas

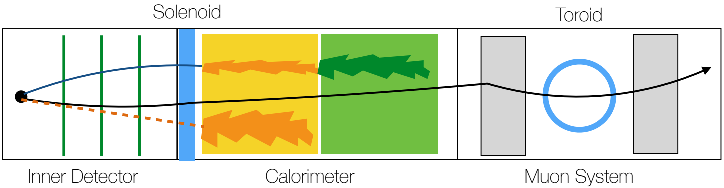
- would like to see more **common HEP tracking software tools**
- ideally under the umbrella of the **HSF**



ATLAS Tracking Software: history, status & prospects

A. Salzburger (CERN) on behalf of the ATLAS Tracking SW

Current structure - from reality to repository



Tracking					
Geometry (TrkDetDescr)	Event Data Model (TrkEvent)	A Extrapolation (TrkExtrapolation)	A Fitting (TrkFitter)	A Calibration, general (TrkTools)	A Alignment (TrkAlignment)

InnerDetector		
EDM (InDetRecEvent)	A Geometry (InDet TrackingGeometry)	A Tools (InDetRecTools)

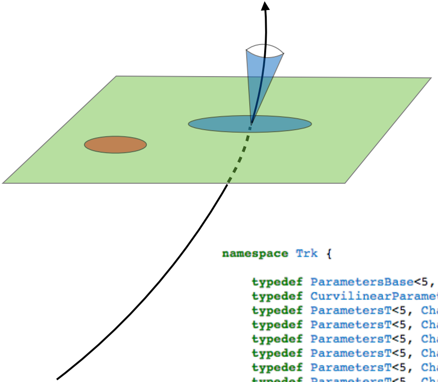
Calorimeter	
EDM	A Geometry (Calo TrackingGeometry)

MuonSpectrometer		
EDM (MuonRecEvent)	A Geometry (Muon TrackingGeometry)	A Tools (MuonRecTools)

A … embedded in Gaudi/Athena structure with AlgTools/Algorithms/Services

ATLAS Tracking Geometry - in a sketch (1)

`Trk::Surface` class acts as a representation of a detector element (proxy mechanism allows to bind it to basically all geometry models)

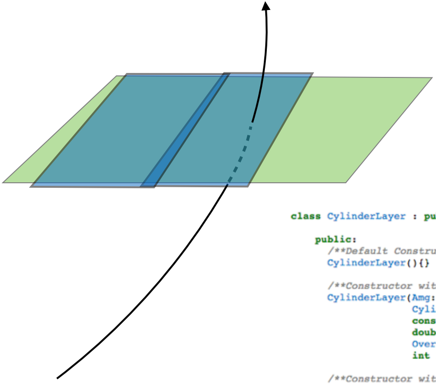


for measurements
and track parameterisation

```
namespace Trk {

    typedef ParametersBase<5, Charged>                         TrackParameters;
    typedef CurvilinearParametersT<5, Charged, PlaneSurface> CurvilinearParameters;
    typedef ParametersT<5, Charged, ConeSurface>               AtaCone;
    typedef ParametersT<5, Charged, CylinderSurface>           AtaCylinder;
    typedef ParametersT<5, Charged, DiscSurface>               AtaDisc;
    typedef ParametersT<5, Charged, PerigeeSurface>            Perigee;
    typedef ParametersT<5, Charged, PlaneSurface>              AtaPlane;
    typedef ParametersT<5, Charged, StraightLineSurface>       AtaStraightLine;

}
```

ATLAS Tracking Geometry - in a sketch (2)

`Trk::Surface` class acts as a base for `Trk::Layer`

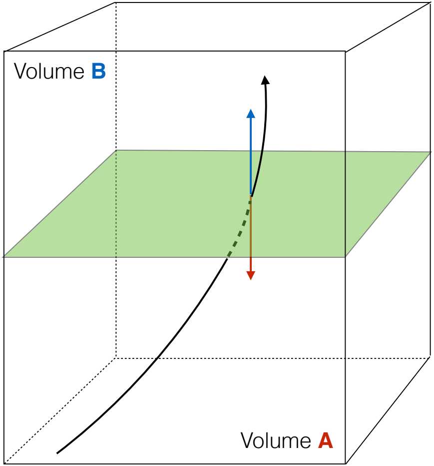


for grouping objects, e.g
detector elements on layers

```
class CylinderLayer : public CylinderSurface, public Layer {

    public:
      /**Default Constructor*/
      CylinderLayer(){}

      /**Constructor with CylinderSurface components and  MaterialProperties */
      CylinderLayer(Amg::Transform3D* transform,
                    CylinderBounds* cbounds,
                    const LayerMaterialProperties& laymatprop,
                    double thickness = 0.,
                    OverlapDescriptor* od = 0,
                    int laytyp=int(Trk::active));

      /**Constructor with CylinderSurface and  MaterialProperties */
      CylinderLayer(CylinderSurface* cyl,
                    const LayerMaterialProperties& laymatprop,
                    double thickness = 0.,
                    OverlapDescriptor* od = 0,
                    int laytyp=int(Trk::active));
```

ATLAS Tracking Geometry - in a sketch (3)

`Trk::Surface` class acts as a (shared) boundaries for `Trk::Volumes`

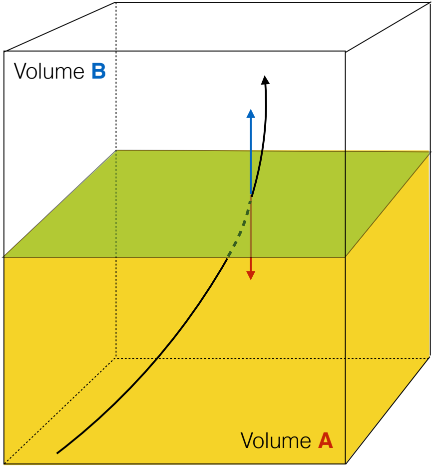


full connective geometry,
i.e. every boundary surface is attached to the next volume(s)

navigation through geometry comes as an intrinsic feature of the extrapolation process.

ATLAS Tracking Geometry - in a sketch (4)

`Trk::Volumes` exist also in a **dense** flavour (e.g. for calorimeter description)



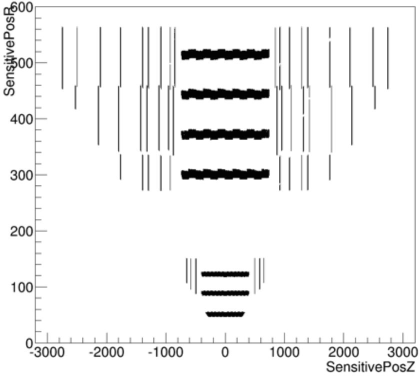
ATLAS has developed a special propagation engine for propagation through dense material with an instantaneous integration of material interactions

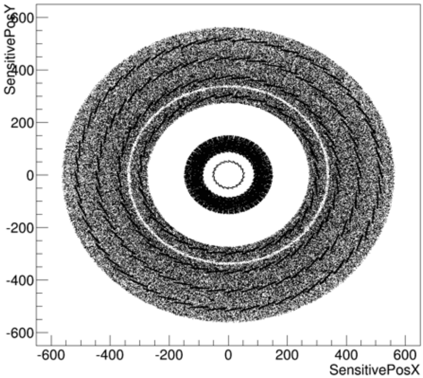
STEP_Propagator
ATL-SOFT-PUB-2008-003

Intermezzo 1 - Fast Track Simulation (Fatras)

ATL-SOFT-PUB-2008-001

- Embedded navigation with Extrapolation engine is a fast track simulation
  - simply changing stochastic material effects integration into MC based one
  - using the reconstruction geometry as a simulation geometry is a common concept for fast simulation





Some concluding remarks

- Is it time to think about reconstruction toolkits ?
  - simulation (Geant) has done this 30 years ago during LEP

    it was simply not possible for experiments to write their full simulation programs (needed a toolkit)

- (Track) reconstruction will be one of the most challenging aspects
  - already for Phase-2 upgrade
  - even more for FCC(-hh)

- New concepts, players and R&D is needed
  - and these are also coming in !

    e.g. Machine learning approaches for HEP:
    [IM LHC Maching Learning WG Meeting, tomorrow](#)

15:55 - 16:15	Machine Learning Challenges in HEP: Upcoming Tracking Challenge 20' Speaker: Andreas Salzburger (CERN)

Tracking software in LHCb

Silvia Borghi, Michel De Cian

on behalf of the LHCb collaboration

Common Track Reconstruction Software Forum, December 3rd 2015

Pattern recognition

- Pattern recognition algorithms are "hand-knitted" for each type of track. Continuous development over the last 10 years (and ongoing).
- However, (almost) no common framework. Ideas were shared, but most of the codes have standalone implementations for most of their tasks.
- Heavy use of parametrisations, no Kalman filter used. Use (pseudo-) χ^2 fits to discriminate good from bad tracks.
- Single threaded CPU implementations, started using vectorisation (mostly Agner Fog's vector class) and vdt, not used on wide scale (for now...).
- No GPU (at the moment).

Track Fitting

- Kalman filter used for track fitting. Using simplified geometry (averaged material description) to speed up fitting (full geometry available if needed).
- Neural net (TMVA trained) for reducing number of fake tracks after Kalman filter, improved "by hand" to make it faster (mostly activation function).
    - Track selection performed with loose χ^2 cut and cut on neural net output.
- Kalman filter is a package, that can be used for all track types: Run I, II and upgrade without much adaptation.
- Speed has been improved, still large contributor to timing in track reconstruction.

Common Software Infrastructure

Benedikt Hegner
(CERN)

Common Track Reconstruction Software Forum
3.12.2015

What to do for collaborative working

- Enable collaborative working with making your source code available

- Allow collaborative working by putting your software under a proper license

- Support collaborative working by setting up build and testing infrastructures and nightlies

- Avoid “impedance mismatch”
    - how much time do we loose on making different tools with different conventions work w/ each other

Making source code available

- Git is state-of-the-art for code management

- Many free services around one could use, e.g.
  - GitHub
  - GitLab at CERN

- Both provide nice features like easy forking, merge requests, code review, etc

- The HSF is an organization on GitHub
  - but obviously any public place for code does the job

Licenses

- Our community is very bad when it comes to licensing
  - Often forgotten or ignored
  - Wrongly applied

- Should make sure new efforts do it properly from the start

- Boundary conditions given by
  - The fact that things have to stay open
  - Your personal take on the free software movement
  - Software you take advantage of and their licenses
  - Rules of your collaboration and employer

- Licenses to consider
  - GPL - if your and all software using it should stay free
  - LGPL - if your software and all changes to it should stay free
  - Apache2 - if you want to provide your software w/o little constraints on people using it
  - Use other licenses only if there is a **strong** reason for it

- Some more information on licenses in HEP available here and here

Development infrastructure and nightly builds

- To share software with others one has to make sure it compiles, runs and yields proper physics…

  … outside the environment it was originally developed in!

- Multiple free nightly build services for open-source projects available, like Travis CI
  - Nicely integrate with GitHub / GitLab
  - Allow compilation and simple tests

- They do not easily cover
  - (CPU) performance studies or validation do not fit into that model
  - Multiple platform support
  - “Exotic” machines
  - Direct debugging of failures

- Doing it properly involves some boring setup and maintenance work
  - People rarely have time for that!

- Idea by HSF contributors is to set up a basic build and test cluster at CERN the various tracking software projects can take advantage of
  - Do the work only once!
  - Taking advantage of jenkins and docker containers
  - Allowing interactive access for debugging