Tracking Code

Paul Dauncey,

Imperial College London

Motivation: ECAL position resolution

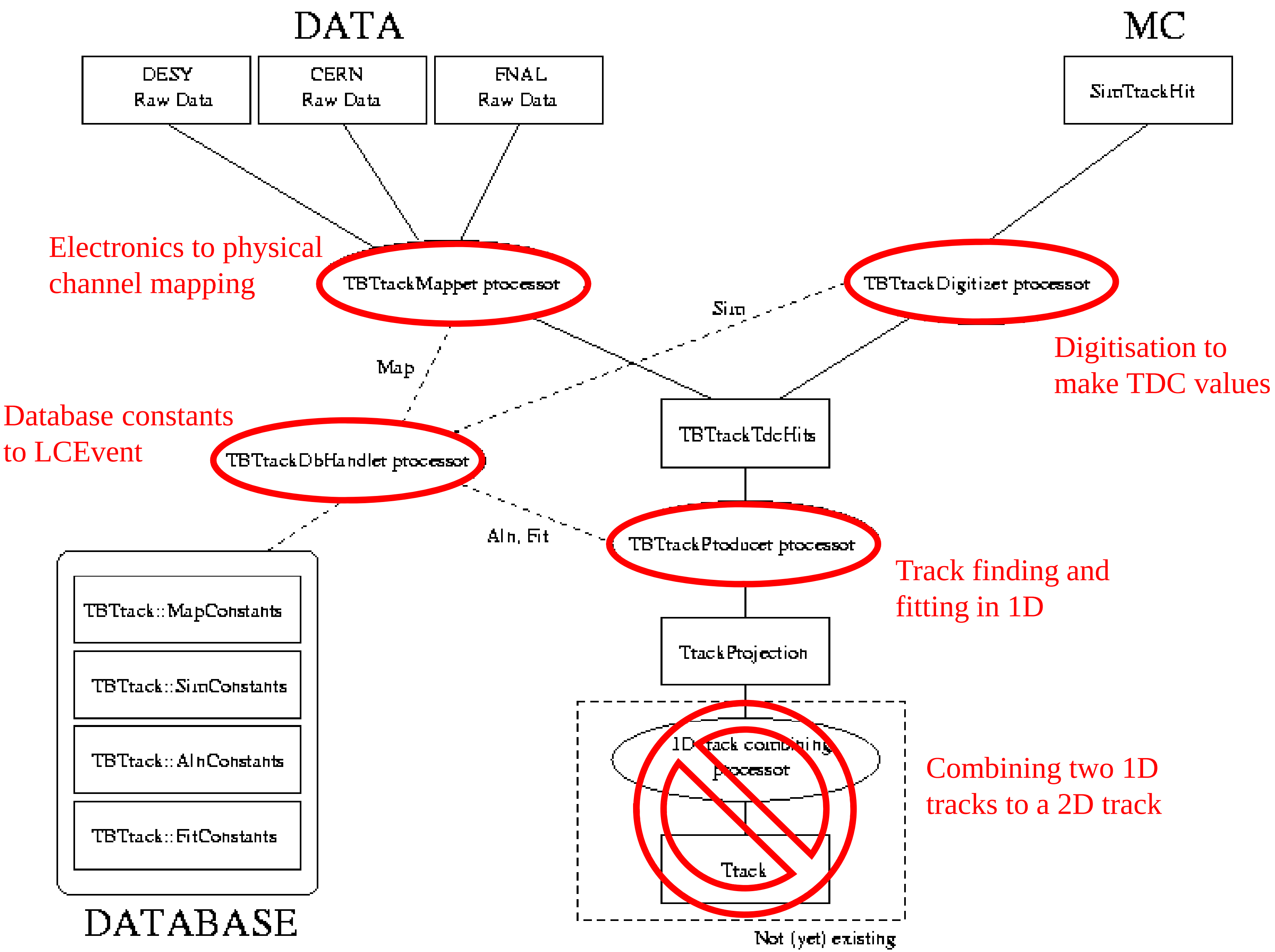
- From the analysis document submitted for LCWS

Table 1: *Systematic errors on the track resolutions valid for both x and y coordinates. The total systematic error is taken as the quadratic sum of the individual contributions.*

	Beam Energy (GeV)					
Source	1.0	2.0	3.0	4.0	5.0	6.0
Position resolution (mm)						
Simulation statistics	0.02	0.01	0.01	0.01	0.01	0.01
Residual misalignment	0.16	0.09	0.06	0.04	0.02	0.02
Material modelling	0.13	0.07	0.04	0.03	0.03	0.02
Intrinsic resolution	0.05	0.05	0.05	0.05	0.05	0.05
Background rate	0.05	0.02	0.02	0.02	0.01	0.01
Total systematic error	0.22	0.12	0.09	0.07	0.06	0.06
Angle resolution (mrad)						
Simulation statistics	0.02	0.01	0.01	0.01	0.01	0.01
Residual misalignment	0.02	0.02	0.01	0.00	0.00	0.00
Material modelling	0.23	0.12	0.08	0.06	0.05	0.04
Intrinsic resolution	0.03	0.02	0.02	0.02	0.02	0.02
Background rate	0.14	0.04	0.02	0.02	0.01	0.01
Total systematic error	0.27	0.13	0.09	0.07	0.06	0.05

- These systematics were evaluated by
  - Changing the relevant parameter in the reconstruction (or digitisation)
  - Rerunning the relevant parts of the track reconstruction (and digitisation)
  - Finding the resulting change to the position (and angle) resolution
- The reconstruction code was written specifically to support this

Reconstruction code organisation



Critical features

- Data and MC share common input format
  - TbTrackProducer does all “real” work
  - TBTrackMapper code does “digital” mapping only (almost)
  - TBTrackDigitiser puts data in the same format
- Database is used to store all parameters
  - Reconstruction cuts as well as constants
  - Ensures parameters will always be known in future
- All database values used are put into the LCEvent
  - Isolates all database interactions from the rest of the reconstruction; very useful for independent code development
  - Only TBTrackDbHandler interacts with database itself
  - Other processors use the values found in the LCEvent
  - Makes use of nice feature that database and event data are in same format

Features for systematics studies

- Rerun on reconstructed data, not raw data
  - Most of reconstruction untouched so don’t want to redo this
  - Cannot run other reconstruction “wrongly” and get other changes which appear to be systematic under study
  - Keeps processing time per run very short; few minutes
- Trivially reproducible
  - Crucial to controlling changes during systematics studies
  - Database constants and event data are both in LCEvent
  - No issues of database being updated since reprocessing, etc.
- Does not require access to database
  - All data in LCEvent so database not required after original reconstruction
  - No knowledge required; no version-tagging, local copies, etc.
- Does not require writing new output files
  - Original track collection replaced
  - Rerun done on the fly and can be analysed in the same job

Example of use

- Example: systematics due to alignment errors
  - Run analysis with modified alignment and see effect on track resolution
- First processor (not normally run in reconstruction)
  - Finds AlnConstants collection in LCEvent and modifies locally
  - Deletes AlnConstants collection and replaces with modified version
  - Also deletes TrackProjection collection of 1D tracks
  - Technically nasty as LCEventImpl sets event data to be read-only; is this needed?
- Secondly, run TBTrackProducer exactly as previously
  - Processor picks up modified AlnConstants values from LCEvent
  - Produces identical format output as original with same collection name so all analysis code downstream works without modifications
- Similar if systematic requires digitisation
  - First processor removes TBTrackTdcHits also in this case
  - Run TBTrackDigitisation before TBTrackProducer

Reconstruction requirements

- Not feasible to fully study all raw data "by hand" before a central reconstruction pass
- Studies which can be done on raw data
  - Electronics to physical mapping can be checked on example runs
  - A few runs can be used to get internal alignment and relative drift velocity
  - Rough global positioning possible as defined by beam
- First reconstruction
  - Assumes runs throughout run period have same alignment
  - If alignment is time-dependent, then shows up as drop in efficiencies and/or worse resolution and/or lower number of matches to ECAL showers
  - With reconstructed ECAL data; compare track projections to shower barycentres to get absolute scale of (equivalent of) drift velocity
- Second (and subsequent) reconstructions
  - Put in time-dependent alignments (if needed) and correct for absolute drift velocity

Reconstruction requirements (cont)

- Scattering contribution to track fit is energy-dependent
  - Differ run-to-run so significant work to load database before reconstruction
  - Actual matrices used calculated from MC so need simulation runs with a wide range of energies to work with
  - For CERN; currently put to zero; extra inefficiencies at lowest energies
- Hooks available for fitting "backwards" (i.e. upstream)
  - Used for determining beam spot size and angular divergence to put into Mokka generation
  - Also can use beam spot, once known, as extra constraint on track fit
  - Beam spot determination must have scattering as origin is so far upstream
- All this requires coordination of central reconstruction passes
  - It is unrealistic to expect a single (or even two) reconstruction runs is enough
  - Needs reconstruction pass start dates announced well in advance to prepare
  - Need to allow time for several passes before next release of results

MC digitisation

- Reseed random generator for every event
  - Repeatable so minimise statistical fluctuations for systematic studies
  - Currently from event number; better ideas?
- All parameters specified from database in SimConstants
  - Rate of noise hits, intrinsic resolution of chambers, efficiency of chambers, alignment (to convert spatial position to TDC hit value)
  - All information stored; digitisation is reproducible
- Needs at least one real data and MC reconstruction pass to know correct values to be used in MC
  - Noise, resolution and efficiency have to be matched to real data
  - These may depend on beam setting so potentially need different MC values for each run
- TBTrackProducer reconstruction code works on data and MC
  - No potential biases due to different algorithms in data and MC

Digitisation (cont)

- Mokka simulation has no "internal" chamber structure imposed
  - Translation of hits to TDC values purely in TBTrackDigitisation
  - Allows arbitrary alignment constants for MC to be chosen after Mokka; can be adjusted to agree with data without regeneration
  - Only consistency requirement between Mokka and reconstruction geometry is in z position of chambers; these are not values which change
- Database alignment constants for MC digitisation and for MC reconstruction are separately specified
  - Allows misalignment to be included in MC
- Run dependence means unrealistic to have parameters in steering files when reconstructing large samples of MC runs
  - E.g. setting equivalent run number in steering file for every run ☹
  - Need automated run number equivalence determined from MC file itself

Conclusions

- Tracking is an example of a code arrangement which allows
  - Easy and quick rerunning of reconstruction and digitisation
  - Following first reconstruction, no database access is required
  - Same code for almost all data and MC reconstruction
- This is one possible implementation model
  - Serves as existence proof that this is feasible
  - Not necessarily the best but some features could be reused
- Other aspects
  - Several reconstruction passes will be required
  - Timely generation and reconstruction of MC is needed for optimal reconstruction constants
  - Data and MC reconstruction need to be produced in step to allow values to be tuned