Alignment using DetectorModel

- Existing Infrastructure for using alignment
- Alignment objects in the database
  - What's there by default
  - How to add new things
  - How to access
- Proposals for global and local tracking alignments
  - Module Sequences
  - Fitting
- Overlap of Alignment and Calibration

Existing Alignment Infrastructure

- in DetectorModel
  - DetElems can be aligned
    - Globally (all elements in a set relative to the global x,y,z coordinate system)
    - Locally by element (individual element relative to its own local coordinate system)
    - Locally In sets (DetElemSet)
      - contained elements are slaves to the motion of the 'master' object
      - Used for Dirc barbox alignment
    - Alignments can be applied and removed
    - Built in to the DetElem base class: no user code required
  - Alignment parameters are defined by DetAlignElem class
    - translations (x,y,z) and rotations (small!) about axes
    - Uses CLHEP classes HepAlignAngles and HepTransform
    - Used for both global and local alignments
    - A collection of DetAlignElem objects can locally align an entire DetSet at 1 time
      - ID # and name match with DetElems
    - Can be included as a data member of a persistent class
      - SvtPGlobalAlign, SvtPWaferAlign, + SvtPPassiveAlign in SvtGeom package
    - Very similair in structure to CalParmChan (more later)
  - Alignment moves both the material (for KalmanFit) and hits (for SvtWafers)
  - Hash tables inside DetSurfaceSet are recomputed automatically when the alignment changes (on the next intersection)

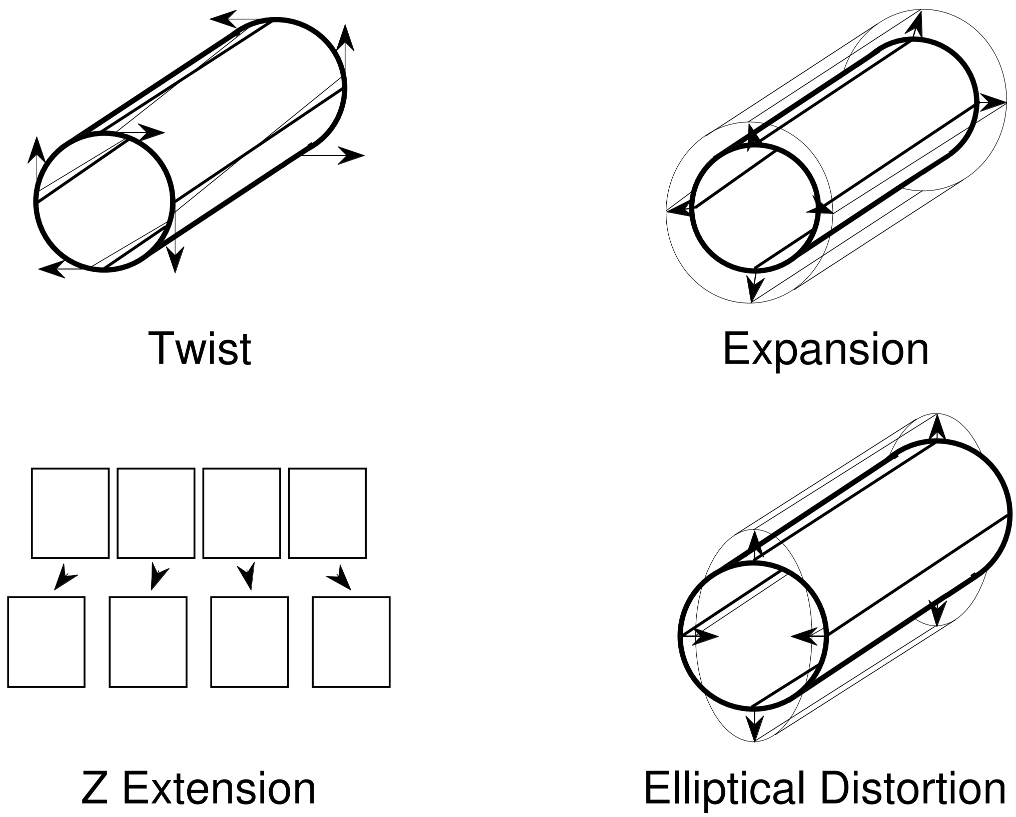
Alignment Interface (for Svt)

- in SvtGeom
  - Apply new alignment (removes old first)
    - global, wafers, and passive elements separately
  - Store alignment in the database
    - Independent time sequences in conditions database for global, wafer, and passive alignments
    - SvtPGlobalAlign, SvtPWaferAlign, (SvtPPasiveAlign)
    - Nominal geometry stored separately (SvtPLayout)
- in SvtEnv
  - Hidden behind SvtEnv::getSvtDetector
  - Implemented in SvtDetectorProxy
  - Automatically follows time dependence of alignments using the event time
    - DON'T CACHE objects returned by the environment in reco code!!!
  - Interface allows manual override (optional BdbTime* argument to getSvtDetector)

Svt Alignment Constant Loading

- Performed by SvtEnvLoad (called by gmake database.import)
- Loads time-ordered list of layouts
  - MDC1 (Jan 1,1990), IR2 (Jan 1,1999) layouts
  - Special version for testbeam layouts
  - Question: what layout do we use for MDC2?
- Loads time-ordered list of global alignments (GRL)
  - Specific distortions for specific time intervals
  - Loaded into the future, to avoid conflict with normal MC analysis
- Loads time-ordered list of wafer alignments (currently null)
  - Need to prepare files with pathological distortions (global deformations)
  - Should place in the future
- Can access these using BdbSetTime (in BdbModules)
  - Sets the event time in the environment (at begin and event entries)
    - option 1: use current program time as event time
    - option 2: use specified time, step a specified amount each event (via talkto)
  - Will become a standard module for MDC1 data in the near future

Global Deformations



Alignment Module Sequence* (global)

- Event Selector Module (μ-pairs, hadronic, ...)
- Special track finding in SVT and DCH
  - Wide roads and loose cuts to avoid bias
  - Ignore tracks which aren't well isolated (efficiency is not important)
  - Can be implemented as special parameters in existing modules
- Track Fit (Kalman)
- Track Selector module
  - momentum cuts, uniform spatial distribution, isolation, (overlaps)
  - Based on TrkPredator
- TrkMerge: simply record which SVT + DCH track go together
- Alignment accumulator module (many possible)
  - compute SVT + DCH track consistency (+projections, derivatives,...)
  - Store values for selected tracks in event entry
  - Process (fit) data in endJob entry
    - output parameters stored in database
  - Could use CalChan subclasses for accumulation and fit

* Based on ideas first presented by David Kirkby and myself in 1996, translated to use the Framework.

Alignment Module Sequence (local)

- Same sequence as global up to TrkMerge
  - Different parameters in track selector (overlaps)
- TrkRep (KalRep?) must support special functions
  - Compute consistency of individual hits with remaining fit, + derivatives,...
  - Fit μ-pair to 5 parameters (vertex + boost constraint)
  - Store DCH track as a parameter constraint
- Store Tracks to a dedicated output stream
  - Objy? RW persistence?
- Accumulation, fit, and iteration performed on this output
  - Tracks refit every iteration for current alignment
  - (optionally) constrain using DCH track, survey measurements, vertex
  - Can iteration be supported in the Framework?

Calibration/Alignment overlap

- Some parts of calibration and alignment are similair
  - Accumulation
  - Fitting
- Other parts are not
  - Calibration validation and verification
  - Persistent storage
- Some classes might be reused
  - CalAddChan
    - Accumulation interface in calibration
  - CalFit
    - Processes CalAddChan's after accumulation
  - CalParmChan (Matt Weaver)
    - Stores parameters and covariance
    - 6D CalParmChan data members match exactly with DetAlignElem
- Some technology could be reused
  - Distributed calibration in PR