Introduction to GENFIT

Sebastian Neubert

TUM

23.1.2008

Running the Demo

- Call `recotasks/demo/loadRecoLibs.C` from your rootlogon.C!
- Get latest revision and activate GEANE in global `CMakeLists.txt`:

  `add_subdirectory (geane)`
- Build it!

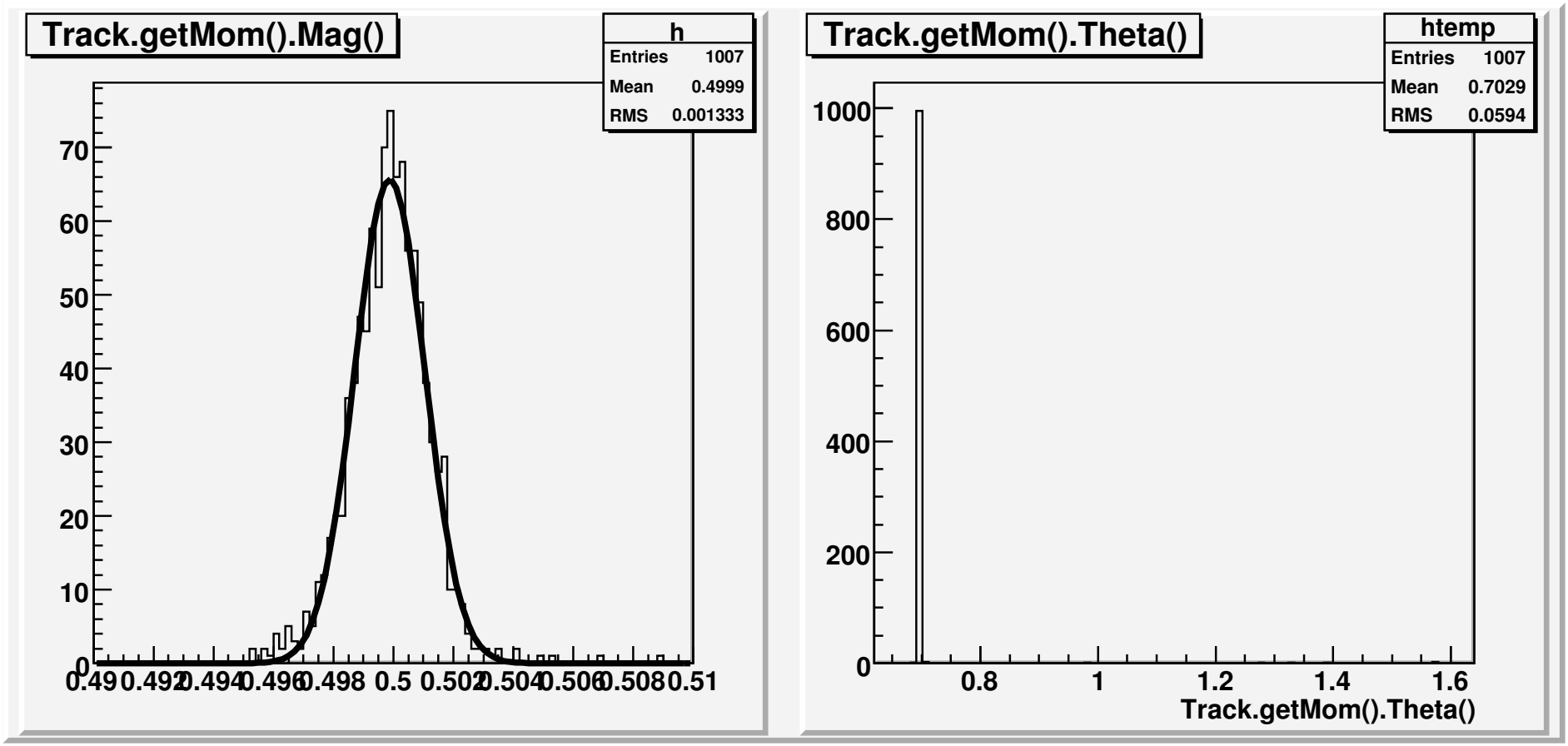
recotasks/demo

`DemoKalmanTask.cxx DemoPatternRecoTask.cxx DemoRecoHit.cxx runDemo.C`
`DemoKalmanTask.h DemoPatternRecoTask.h DemoRecoHit.h runMC.C`

- Start the Monte Carlo simulation with
  `root -l -q recotasks/demo/runMC.C`
  - Creates 1000 single-π events (500MeV, 40°) in the TPC
  - $\rightarrow$ file `demo.mc.root`
  - look at MC-hits with the treeviewer
- Start the reconstruction demo `runDemo.C`
  - $\rightarrow$ file `demo.reco.root`

root -l recotasks/demo/plotDemo.C

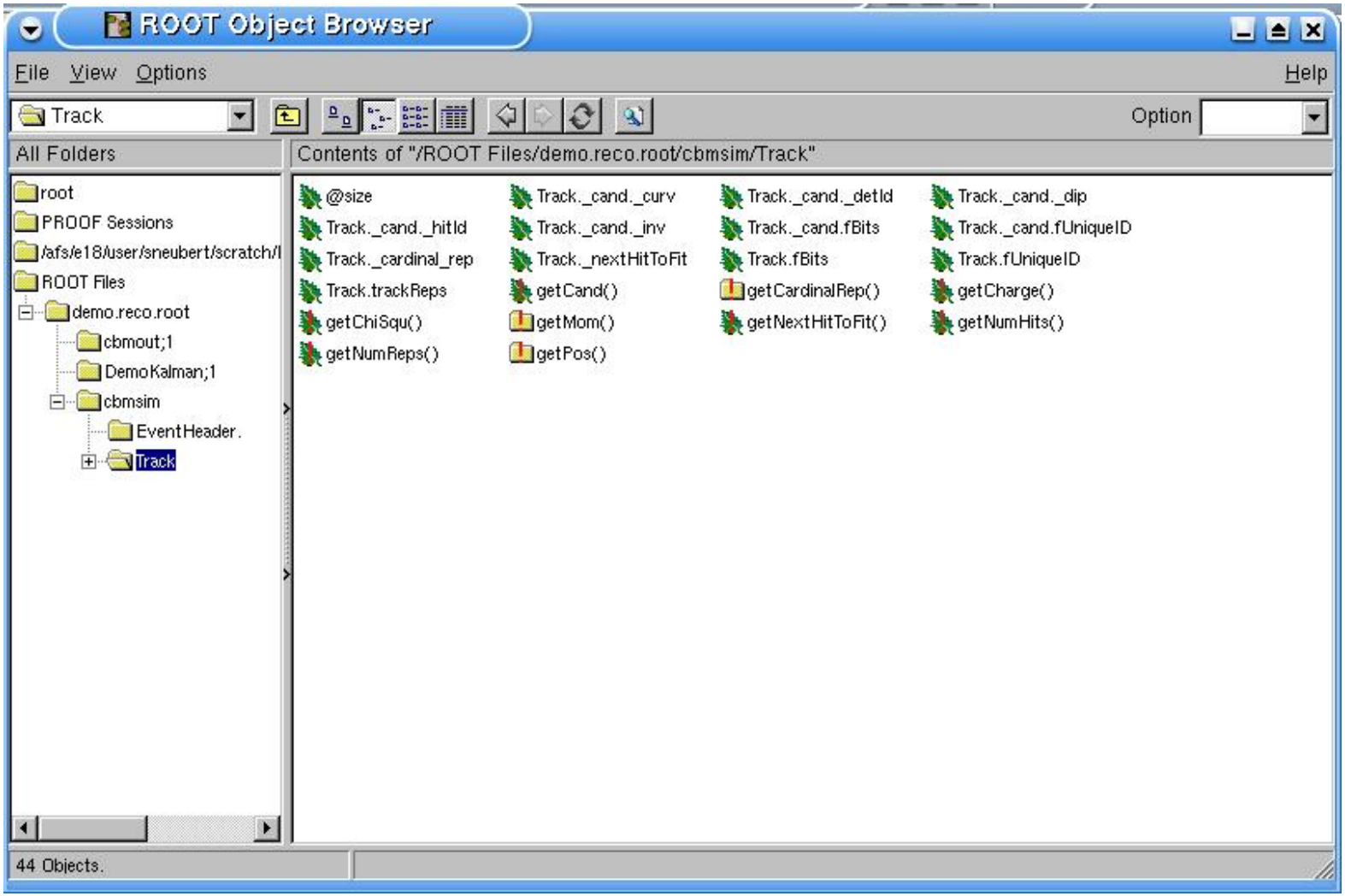
(don't end ROOT afterwards!)



Note that you can access all member functions in the `Draw` - command,

e.g.: `cbmsim->Draw(''Track.getCardinalRep().getCovElem(0,0)'');`

Browse the Track objects with TBrowser



ROOT-Bug(?): TBrowser only works when plotDemo.C has been called before!

What is going on inside runDemo.C?

- `DemoPatternRecoTask`
  - Uses CbmPoints
  - Creates track candidates by looking at monte carlo id
  - Sets up Track objects with LSL track representation
  - Initializes track representation from MC-data
- `DemoKalmanTask`
  - Loads `RecoHits` into the tracks
  - Performs single-pass Kalman fit
  - Fills some histograms

Adding the MVD to the Fit

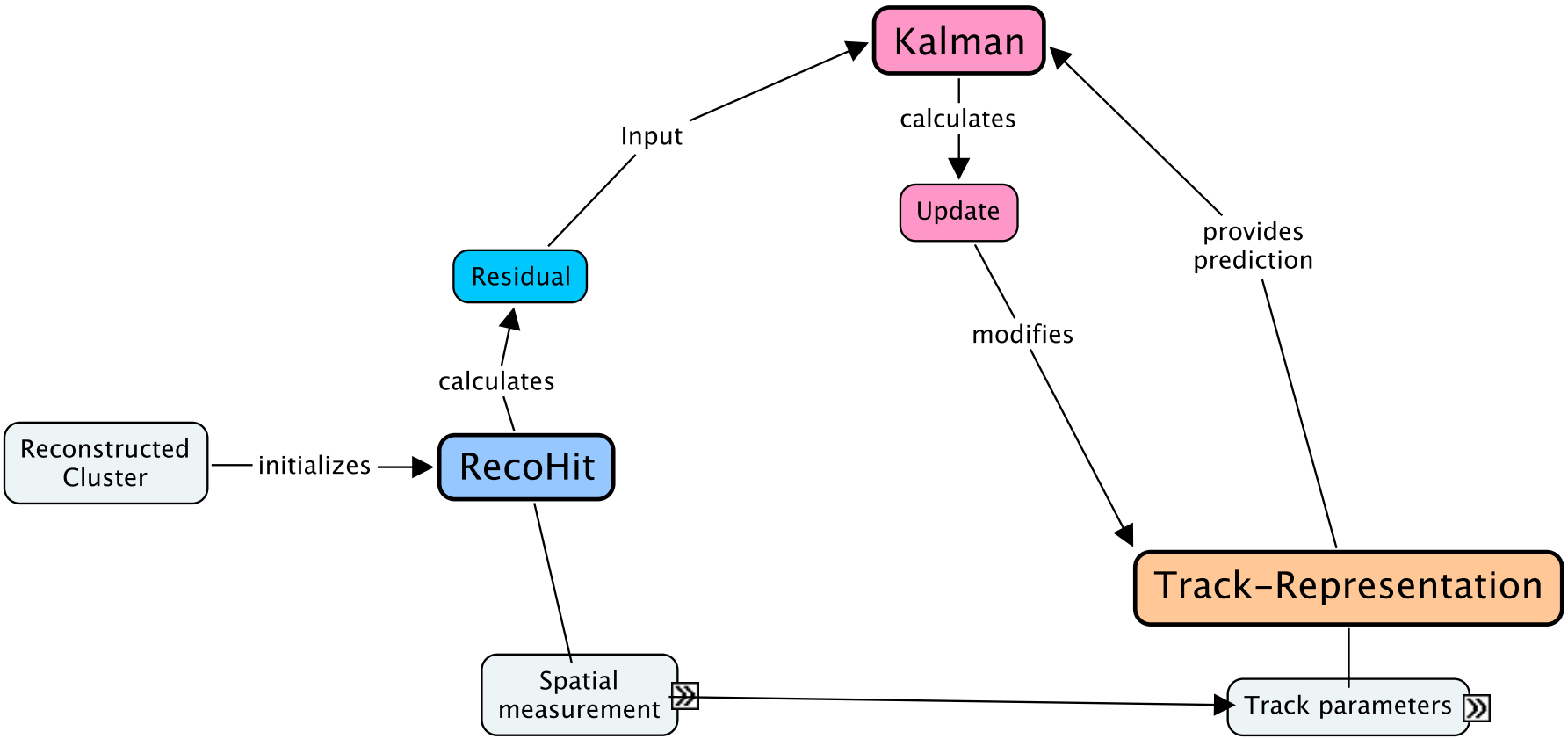
- all MC-Points derive from `CbmMCPoint`
- $\Rightarrow$ we can treat them equally.
- $\Rightarrow$ this makes it possible to switch on a new detector in the macro:

In `runDemo.C`

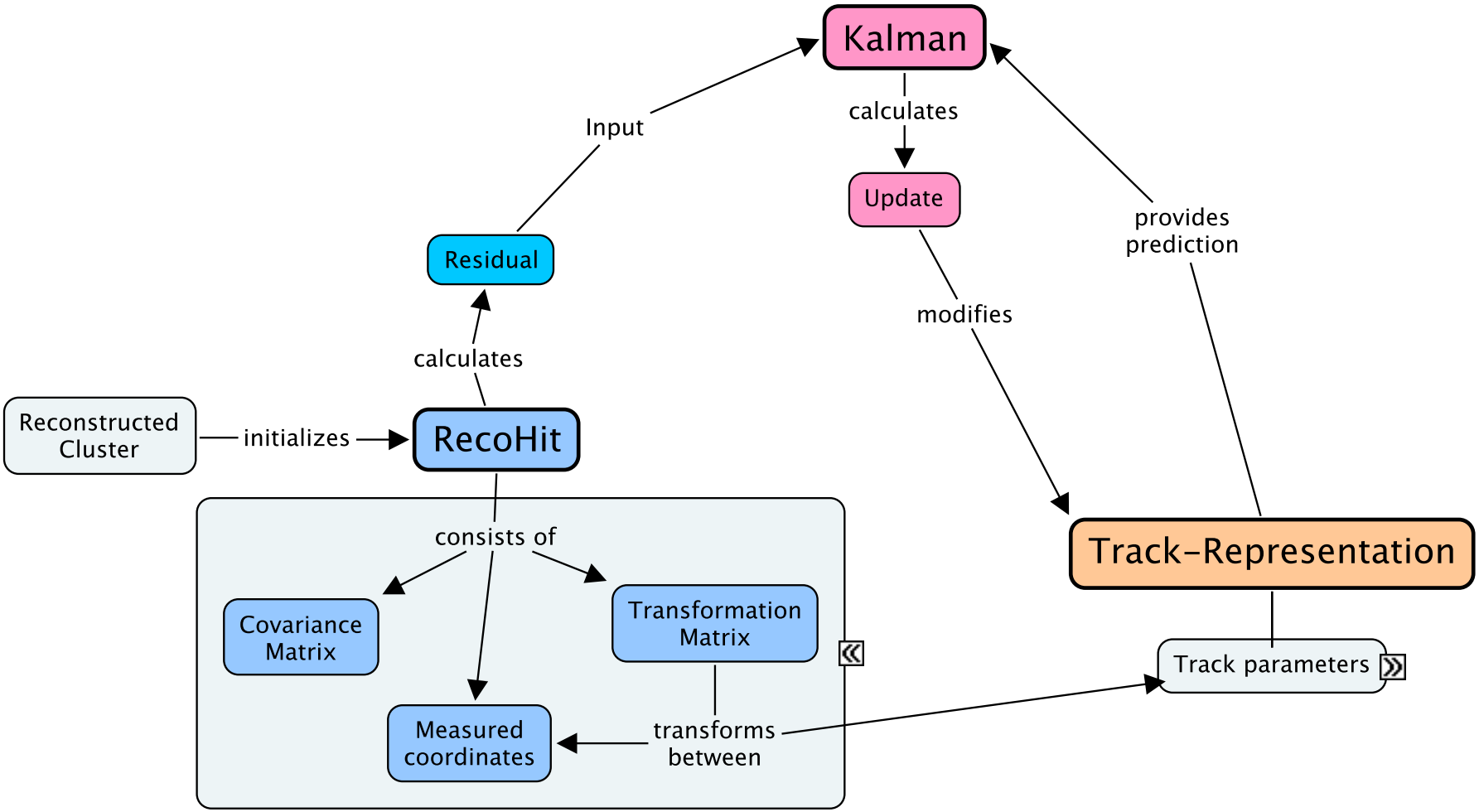
```
// -------------------------------------------------------------
// ----- Reco Sequence ---------------------------------------
DemoPatternRecoTask* DemoPR = new DemoPatternRecoTask();
DemoPR->AddHitBranch(2,"PndTpcPoint");
DemoPR->AddHitBranch(3,"MVDPoint"); // add this line!
DemoPR->SetPersistence();
fRun->AddTask(DemoPR);
DemoKalmanTask* DemoKalman = new DemoKalmanTask();
DemoKalman->AddHitBranch(2,"PndTpcPoint");
DemoKalman->AddHitBranch(3,"MVDPoint"); // add this line!
DemoKalman->SetPersistence();
fRun->AddTask(DemoKalman);
```

- MVDPoint is the name of the TClonesArray containing the MVD data
- The MVD is assigned the detector-id 3

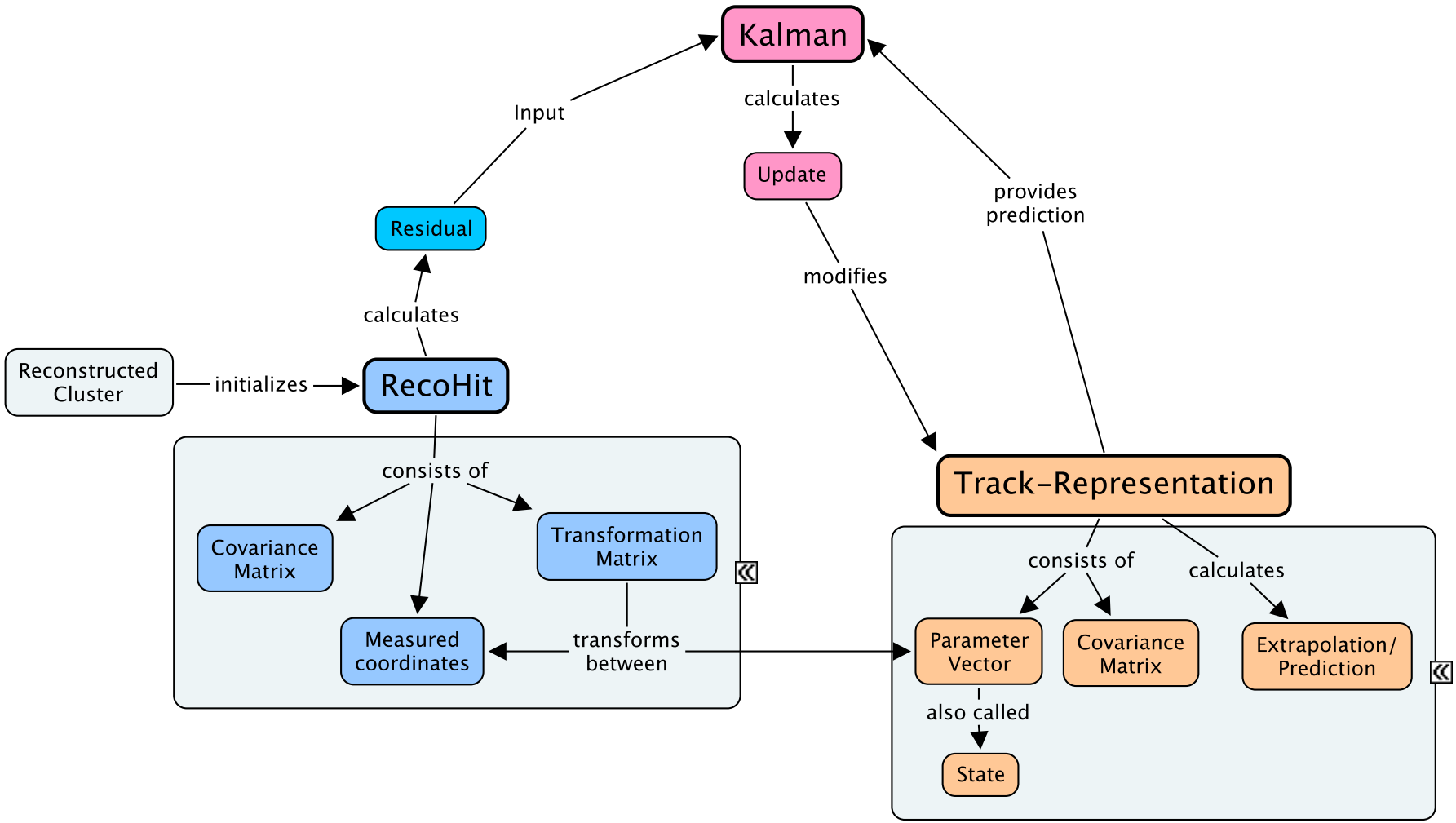
How the GENFIT components work together



How the GENFIT components work together



How the GENFIT components work together



Example: DemoRecoHit

- Takes data from any CbmMCPoint
- Represents a measurement in XY-Plane at fixed Z

Initialization

```
typedef RecoHitIfc<PlanarHitPolicy> PlanarRecoHit;
...
DemoRecoHit::DemoRecoHit(CbmMCPoint* point)
:  PlanarRecoHit(NparHitRep)
{
_hitCoord[0][0] = point->GetX();
_hitCoord[1][0] = point->GetY();
setDetPlane(DetPlane(TVector3(0,0,point->GetZ()),TVector3(1,0,0),
TVector3(0,1,0)));
double sigx=0.1;
double sigy=0.1;
_hitCov[0][0] = sigx*sigx;
_hitCov[1][1] = sigy*sigy;
}
```

- DetPlane $\longrightarrow$ coordinate system of the detector
- Non-planar detectors $\longrightarrow$ `RecoHitIfc<SpacepointHitPolicy>`

Example: DemoRecoHit

Transformation Matrix H

```
DemoRecoHit::setHMatrix(const AbsTrackRep* stateVector,
const TMatrixT<double>& state)
{
if (dynamic_cast<const LSLTrackRep*>(stateVector) != NULL) {
// LSLTrackRep (x,y,x',y',q/p) and DemoRecoHit (x,y)
_HMatrix.ResizeTo(NparHitRep,5);
//
_HMatrix[0][0] = 1.;
_HMatrix[0][1] = 0.;
_HMatrix[0][2] = 0.;
_HMatrix[0][3] = 0.;
_HMatrix[0][4] = 0.;
//
_HMatrix[1][0] = 0.;
_HMatrix[1][1] = 1.;
_HMatrix[1][2] = 0.;
_HMatrix[1][3] = 0.;
_HMatrix[1][4] = 0.;
}
...
}
```

Advanced Example: TPC reconstruction

- `macro/tpc/tutorial/runMC.C`
- `macro/tpc/tutorial/runDigi.C`
- `macro/tpc/tutorial/runReco.C`
- open `tutorial.reco.root` and start treeviewer
- Have a look at the `TrackFitStat` objects!

ToDo

- Write your own RecoHit (Subdetector experts in charge!)
  - Inherit from proper `RecoHitIfc`
  - Provide initialization constructor from your clusters
  - Provide proper DetPlane initialization
  - Implement HMatrix for the TrackReps you want to use
  - Call for help if needed! Specific questions welcome!
- Get GEANE running (`DemoPR->useGeane();`) (Expert needed?)
- Check Kalman-Filter (Expert needed!)
  - General cross-check
  - Implement smoothing
  - Implement multiple passes
- Build common tracking sequence (Manager needed!)
- Use GENFIT to implement other tracking stuff (Expert needed!)
  - Vertexing!
  - See e.g. `recotasks/V0Selector.cxx` for simple V0 reconstruction!

DemoPatternRecoTask – Initializing a track

- A track consistes of
  - a list of hit-indices
  - (at least one) track representation

Track: public TObject – instantiation:

```
AbsTrackRep* rep=new LSLTrackRep(); // note interface class!

Track* trk=new Track(rep);
```

- Hit-indices:
  - Detector-Id
  - Index of hit in TClonesArray

Adding hits to the track:

```
trk->addHitIndices(detId,index);
```

Further things you can do

Disable track rep from fit

```
trk->getTrackRep(0)->setStatusFlag(2);
```

- need a convention what the status codes mean

Add further track reps

```
trk->addTrackRep(another_rep);
```

- all reps will be fitted in parallel

Store tracks

```
rack* trk=new((*_trackArray)[_trackArray->GetEntriesFast()]) Track(rep);
```

- hit indices will be stored
- reco hits will NOT be stored (see next slide)
- track reps will be stored

Loading hits into the track

In the track only hit-indices are stored

- No deep copy of hits needed
- A hit can be member of several tracks
- Different types of hits in one track

DemoKalmanTask – Building a RecoHitFactory

```
_HitFactory = new RecoHitFactory();
.
RecoHitProducer* prod1=new RecoHitProducer<RawMvdHit,MvdRecoHit>(mvdArray);
RecoHitProducer* prod2=new RecoHitProducer<TpcDigi,TpcRecoHit>(tpcArray);
.
_HitFactory->addProducer(MvdDetId,prod1);
_HitFactory->addProducer(TpcDetId,prod2);
```

Loading the RecoHits

```
trk->addHitVector(_HitFactory->createMany(trk->getDetIDs(),trk->getHitIndices()));
```

Running the Kalman Filter

Fitting

```
Kalman fitter;
try{
fitter.processTrack(trk);
}
catch (FitterException e){
std::cout<<e.what()<<std::endl;
}
```

- Only TrkReps with statusflag==0 are processed
- After fitting: Track parameters at last hit
- Error handling with exceptions (see `genfit/FitterExceptions.h`)

Accessing fit result: TrkRep

```
double p=trk->getTrackRep(0)->getMom().Mag();
double chi2=trk->getTrackRep(0)->getChiSqu();
```