Updates in new_geom branch

Container classes

Magnetic field

VTX digitizer

MSD digitizer

TW digitizer

TW rec point

MC par geo

IT tracking

VTX tracking

VTX vertexing

Global reconstruction Action

Event Display

Conclusion

Container class (i)

- Add new base class for cluster/point

```
class TAGcluster : public TAGobject {
  public:
                      TAGcluster();
                      TAGcluster(const TAGcluster& cluster);
   virtual           ~TAGcluster() {}

   //! Get position in local frame
   virtual const TVector3&  GetPosition() const = 0;

   //! Get position error in local frame
   virtual const TVector3&  GetPosError() const = 0;

   //! Get position in detector frame
   virtual const TVector3&  GetPositionG() const = 0;

   //! Get position error in detector frame
   virtual const TVector3&  GetPosErrorG() const = 0;

   //! Add MC track Idx
   void                AddMcTrackIdx(Int_t trackIdx);
   //! Get MC info
   Int_t               GetMcTrackIdx(Int_t index)      const   { return fMcTrackIdx[index];    }
   Int_t               GetMcTracksN()                  const   { return fMcTrackIdx.GetSize(); }

private:
   TArrayI             fMcTrackIdx;              // Idx of the track created in the simulation
   map<int, int>       fMcTrackMap;              // Map of MC track Id

   ClassDef(TAGcluster,1)
};
```

➡ Virtual interface to get position/error in local and detector frame

Container class (ii)

- Interfaced with new class

```
class TAVTbaseCluster : public TAGcluster {
. . .
};
```

```
class TAMSDcluster : public TAGcluster {
. . .
};
```

```
class TATWpoint : public TAGcluster {
. . .
};
```

➡ Ease loop over clusters/points for global reconstruction

(Optimized some member definitions)

Container class (iii)

- TATWpoint: Add MC index info in point constructor

```
TATWpoint::TATWpoint( double x, double dx, TATWntuHit* colHit, double y, double dy, TATWntuHit* rowHit )
: TAGcluster(),
   m_position(x, y, 0),
   m_posErr(dx, dy, 0),
   m_columnHit(new TATWntuHit(*colHit)),
   m_rowHit(new TATWntuHit(*rowHit)),
   m_chargeZ(0),
   m_chargeZProba(0.)
{
   m_column = m_columnHit->GetBar();
   m_row    = m_rowHit->GetBar();

   m_de1    = m_columnHit->GetEnergyLoss();
   m_de2    = m_rowHit->GetEnergyLoss();
   m_time   = m_columnHit->GetTime();

   for (Int_t j = 0; j < m_rowHit->GetMcTracksN(); ++j) {
      Int_t idr = m_rowHit->GetMcTrackIdx(j);
      for (Int_t k = 0; k < m_columnHit->GetMcTracksN(); ++k) {
         Int_t idc = m_columnHit->GetMcTrackIdx(k);
         if (idr == idc)
            AddMcTrackIdx(idr);
      }
   }
}
```

➡ Keep only MC indexes present in both row/col hits

Container class (iv)

- Add error computing on vertex

```
class TAVTvertex : public TAGobject {

private:
    TClonesArray*   fListOfTracks;      // list of track associated to the vertex
    TVector3        fVertexPosition;    // vertex position
    TVector3        fVertexPosError;    // vertex position error

. . .
};
```

```
Bool_t TAVTactNtuVertexPD::SetVertex()
{
    . . .
    Int_t nTracks = ntuTrack->GetTracksN();
    for(Int_t q = 0; q < nTracks; ++q ) {
       . . .
       TVector3 pos = track0->Intersection(fVtxPos.Z());
       pos -= fVtxPos;
       pos.SetXYZ(pos[0]*pos[0], pos[1]*pos[1], 0);
       err += pos;
    }
    if (vtx->GetTracksN() > 0) {
      err.SetXYZ(TMath::Sqrt(err[0]), TMath::Sqrt(err[1]), 0);
      err *= 1/float(vtx->GetTracksN());
      vtx->SetVertexPosError(err);
   . . .
    }
  . . .
}
```

➡ Error compute from the residual btw the vertex pos and the extrapolated tracks

Magnetic field

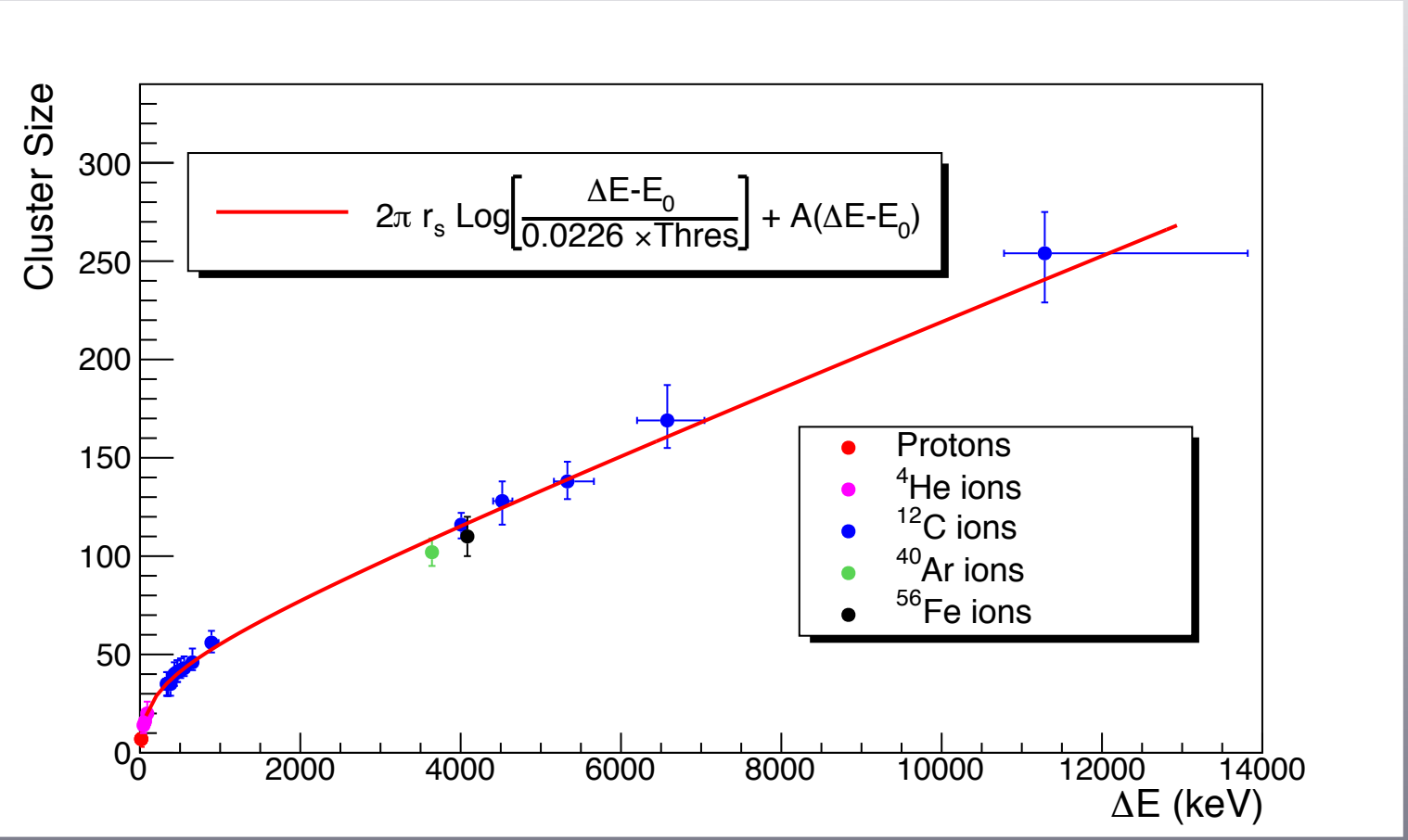
- TADIgenField: new class interfaced with TADIgeoField for GenFit

```
class TADIgenField : public AbsBField {

private:
   TADIgeoField* fField;

public:
   TADIgenField(TADIgeoField* field);
   virtual ~TADIgenField();

   TVector3 get(const TVector3& position) const;
   void get(const double& posX, const double& posY, const double& posZ,
                  double& Bx,         double& By,         double& Bz) const ;


   ClassDef(TADIgenField,1)
};
```

➡ Useful for GenFit global reconstruction

VTX digitizer

- Cluster size: new data (data from C.A. Reidel)



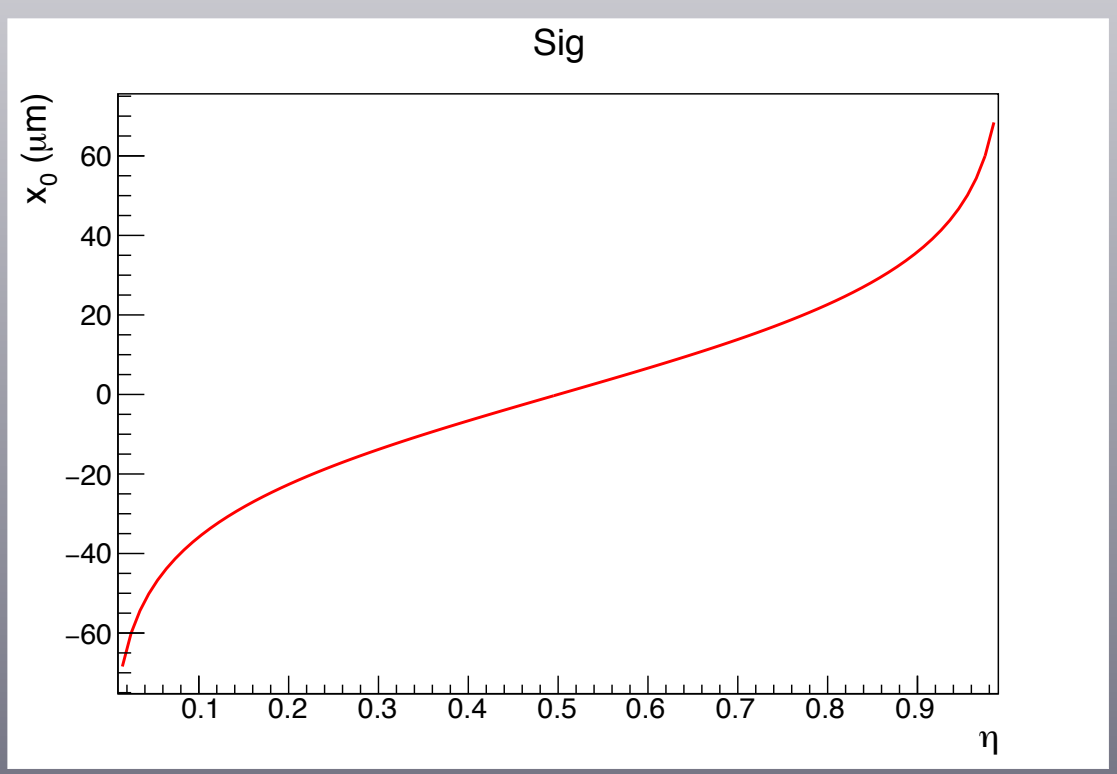
➡ New fitted parameters and new linear parameter added in TAVTbaseDigitizer

MSD digitizer (i)

- Using the η variable vs the impact point X_0

$$\eta = \frac{PH_r}{PH_r + PH_l}$$

where PH_r and PH_l are the pulse height of right and left strip



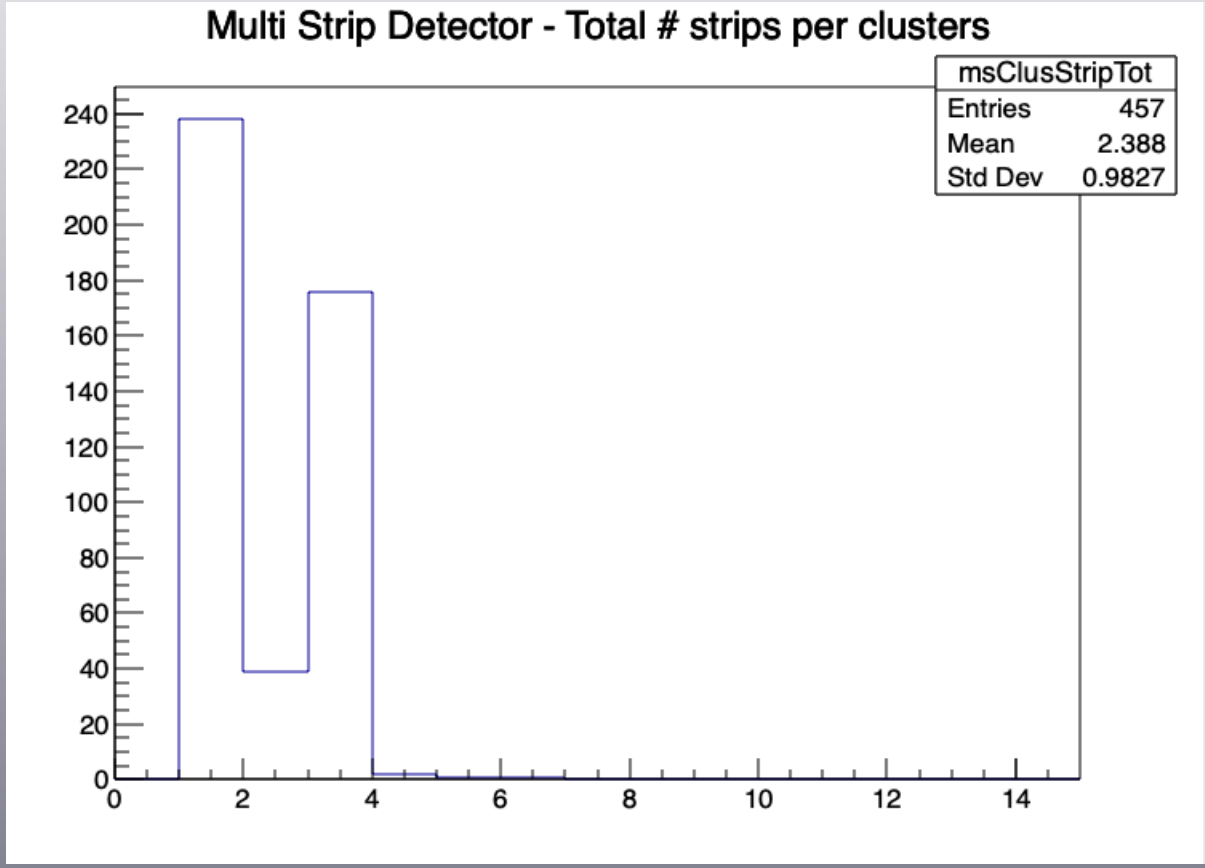
Strip right is located at +75 μm (half pitch)

➡ Gives the charge sharing btw two strips for a given X_0

MSD digitizer (ii)

- Define 3 cases:
  - $\eta > 0.6$ and $\eta < 0.9$: 3 strips fired
  - $\eta > 0.4$ and $\eta < 0.6$: 2 strips fired
  - $\eta > 0.9$: 1 strip fired

➡ and symmetric



➡ limits defines arbitrary, tune with real data

➡ need to take into account 1 strip out of 3 is read out

TW digitizer (i)

- TWdigitizer : add map for pileup particles in the same bar, corrected position

```
class TATWdigitizer : public TAGbaseDigitizer {

. . .
   map<int, TATWntuHit*> fMap; //! map for pileup
};
```

```
Bool_t TATWdigitizer::Process(Double_t edep, Double_t x0, Double_t y0, Double_t /*zin*/,
                              Double_t /*zout*/, Double_t time, Int_t id, Int_t Z)
{
. . .
   pos = (timeB-timeA)/(2*fTofPropAlpha);
. . .
   tof *= TAGgeoTrafo::PsToNs();
. . .
if (fMap[id] == 0) {
      fCurrentHit = (TATWntuHit*)fpNtuRaw->NewHit(view, id, energy, tof, pos, chargeCOM,
                                                  chargeA ,chargeB, timeA, timeB);
      fCurrentHit->SetChargeZ(Z);
      fMap[idA] = fCurrentHit;
   } else {
      fCurrentHit = fMap[id];
      fCurrentHit->SetChargeChA(fCurrentHit->GetChargeChA()+chargeA);
      fCurrentHit->SetChargeChB(fCurrentHit->GetChargeChB()+chargeB);
      // take the shortest time
      if (timeA < fCurrentHit->GetChargeTimeA()) fCurrentHit->SetChargeTimeA(timeA);
      if (timeB < fCurrentHit->GetChargeTimeB()) fCurrentHit->SetChargeTimeB(timeB);
. . .
   }
. . .
}
```

TW rec point

- TATWactNtuPoint: limited search distance

```
//_____________________________________________________________________________
//
Bool_t TATWactNtuPoint::FindPoints()
{
 . . .

   Float_t minDist = 0;
   Int_t minId = -1;
   Bool_t best = false;

   Int_t layer1 = 0;
   Int_t nHits1 = pNtuHit->GetHitN(layer1);
   if (nHits1 == 0) return false;

   Int_t layer2 = 1;
   Int_t nHits2 = pNtuHit->GetHitN(layer2);
   if (nHits2 == 0) return false;

   for (Int_t i = 0; i < nHits1; ++i) {
      minDist = pGeoMap->GetBarHeight() - pGeoMap->GetBarWidth(); // borders have no overlapp
. . .
     }
   }

   return true;
}
```

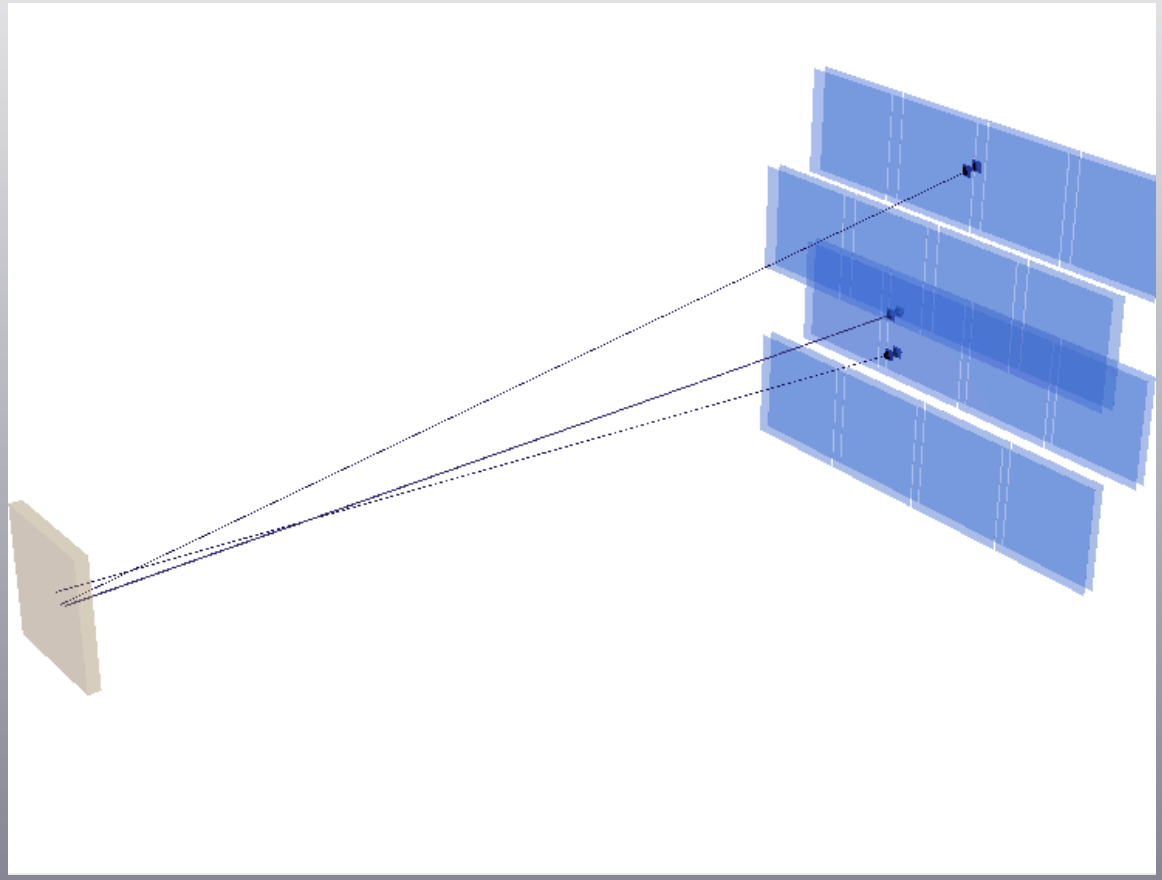
MC par geo

- TAGparGeo : add material name for beam

```
class TAGparGeo : public TAGparTools {
. . .
// Beam parameters
   struct BeamParameter_t : public  TObject {
     TVector3  Position;      // Beam position (cm)
     TVector3  AngSpread;     // Angular spread (cos)
     Float_t   AngDiv;        // Angular divergence (mrad)
     Float_t   Size;          // size of beam, -1 no size (cm)
     TString   Shape;         // Shape of beam
     Float_t   Energy;        // Beam energy
     Int_t     AtomicNumber;  // Z of the beam
     Float_t   AtomicMass;    // A of the beam
     TString   Material;      // Beam material
     Int_t     PartNumber;    // Number of particles in beam
   };
. .  .
   ClassDef(TAGparGeo,2)
};
```

IT tracking (i)

- Add tracking for IT: standard algorithm

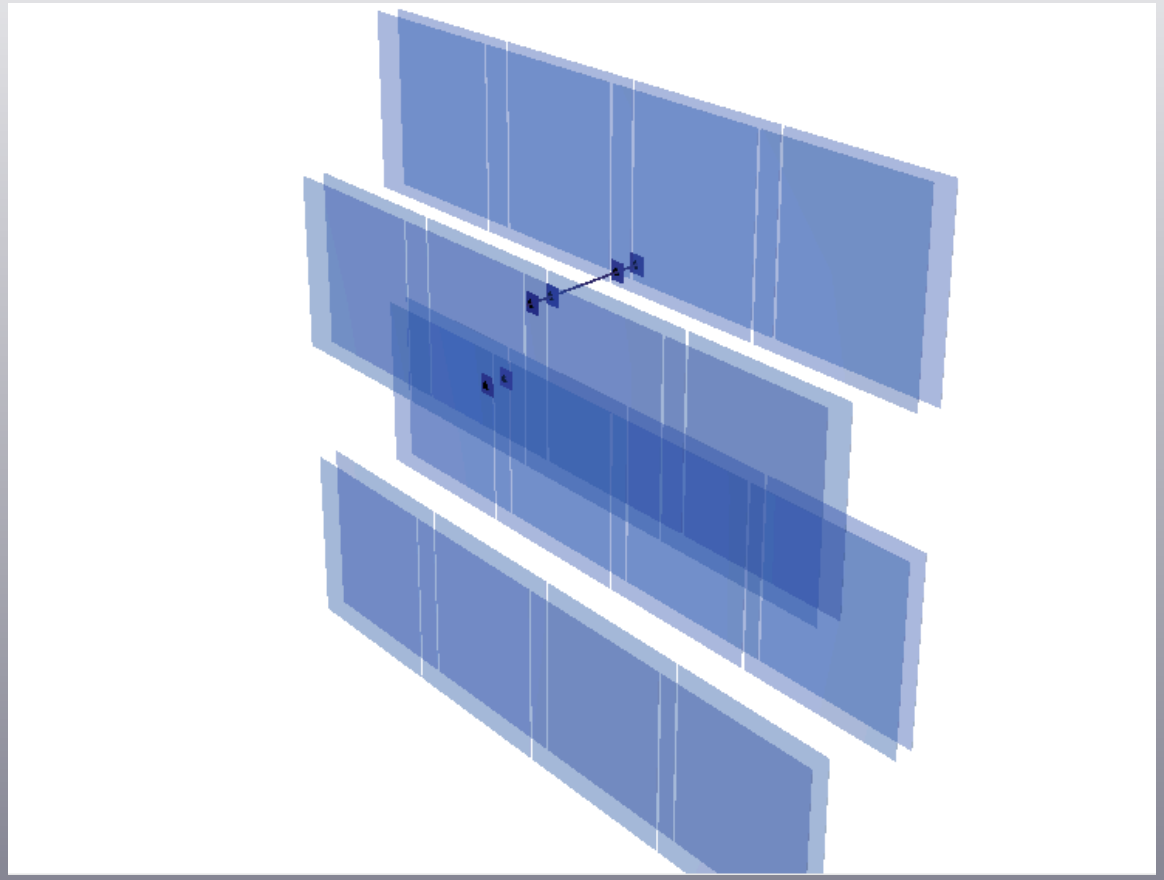


➡ Reconstruct track with at least 2 clusters (increase route, 300 to 600 μm)

- Add tracking action in BaseReco: TAITactNtuTrack
- Add tracks in TAFOeventDisplay class (straight line, neglecting Mag Field)

IT tracking (ii)

- Add tracking for IT: combination algorithm



➡ Reconstruct track with at least 3 clusters

- Add tracking action in BaseReco: TAITactNtuTrackF
- Add tracks in TAFOeventDisplay class

VTX tracking

- TAVTactNtuTrackF : introduce a map to avoid double use of a cluster for tracks

```
class TAVTactNtuTrackF : public TAVTactBaseNtuTrack {

public:

   explicit  TAVTactNtuTrackF(const char* name      = 0,
                              TAGdataDsc* p_ntuclus  = 0,
                              TAGdataDsc* p_ntutrack = 0,
                              TAGparaDsc* p_config   = 0,
                              TAGparaDsc* p_geomap   = 0,
                              TAGparaDsc* p_calib    = 0,
                              TAGdataDsc* p_bmtrack  = 0);
   virtual ~TAVTactNtuTrackF();

private:

   Bool_t FindTiltedTracks();
   Bool_t IsGoodCandidate(TAVTtrack* track);

private:
   map<TAVTcluster*, int> fMapClus;

   ClassDef(TAVTactNtuTrackF,0)
};
```

➡ Not very optimized

VTX vertexing (i)

- TAVTactNtuVertexPD: optimized vertex with binary search instead of a loop

```
void TAVTactNtuVertexPD::SearchMaxProduct(TAVT
     TAVTtrack* linej, Int_t i, Int_t j)
{
  . . .
    Double_t actualProb = 0;
    Double_t maxProb = -10e+10;
    //loop over Z to find the maximum
    for(Double_t iz = fMinZ; iz < fMaxZ; iz +=
        vertexPoint = ComputeVertexPoint(linej
        fi = ComputeProbabilityForSingleTrack(
        fj = ComputeProbabilityForSingleTrack(
        actualProb = fi*fj;
        if(actualProb > maxProb){
            maxProb = actualProb;
            maxPosition = vertexPoint;
        }

    }



    //save the values into a structure
    fProbValuesMax[i*fTracksN + j] = maxProb;
    fRValuesMax[i*fTracksN + j] = maxPosition;
}
```

```
void TAVTactNtuVertexPD::SearchMaxProduct(TAVTtrack* linei,
     TAVTtrack* linej, Int_t i, Int_t j)
{
  . . .
   // Binary search for maximum of probability
   while (TMath::Abs(a - b) > fEps ) {

      vertexPointA = ComputeVertexPoint(linei, linej, a);
      fi = ComputeProbabilityForSingleTrack(linei, vertexPointA);
      fj = ComputeProbabilityForSingleTrack(linej, vertexPointA);
      probaA = fi*fj;

      vertexPointB = ComputeVertexPoint(linei, linej, b);
      fi = ComputeProbabilityForSingleTrack(linei, vertexPointB);
      fj = ComputeProbabilityForSingleTrack(linej, vertexPointB);
      probaB = fi*fj;
      slope = (probaB - probaA)/(b-a);
      if (slope > 0)
         a = (a+b)/2.;
      else
         b = (a+b)/2.;
   }

   //save the values into a structure
   fProbValuesMax[i*fTracksN + j] = (probaA+probaB)/2.;
   fRValuesMax[i*fTracksN + j]    = (vertexPointA+vertexPointB)*0.5;
}
```

➡ gain of 1.5 in CPU time and gain a factor ~3 in resolution

VTX vertexing (ii)

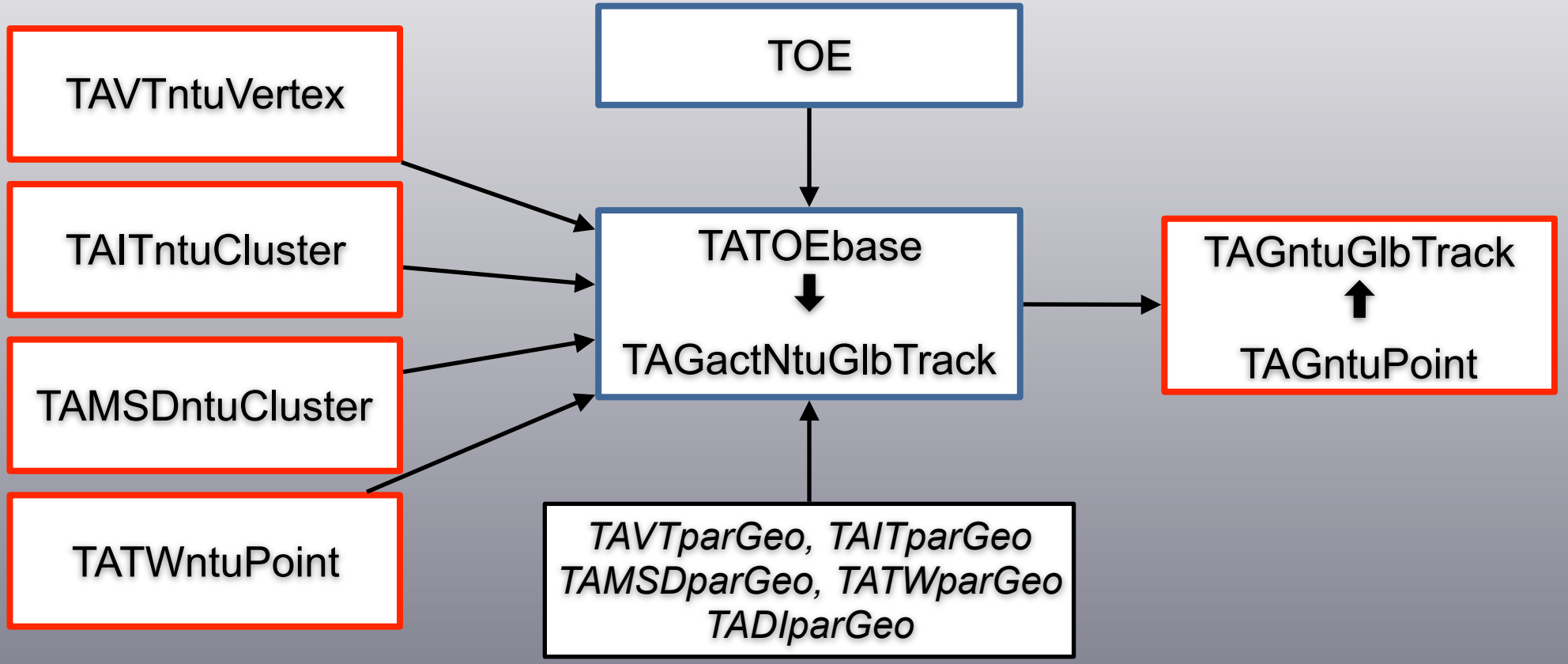
- TAVTactBaseNtuVertex: don’t take closest vertex with BM track

```
Bool_t TAVTactBaseNtuVertex::CheckBmMatching()
{
. . .
    Float_t min = config->GetAnalysisPar().PlanesForTrackMinimum; // to be tuned (sigma ~ 200)
 . . .
   for (Int_t i = 0; i < pNtuVertex->GetVertexN(); ++i) {
      TAVTvertex* vtx = pNtuVertex->GetVertex(i);

      TVector3 vtxPosition = vtx->GetVertexPosition();
      vtxPosition          = fpFootGeo->FromVTLocalToGlobal(vtxPosition);
      TVector3 bmPosition  = fpFootGeo->FromGlobalToBMLocal(vtxPosition);

      bmPosition   = bmTrack->PointAtLocalZ(bmPosition.Z());
      bmPosition   = fpFootGeo->FromBMLocalToGlobal(bmPosition);
      TVector3 res = vtxPosition - bmPosition;

      if (res.Perp() < min) {
        vtx->SetBmMatched();

      . . .
      }
   }
   return true;
}
```

➡ take vertex within a given distance with BM track (includes scattered tracks)

Global Reconstruction Action (i)

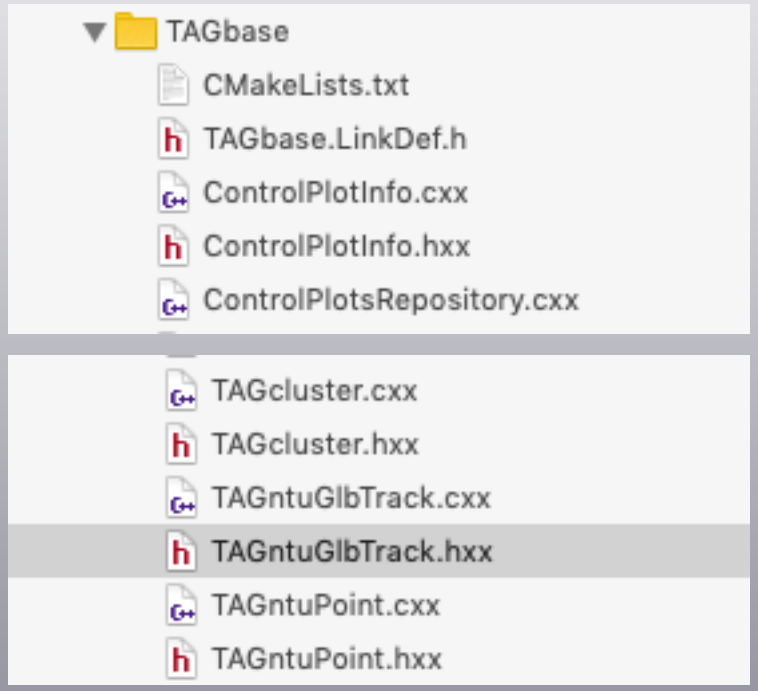
- Scheme:



- Containers as input from VTX, IT, MSD and TW
- Container as output global tracks with global points
- TOEbase: interface between TOE and root

Global Reconstruction Action (ii)

- TAGntuGlbTrack:



➡ Put TAGntuGlbTrack and TAGntuPoint is TAGbase folder cos needed by TATOEbase

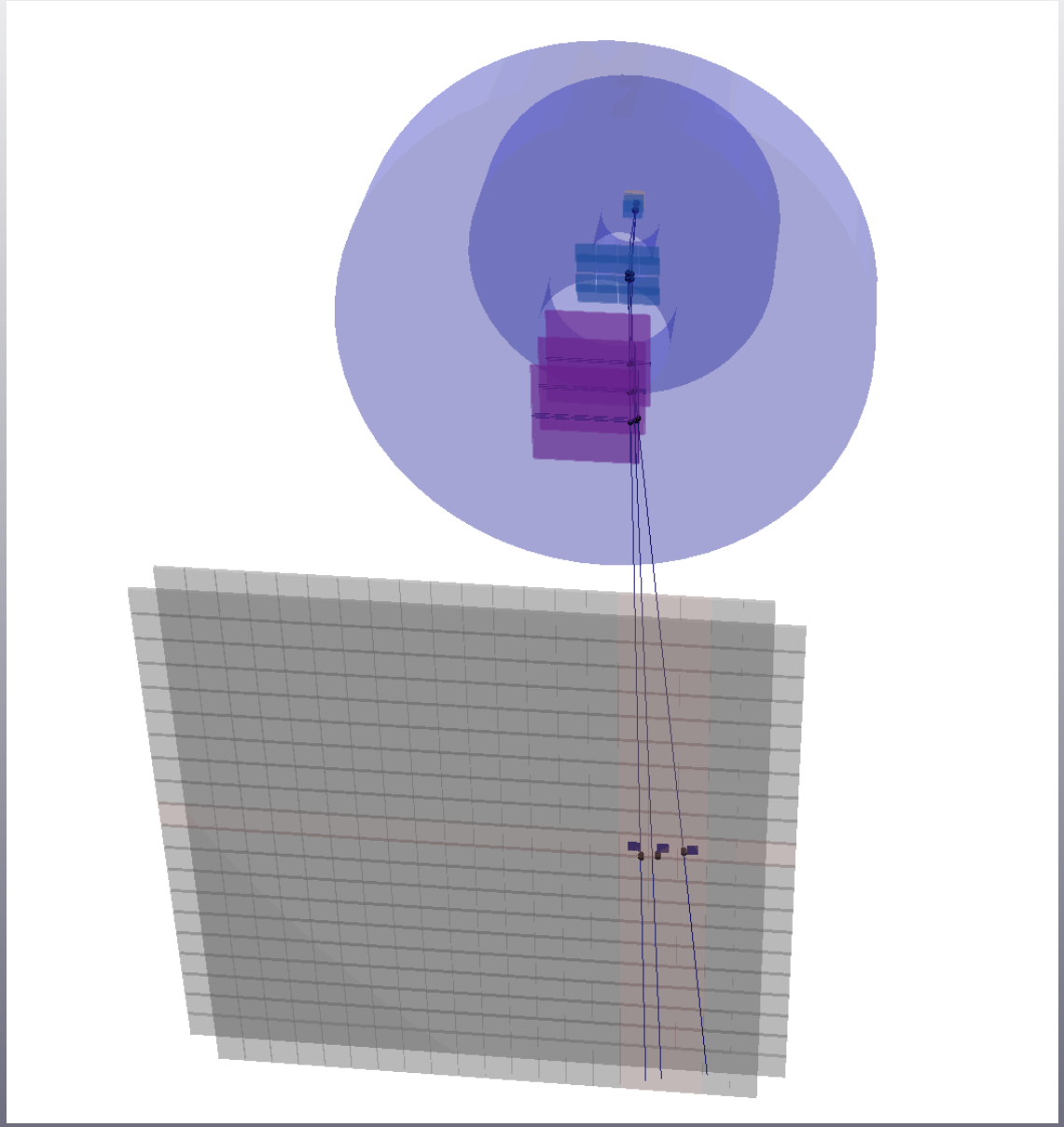
Global Reconstruction Action (iii)

- TAGpoint: add MC track id containers

```
cclass TAGpoint : public TAGobject {

private:
   TVector3    fPosition;      // position in FOOT framework
   TVector3    fPosError;      // position error in FOOT framework
   TVector3    fMomentum;      // momentum in FOOT framework
   TVector3    fMomError;      // momentum error in FOOT framework
   Int_t       fChargeZ;       // Charge Z

   TArrayI            fMcTrackIdx;              // Idx of the track created in the simulation
   std::map<int, int> fMcTrackMap;              //! Map of MC track Id

public:
   TAGpoint();
   TAGpoint(TVector3 pos, TVector3 posErr);
   TAGpoint(TVector3 pos, TVector3 posErr, TVector3 mom, TVector3 momErr, Int_t chargeZ = 0);
   ~TAGpoint() {};

. . .
   ClassDef(TAGtrack,2)
};
```

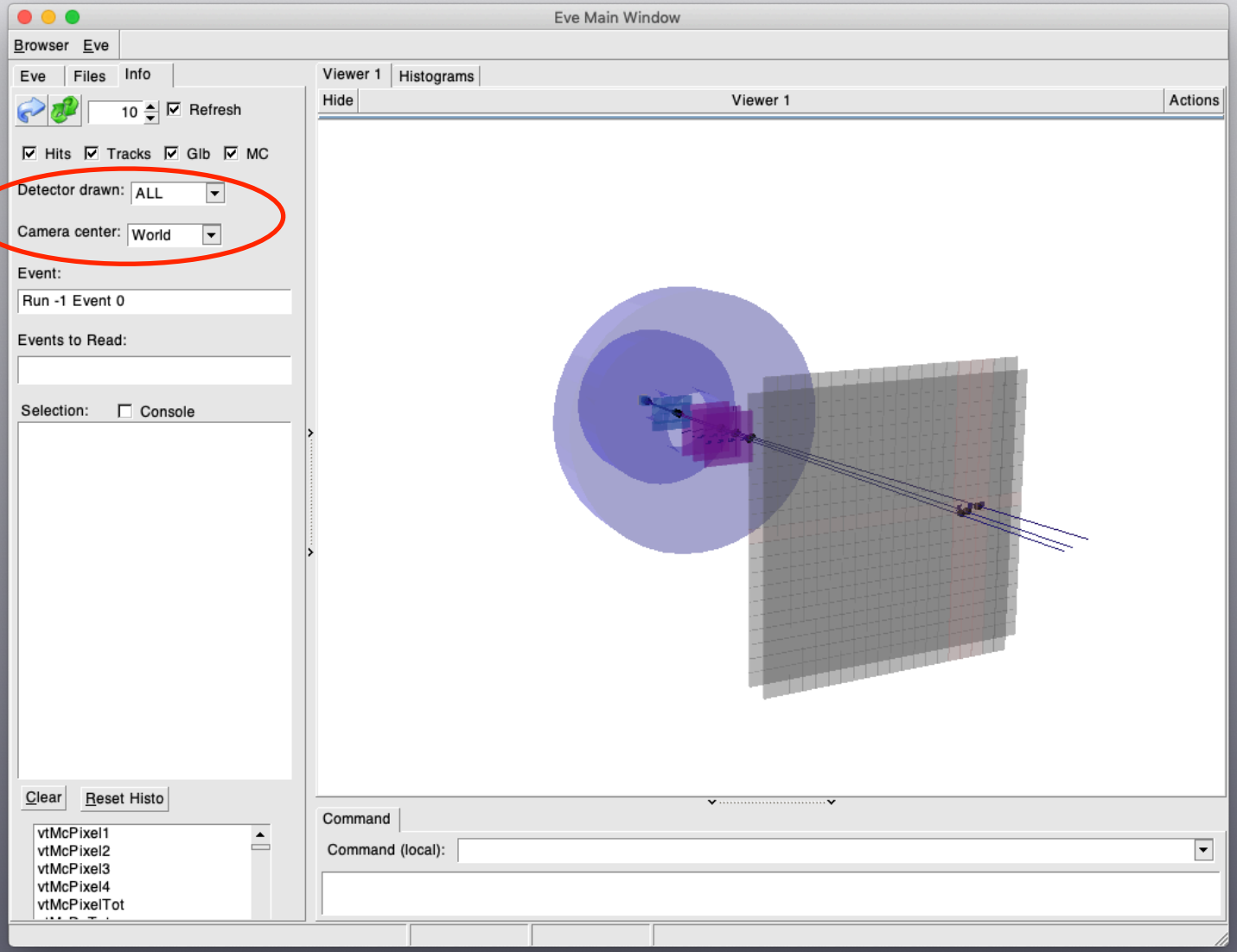
Global Reconstruction Action (iv)

- TOE action: results
  - ➡ Full reconstruction with TOE
  - ➡ No use of MC info (excp. TW)
  - ➡ More in Alexandre's talk



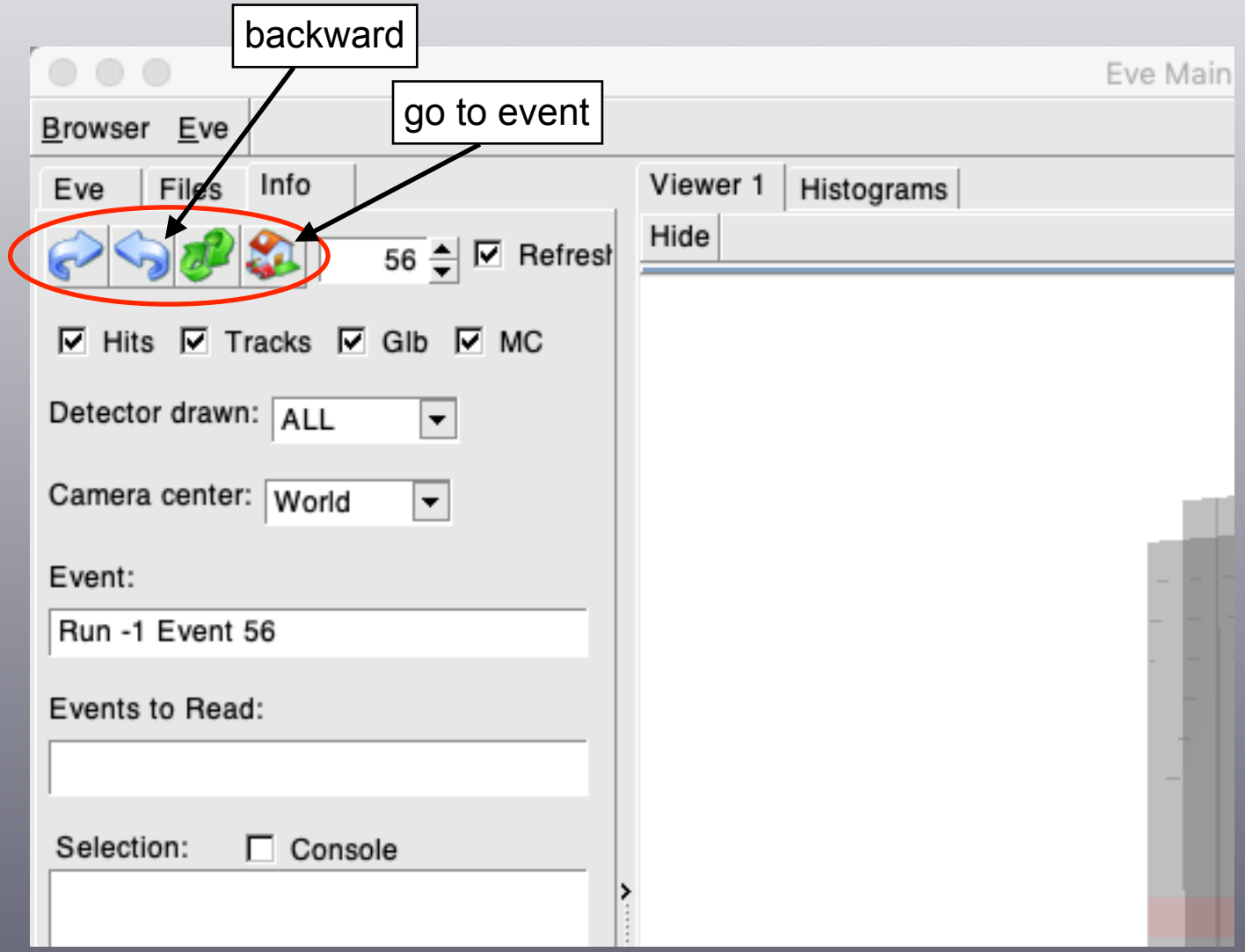
Event Display (i)

- Add menu box to set camera center



Event Display (ii)

- Add button for back navigation and for going to a given event (only MC)



Dictionary

- Remove dictionary generation for digitizer

```
#pragma link C++ class TAVTactVmeWriter+;
#pragma link C++ class TAVTactStdRaw+;
#pragma link C++ class TAVTactStdDaqRaw+;
#pragma link C++ class TAVTactNtuRaw+;

#pragma link C++ class TAVTactBaseNtuMC+;
#pragma link C++ class TAVTactNtuMC+;

#pragma link C++ class TAVTbaseDigitizer+;
#pragma link C++ class TAVTdigitizerE+;
#pragma link C++ class TAVTdigitizerG+;

#pragma link C++ class TAVTntuCluster+;
#pragma link C++ class TAVTcluster+;
#pragma link C++ class TAVTbaseCluster+;
```

```
#pragma link C++ class TAVTactVmeWriter+;
#pragma link C++ class TAVTactStdRaw+;
#pragma link C++ class TAVTactStdDaqRaw+;
#pragma link C++ class TAVTactNtuRaw+;

#pragma link C++ class TAVTactBaseNtuMC+;
#pragma link C++ class TAVTactNtuMC+;

#pragma link C++ class TAVTntuCluster+;
#pragma link C++ class TAVTcluster+;
#pragma link C++ class TAVTbaseCluster+;
```

➡ Done for all other detectors

Optimization/structure

- Following basic C++ rules

```
class TATWdatRaw : public TAGdata {
public:

  TATWdatRaw();
  virtual          ~TATWdatRaw();
  Int_t              GetHitsN() const;
  TATWrawHit*        GetHit(Int_t i_ind);
  const TATWrawHit* GetHit(Int_t i_ind) const;
  void        NewHit(TWaveformContainer *w);
. . .
private:
  static TString fgkBranchName;    // Branch name in TTree
  Int_t           fHitsN;           //
  TClonesArray*   fListOfHits;
  Int_t           fRunTime;
};
```

- Put data members in private, disentangle class members, homogeneity for setter/ getter name, etc…:

➡ Done for GlobalPar, TWdatRaw, TABMdatRaw, TABMntuRaw, TABMntuTrackTr, TACAdatRaw, TASTdatRaw, TABMactNtuTrack, TABMntuHit, TABMparMap, TASTntuRaw, TWaveformContainer, TATWparGeo, TAGbaseWD and TABMparCon (hope this is done, since I've been confined, I had some time to spend on…)

Conclusion

- Optimized interfaces with global reconstruction
- Proper handling of particle pileup for all detectors
- Some optimizations, some minor fix

➡ First time that we can debug the whole scheme of FOOT in new_geom branch (also in master branch thanks to Alessio)

➡ More about TOE in Alexandre presentation