sPHENIX offline software など

蜂谷　崇

概要

目的：

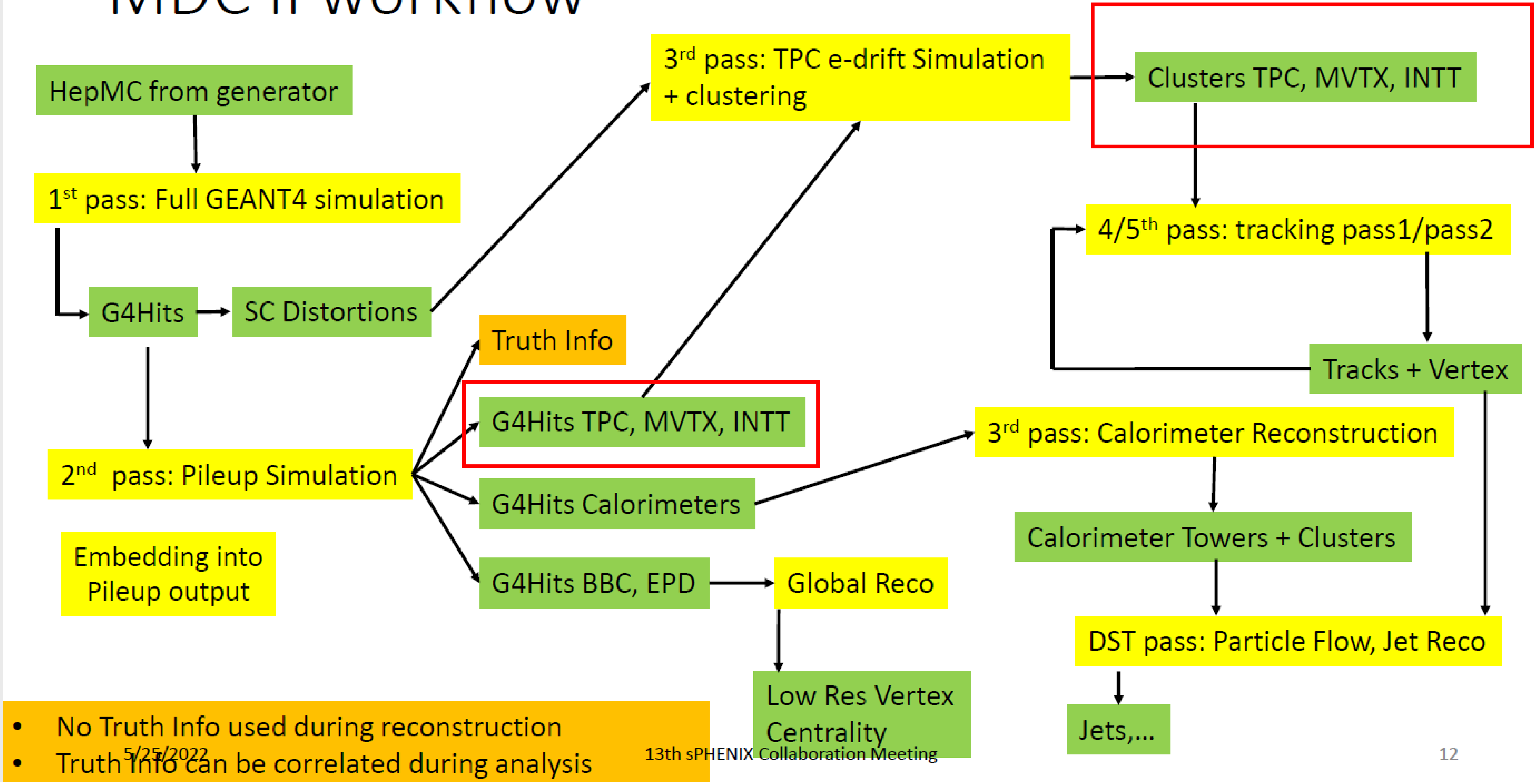
・　sPHENIX reconstruction (Trackingなど）についての概要を知る

・　Joeたちとの打合せで質問することをまとめる

　木曜日9：00pm (8am US)

・　INTTグループがすることをまとめる。

MDC II workflow



HepMC from generator

1st pass: Full GEANT4 simulation

G4Hits

SC Distortions

2nd pass: Pileup Simulation

Embedding into Pileup output

Truth Info

G4Hits TPC, MVTX, INTT

G4Hits Calorimeters

G4Hits BBC, EPD

3rd pass: TPC e-drift Simulation + clustering

Clusters TPC, MVTX, INTT

4/5th pass: tracking pass1/pass2

Tracks + Vertex

3rd pass: Calorimeter Reconstruction

Calorimeter Towers + Clusters

DST pass: Particle Flow, Jet Reco

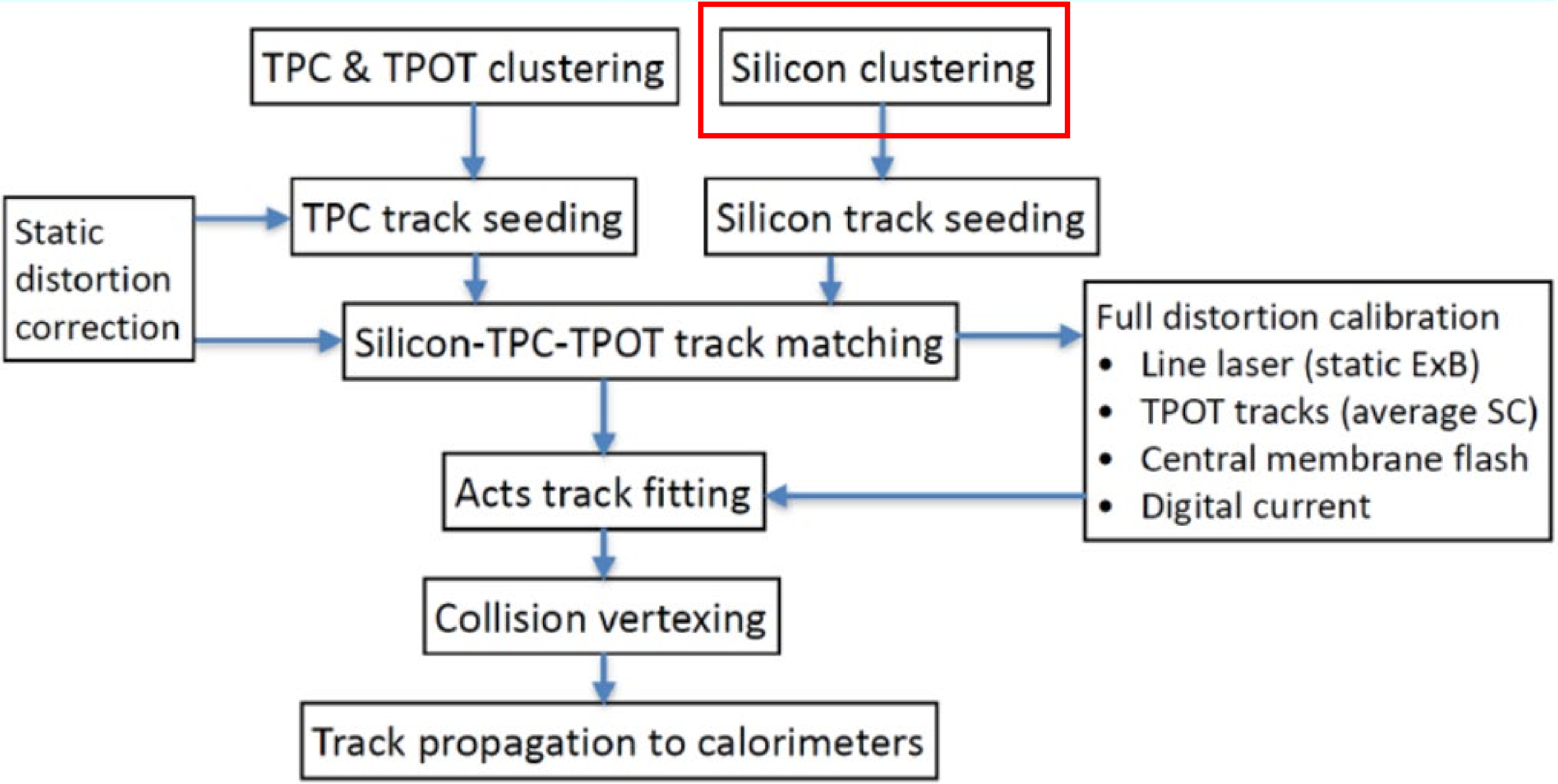
Jets,...

Global Reco

Low Res Vertex Centrality

- No Truth Info used during reconstruction
- Truth Info can be correlated during analysis

Simplified overview of tracking workflow



TPC & TPOT clustering

Silicon clustering

Static distortion correction

TPC track seeding

Silicon track seeding

Silicon-TPC-TPOT track matching

Full distortion calibration

- Line laser (static ExB)
- TPOT tracks (average SC)
- Central membrane flash
- Digital current

Acts track fitting

Collision vertexing

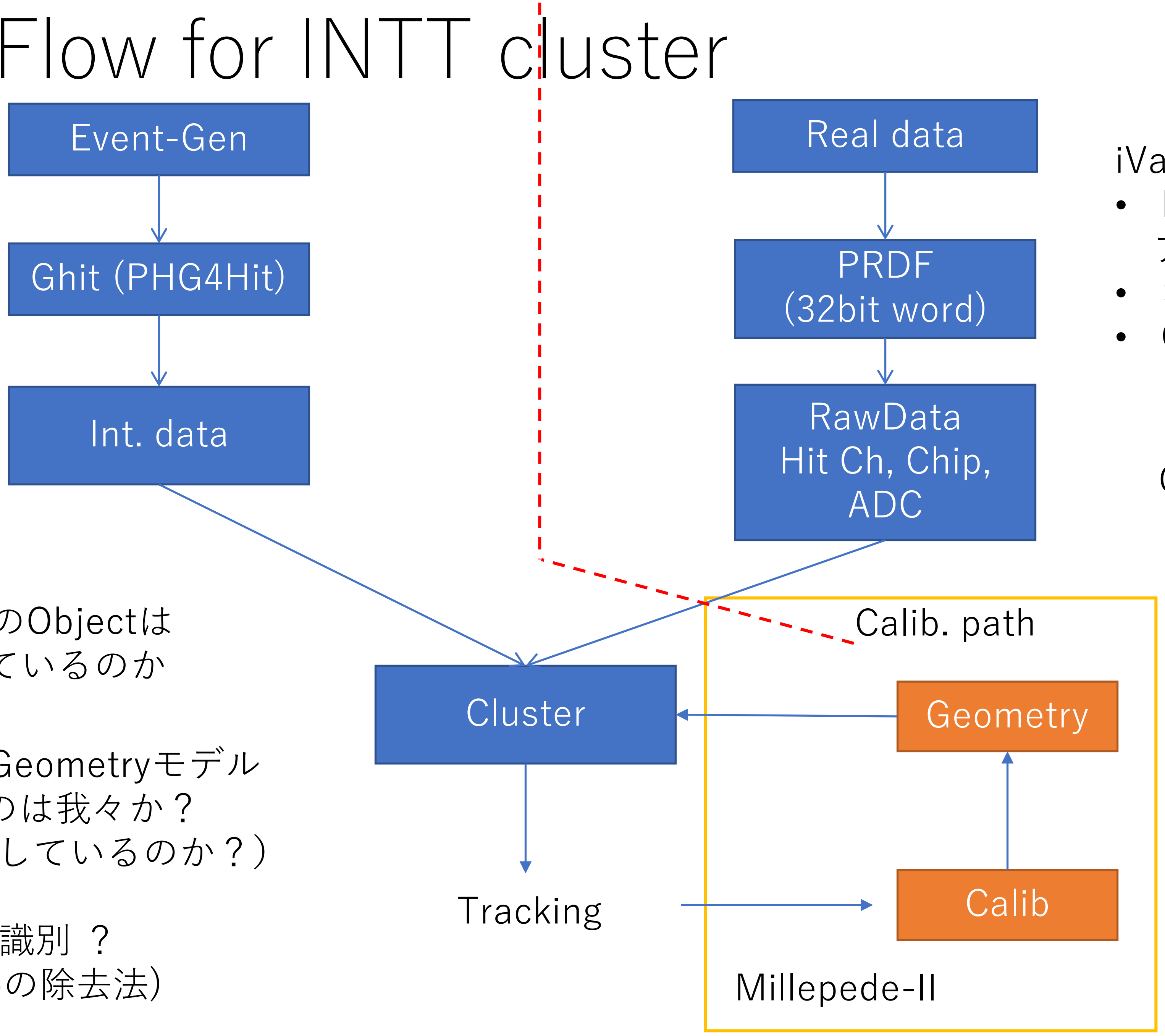
Track propagation to calorimeters

Data Flow for INTT cluster

Q1: ClusterのObjectは
　　決まっているのか

Q2: INTTのGeometryモデル
　　を作るのは我々か？
　（既に存在しているのか？）

Q6: Eventの識別 ？
　　(Pile upの除去法)



Event-Gen

Ghit (PHG4Hit)

Int. data

Real data

PRDF
(32bit word)

RawData
Hit Ch, Chip,
ADC

Cluster

Tracking

Calib. path

Geometry

Calib

Millepede-II

iValue() for INTT

- Packetのデータ
  フォーマットが不明 → Martin
- オンラインモニタ
- QAプロット

Q5: ドキュメントの管理は？
　・既存の置き場所
　・これまでのDOCを参考
　・もしないなら問題提起

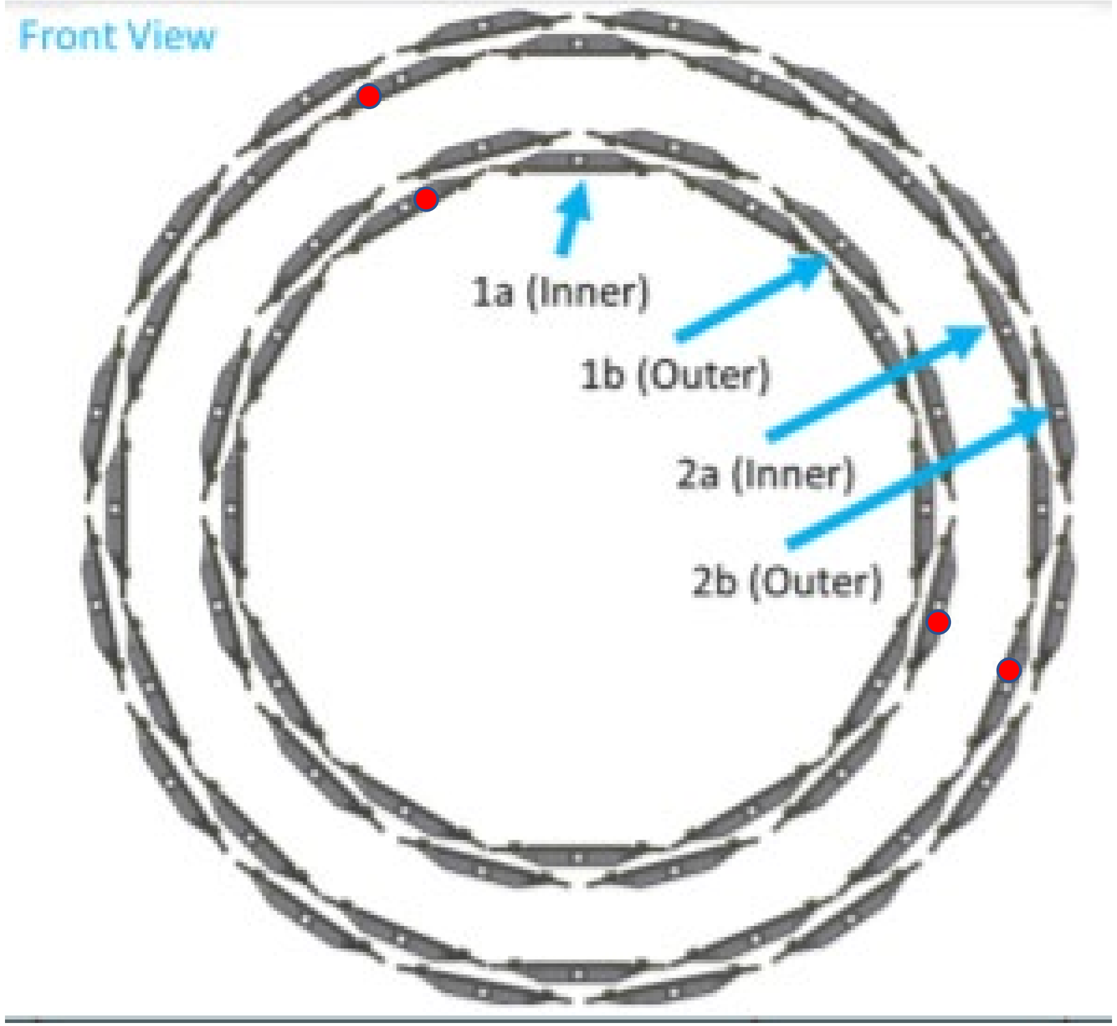
Q4: DBへの
　　インターフェースは？

Q3: Millepedeへの
　　インターフェースは？

Event Display

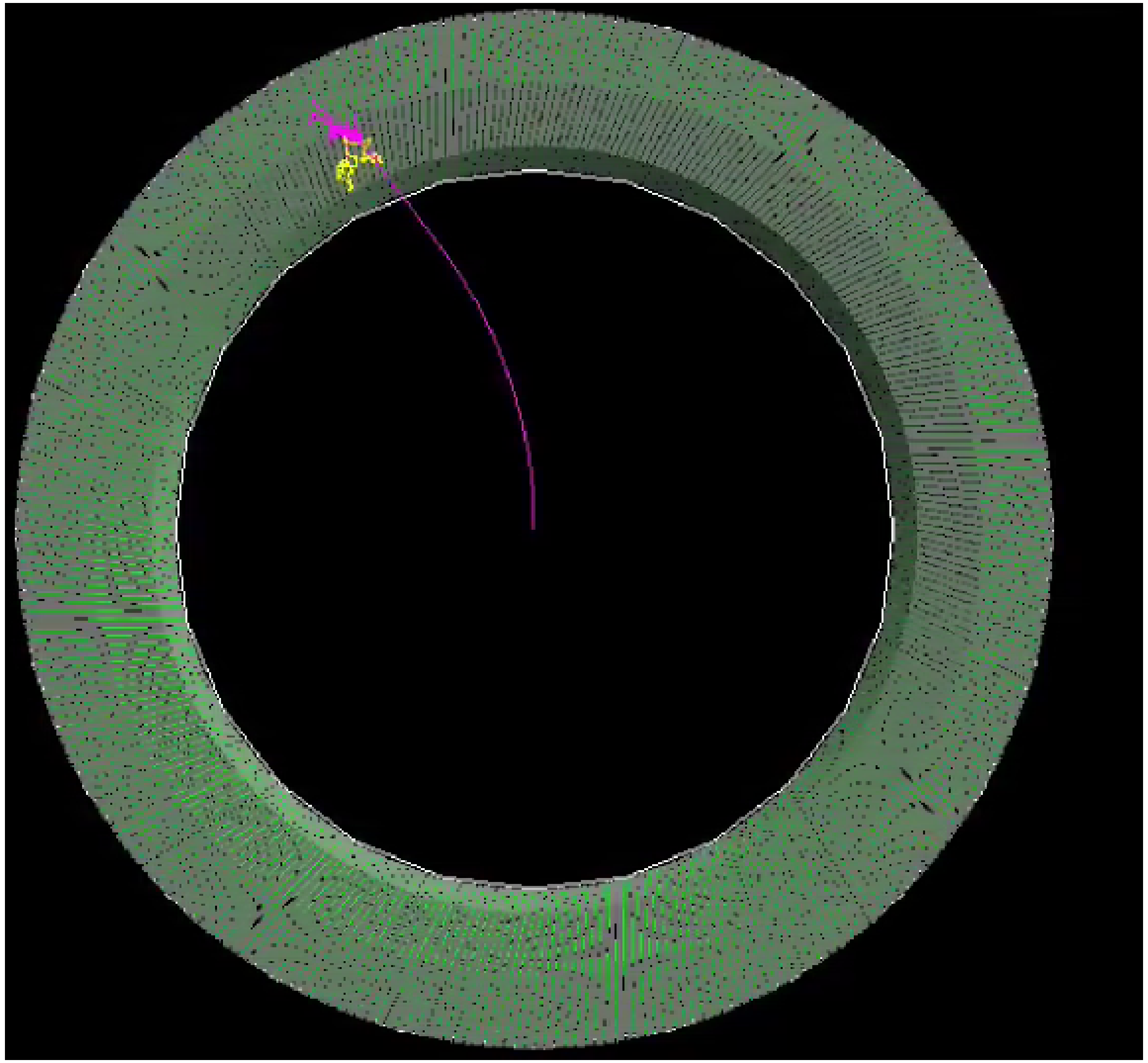
- Event by Event のヒット位置（X-Y, R-Z）（2D）
- 3D モデルでもOK

- MVTXと組み合わせるなど



If everything went well you should see:

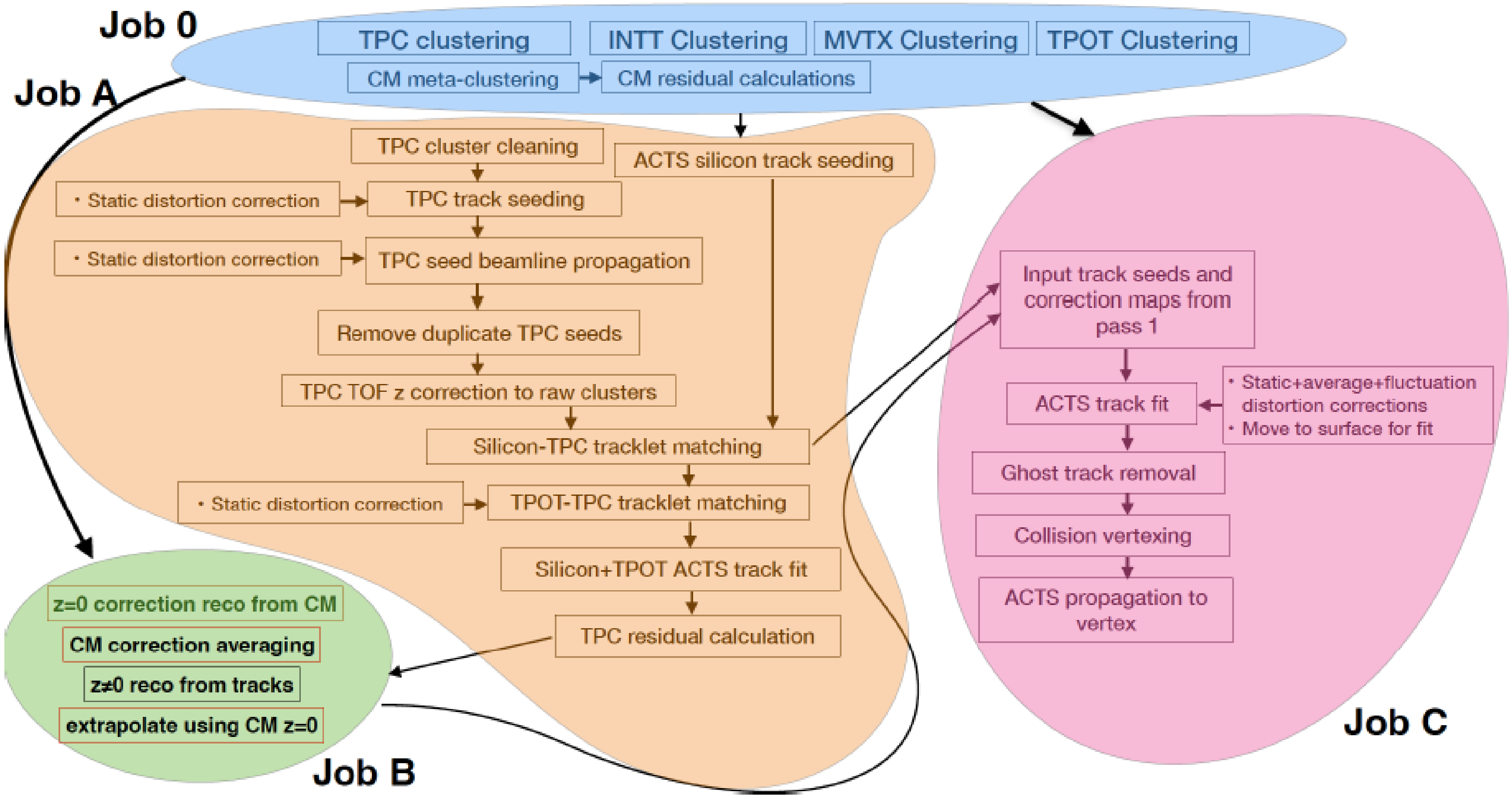
A pi- coming from the center of the ohcal with px = 0; py = 1; pz = 0; Using a particle gun



Tracking and calibration workflow

- We use ACTS package for tracking
  - Flexible geometry description, Kalman filter.
- Job 0: Hit clustering and central membrane flash reconstruction.
- Job A: Track seeding, track assembly, track-based distortion calibration.
- Job B: Calculation of distortion correction maps.
- Job C: Final track fitting with distortion corrections.

Workflow for triggered mode



Job 0

TPC clustering

INTT Clustering

MVTX Clustering

TPOT Clustering

CM meta-clustering → CM residual calculations

Job A

TPC cluster cleaning

ACTS silicon track seeding

• Static distortion correction → TPC track seeding

• Static distortion correction → TPC seed beamline propagation

Remove duplicate TPC seeds

TPC TOF z correction to raw clusters

Silicon-TPC tracklet matching

• Static distortion correction → TPOT-TPC tracklet matching

Silicon+TPOT ACTS track fit

TPC residual calculation

Job B

z=0 correction reco from CM

CM correction averaging

z≠0 reco from tracks

extrapolate using CM z=0

Job C

Input track seeds and correction maps from pass 1

ACTS track fit

• Static+average+fluctuation distortion corrections
• Move to surface for fit

Ghost track removal

Collision vertexing

ACTS propagation to vertex

Shown at S&C

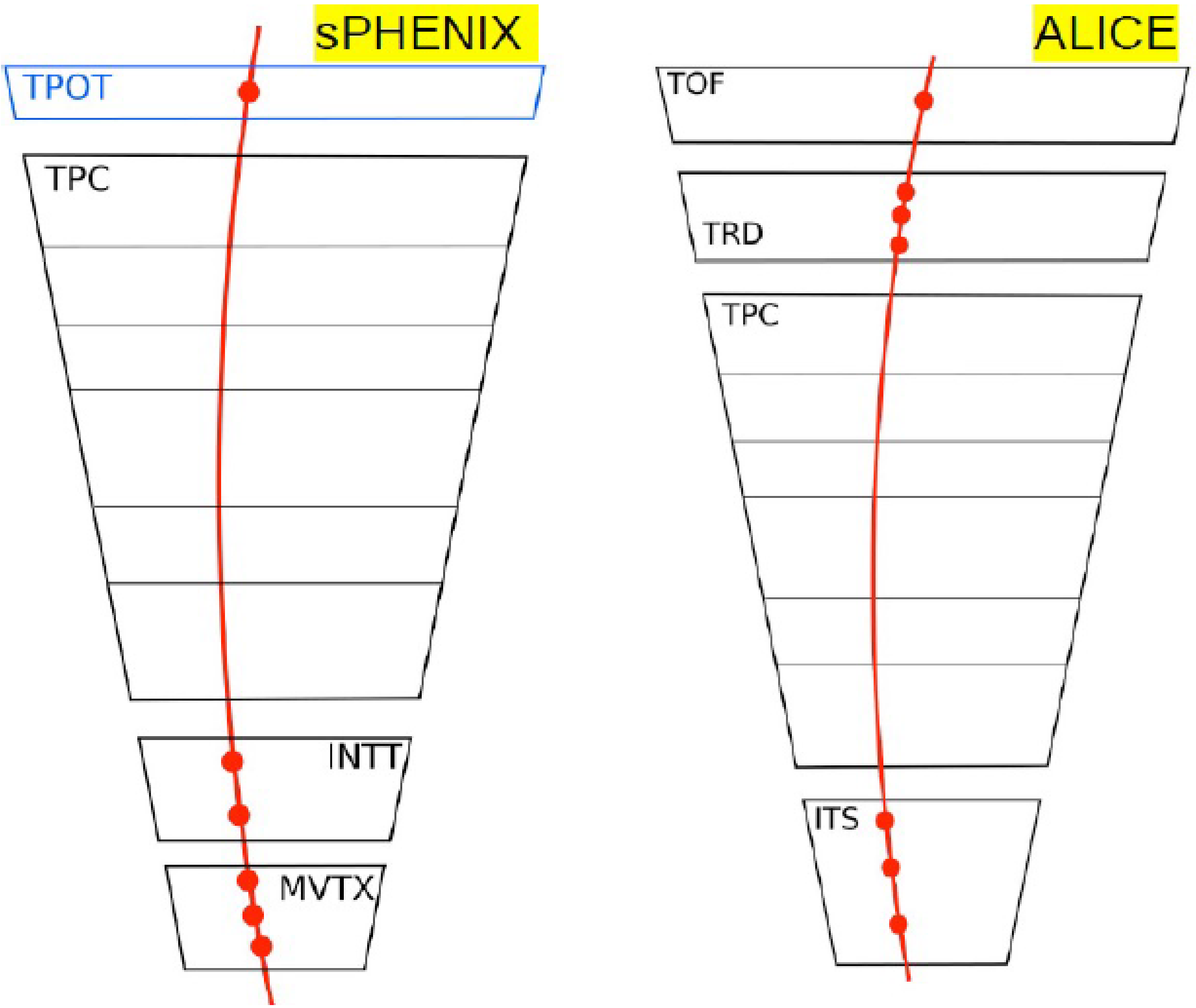
Keys to good tracking

- Accurate hit position and good knowledge of magnetic field will lead to good tracking/momentum resolution
  - There are many factors that distort them.

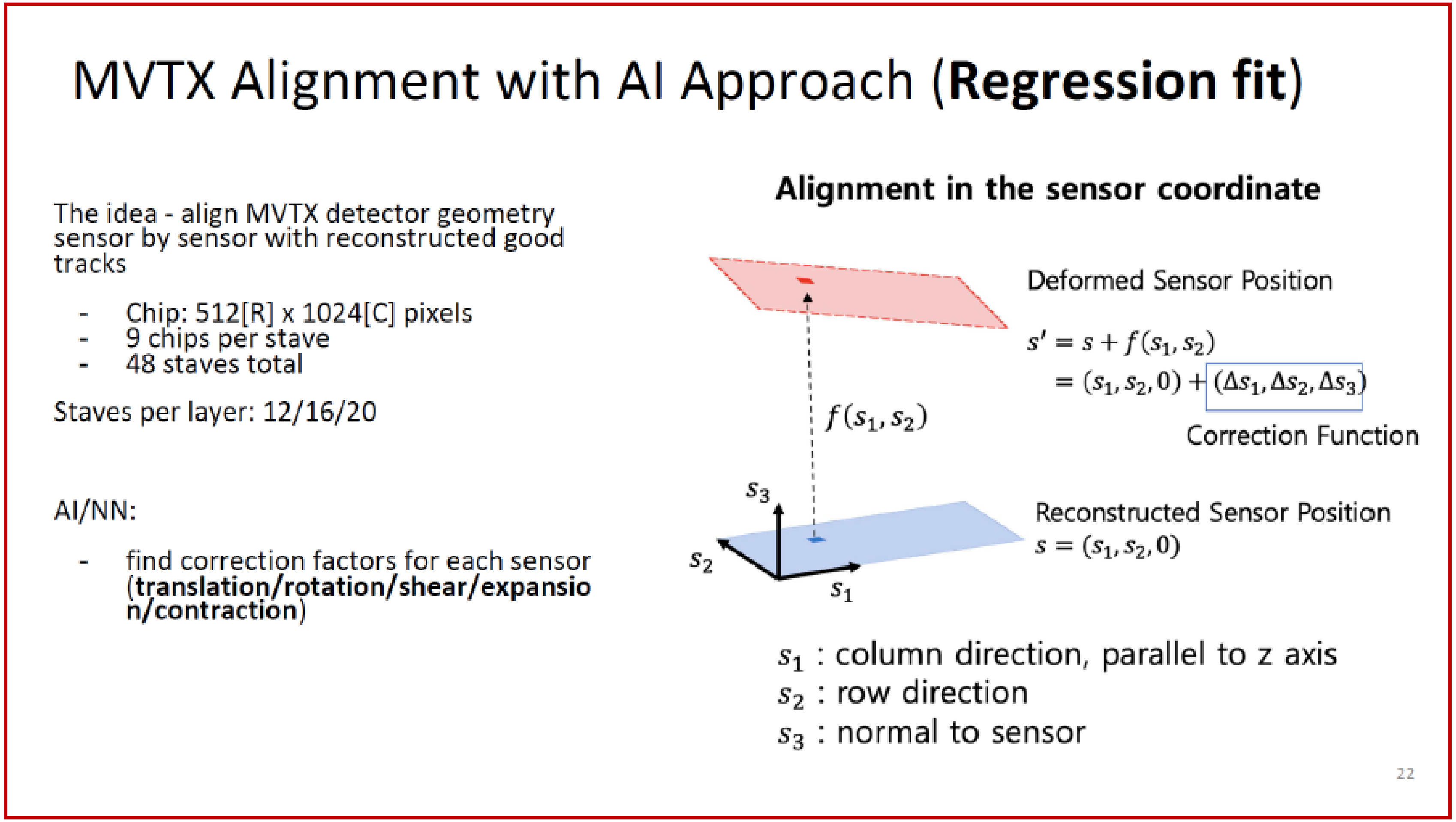
- TPC is the main tracker in sPHENIX.
  - Gateless TPC results in distortion of electric field in TPC
  - More detail in the next slide.

- Goal of 125MeV/c^2 mass resolution of Upsilon will need ~100-150um hit position resolution in TPC

- MVTX/INTT/TPC, and newly introduced TPOT will play for correcting distortion/alignment
  - ALICE has detectors at outer TPC, but sPHENIX didn't have (EMCal has poorer position resolution)



Machine-learning for MVTX (and INTT?) alignment



MVTX Alignment with AI Approach (**Regression fit**)

The idea - align MVTX detector geometry sensor by sensor with reconstructed good tracks

- Chip: 512[R] x 1024[C] pixels
- 9 chips per stave
- 48 staves total

Staves per layer: 12/16/20

AI/NN:

- find correction factors for each sensor (**translation/rotation/shear/expansion/contraction**)

Alignment in the sensor coordinate

Deformed Sensor Position

$$s' = s + f(s_1, s_2)$$
$$= (s_1, s_2, 0) + \boxed{(\Delta s_1, \Delta s_2, \Delta s_3)}$$

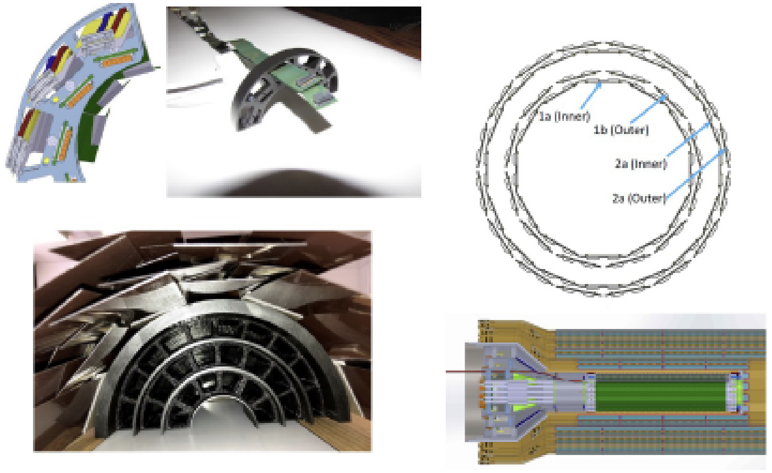
Correction Function

$f(s_1, s_2)$

s_3 s_2 s_1

Reconstructed Sensor Position
$s = (s_1, s_2, 0)$

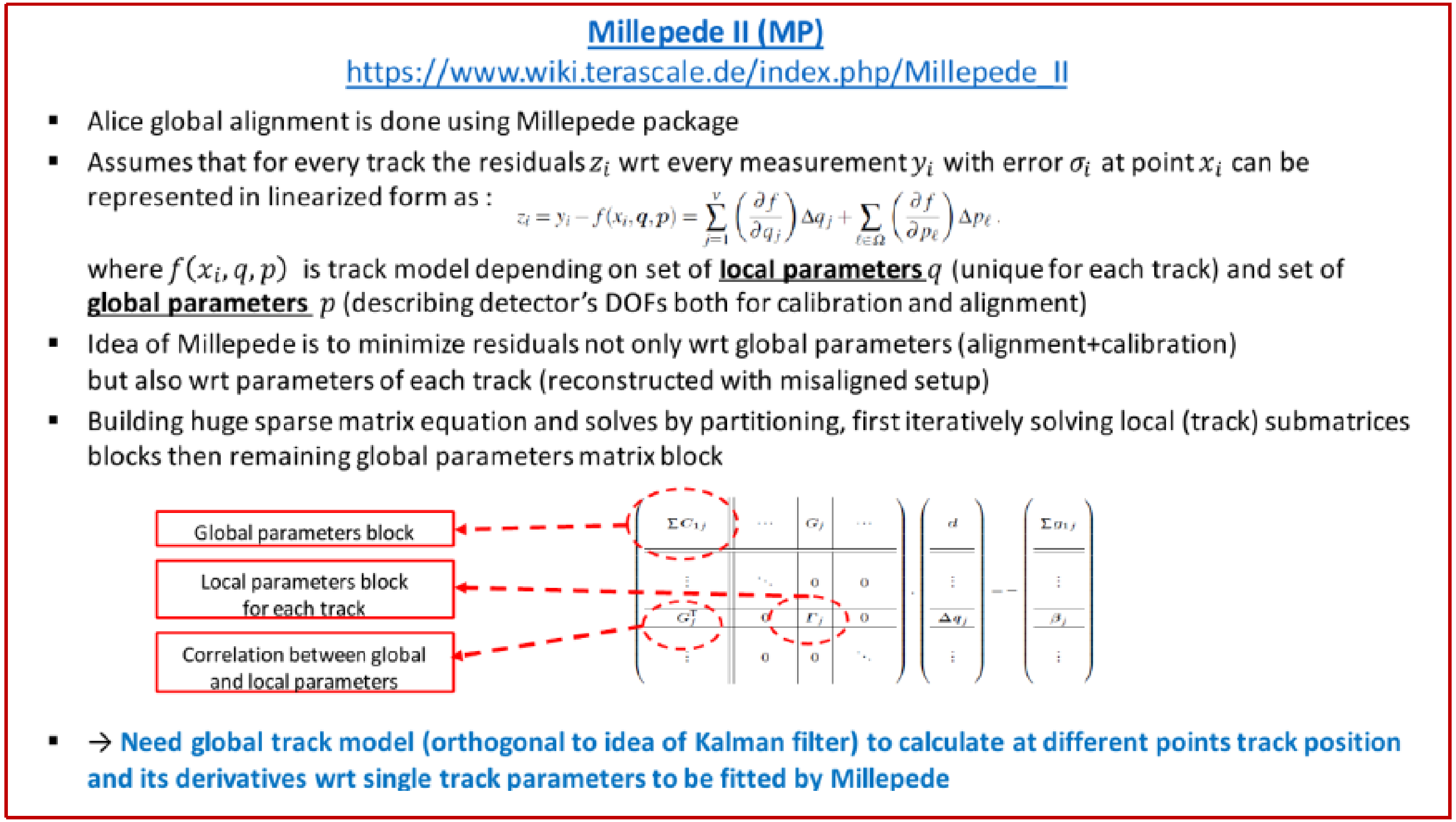
s_1 : column direction, parallel to z axis
s_2 : row direction
s_3 : normal to sensor



Ming on Monday

Global alignment

- Calorimeter people are also actively working on geometry/alignment correction procedure
  - Simulation, survey
  - Eventually, they need tracking too.
- Mellipede is an alignment package widely used, and may be used in sPHENIX
  - ACTs package has a detector alignment tool too, which is under investigation.
- Let N parameters float and fit tracks with M points. This works if M>N.
  - We can not only let geometry parameters float, but also detector-specific parameters float
  - Drift velocity, T0, etc.
- And, minimization….
  - Many algorithms…



Millepede II (MP)

https://www.wiki.terascale.de/index.php/Millepede_II

- Alice global alignment is done using Millepede package
- Assumes that for every track the residuals z_i wrt every measurement y_i with error σ_i at point x_i can be represented in linearized form as : $z_i = y_i - f(x_i, q, p) = \sum_{j=1}^{\nu} \left(\frac{\partial f}{\partial q_j}\right) \Delta q_j + \sum_{\ell \in \Omega} \left(\frac{\partial f}{\partial p_\ell}\right) \Delta p_\ell$.

  where $f(x_i, q, p)$ is track model depending on set of **local parameters** q (unique for each track) and set of **global parameters** p (describing detector's DOFs both for calibration and alignment)
- Idea of Millepede is to minimize residuals not only wrt global parameters (alignment+calibration) but also wrt parameters of each track (reconstructed with misaligned setup)
- Building huge sparse matrix equation and solves by partitioning, first iteratively solving local (track) submatrices blocks then remaining global parameters matrix block

$$\begin{pmatrix} \Sigma C_{1j} & \cdots & G_j & \cdots \\ \vdots & \ddots & 0 & 0 \\ G_j^T & 0 & \Gamma_j & 0 \\ \vdots & 0 & 0 & \ddots \end{pmatrix} \cdot \begin{pmatrix} a \\ \vdots \\ \Delta q_j \\ \vdots \end{pmatrix} = - \begin{pmatrix} \Sigma \alpha_{1j} \\ \vdots \\ \beta_j \\ \vdots \end{pmatrix}$$

Global parameters block

Local parameters block for each track

Correlation between global and local parameters

- → **Need global track model (orthogonal to idea of Kalman filter) to calculate at different points track position and its derivatives wrt single track parameters to be fitted by Millepede**

Jens W, software Mega-workfest (2020)

INTT module in GEANT4 macro

```
// Initialize the selected subsystems
 G4Init();  // if (Enable::INTT) InttInit(); // set parameters to BlackHole parameters

 //---------------------
 // GEANT4 Detector description
 if (!Input::READHITS) G4Setup();    //  if (Enable::INTT) radius = Intt(g4Reco, radius);
 //                                          //   PHG4InttSubsystem* sitrack = new PHG4InttSubsystem("INTT", vpair);
 //------------------
 // Detector Division
 if (Enable::INTT_CELL) Intt_Cells();
 //  if (G4INTT::InttDeadMapOption != G4INTT::kInttNoDeadMap)
 //                  PHG4InttDeadMapLoader* deadMapINTT = new PHG4InttDeadMapLoader("INTT");
 //
 //  PHG4InttHitReco* reco = new PHG4InttHitReco();          //  // new storage containers
 //  PHG4InttDigitizer* digiintt = new PHG4InttDigitizer(); //  // new containers

 //--------------
 // SVTX tracking
if(Enable::TRACKING_TRACK) TrackingInit();
if(Enable::INTT_CLUSTER)   Intt_Clustering();   //  InttClusterizer* inttclusterizer = new InttClusterizer("InttClusterizer",
                                                //    G4MVTX::n_maps_layer, G4MVTX::n_maps_layer + G4INTT::n_intt_layer - 1);
if(Enable::TRACKING_TRACK) Tracking_Reco();
```

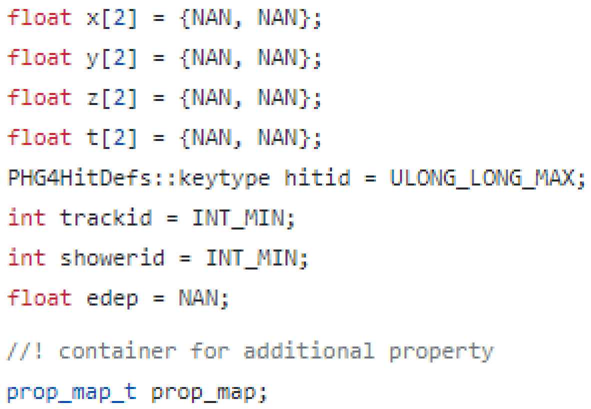
- PHG4InttSubsystem
  - g4_hits (PHG4HitContainer)
    - PHG4Hitv1

- PHG4InttHitReco
  - input: g4_hits (PHG4HitContainer)
  - Output: hitsetcontainer (TrkrHitSetContainerv1)

- PHG4InttDigitizer

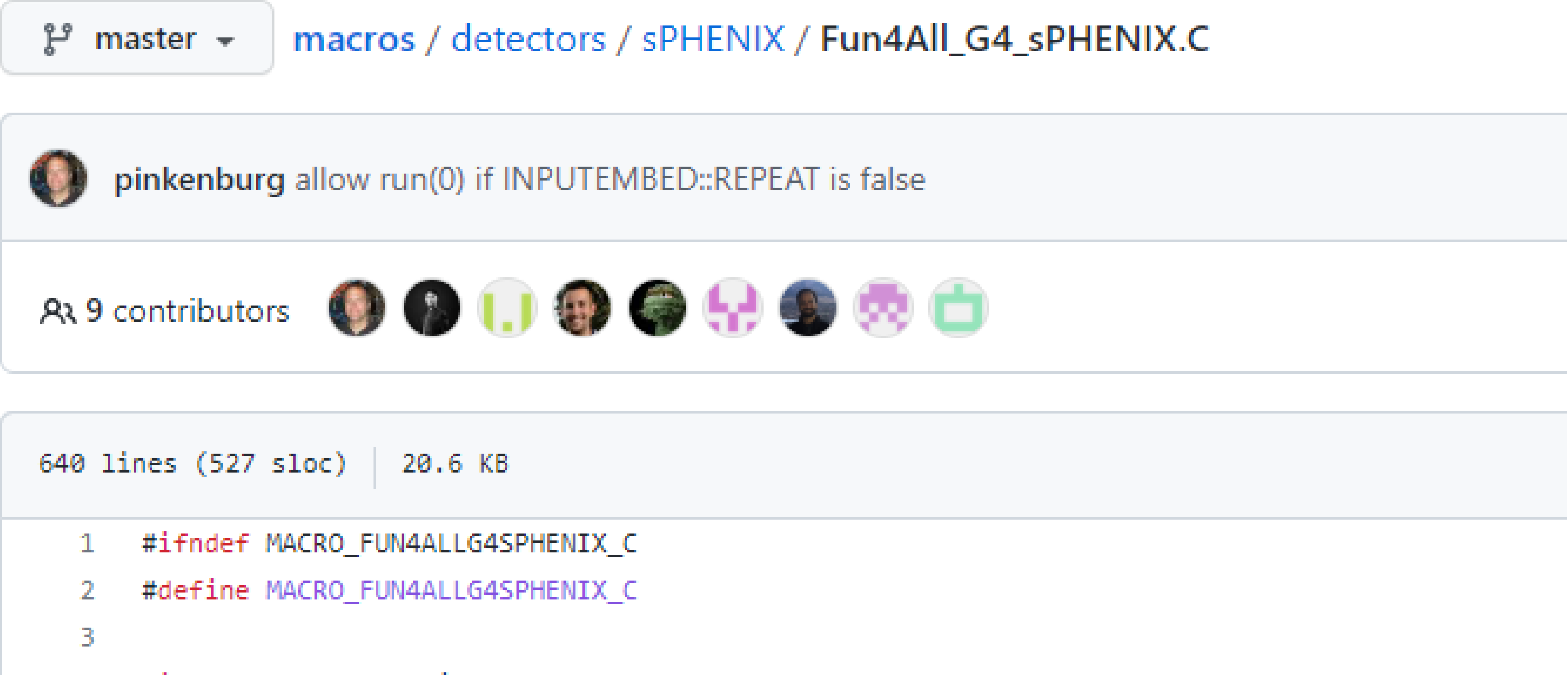
- InttClusterizer

PHG4Hitv1



```
float x[2] = {NAN, NAN};
float y[2] = {NAN, NAN};
float z[2] = {NAN, NAN};
float t[2] = {NAN, NAN};
PHG4HitDefs::keytype hitid = ULONG_LONG_MAX;
int trackid = INT_MIN;
int showerid = INT_MIN;
float edep = NAN;

//! container for additional property
prop_map_t prop_map;
```

Data format in sPHENIX-GEANT4



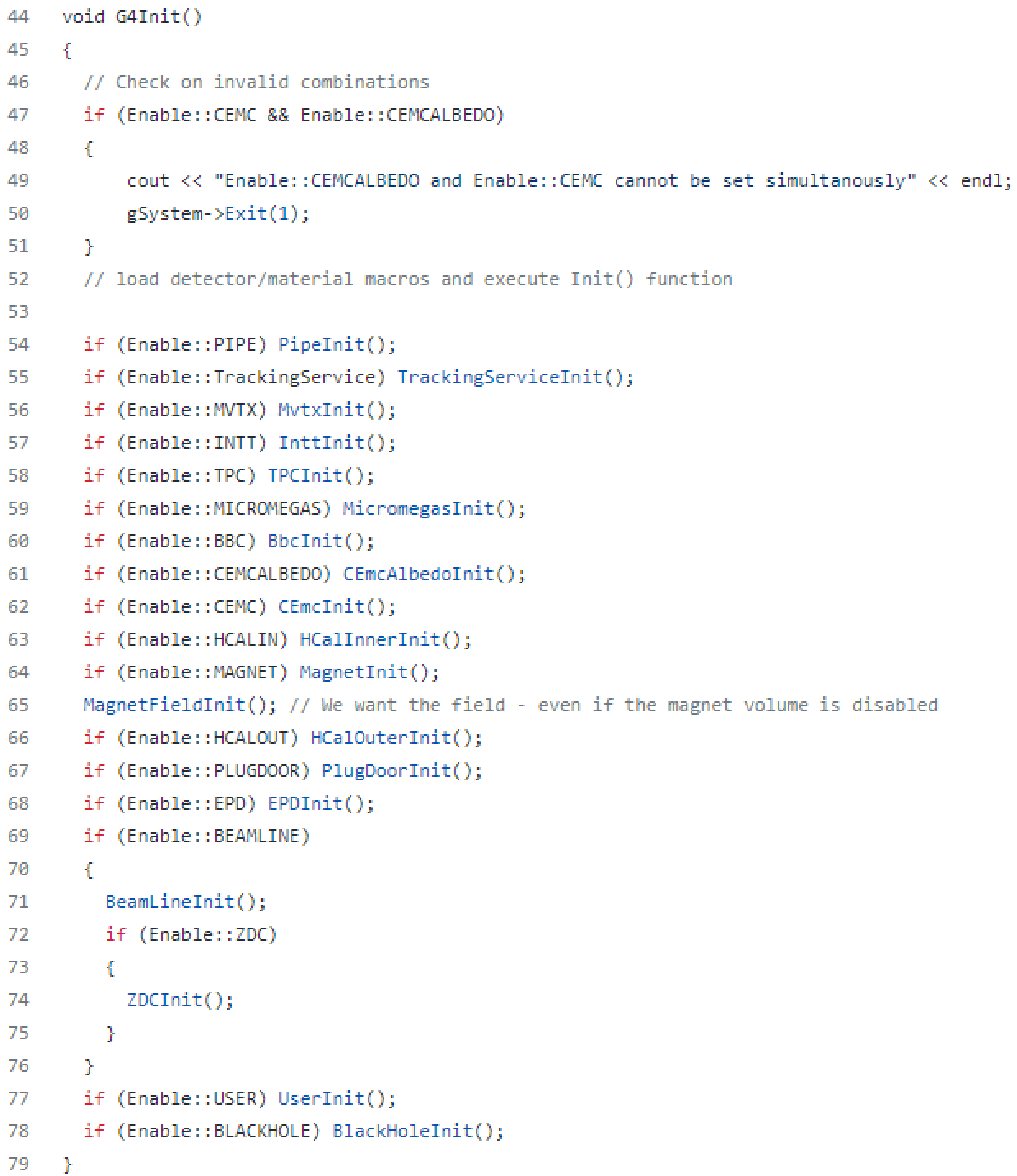
macros / detectors / sPHENIX / Fun4All_G4_sPHENIX.C

pinkenburg allow run(0) if INPUTEMBED::REPEAT is false

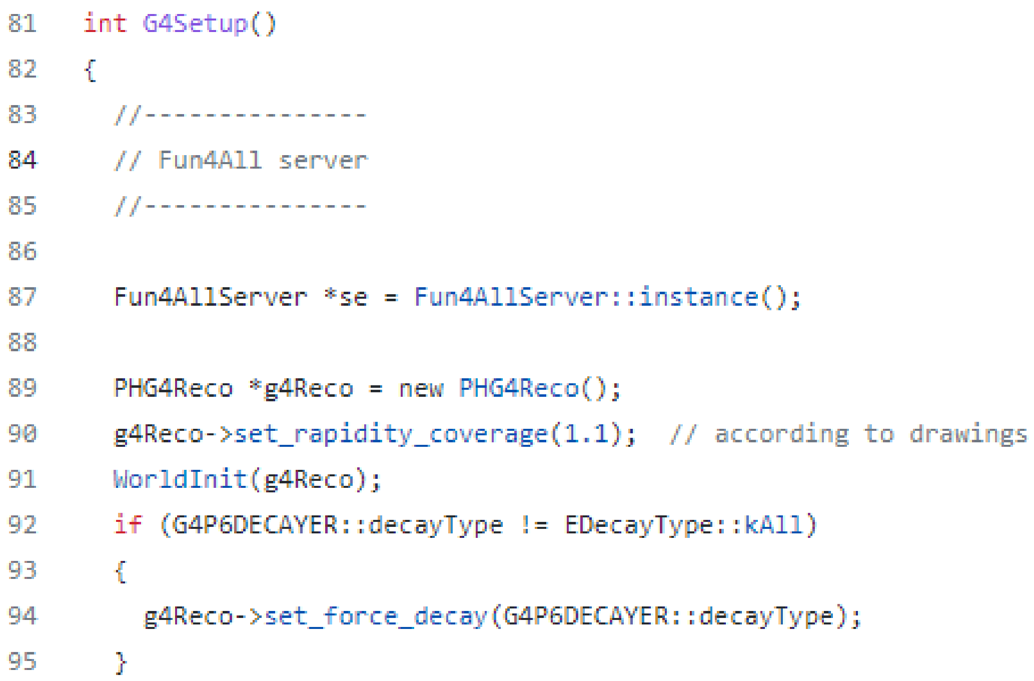
9 contributors

640 lines (527 sloc) | 20.6 KB

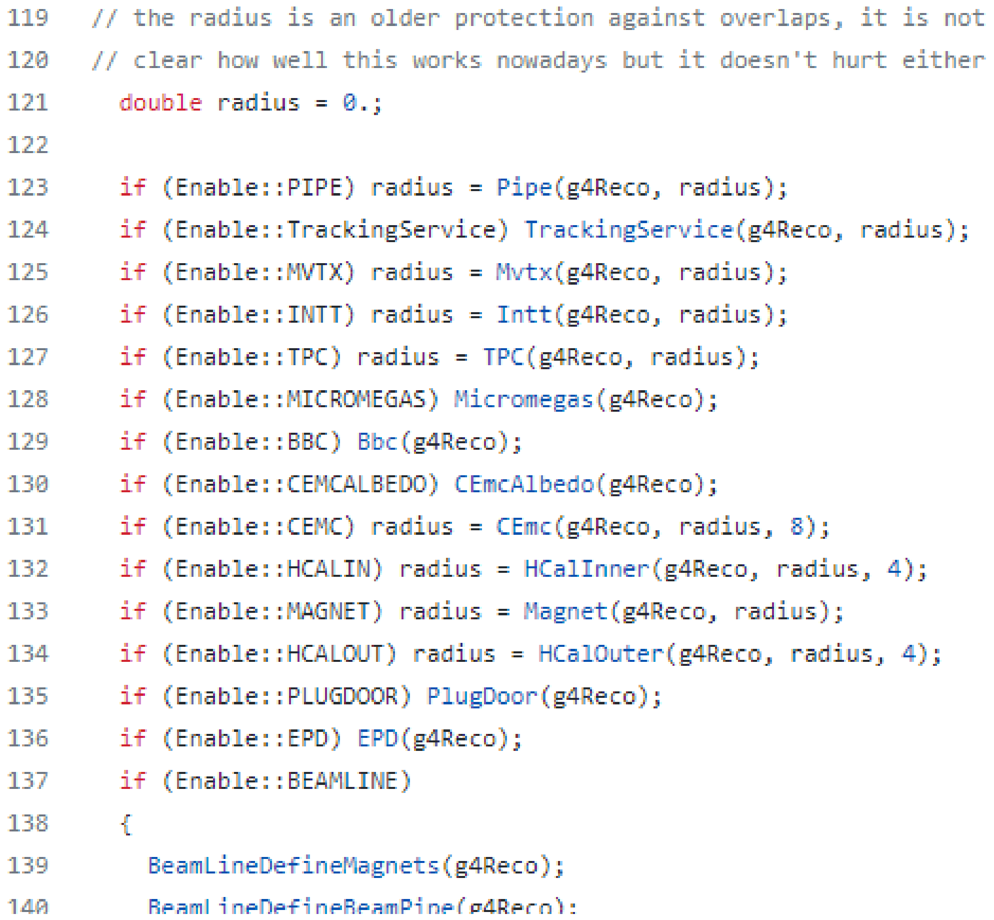
```
#ifndef MACRO_FUN4ALLG4SPHENIX_C
#define MACRO_FUN4ALLG4SPHENIX_C

```



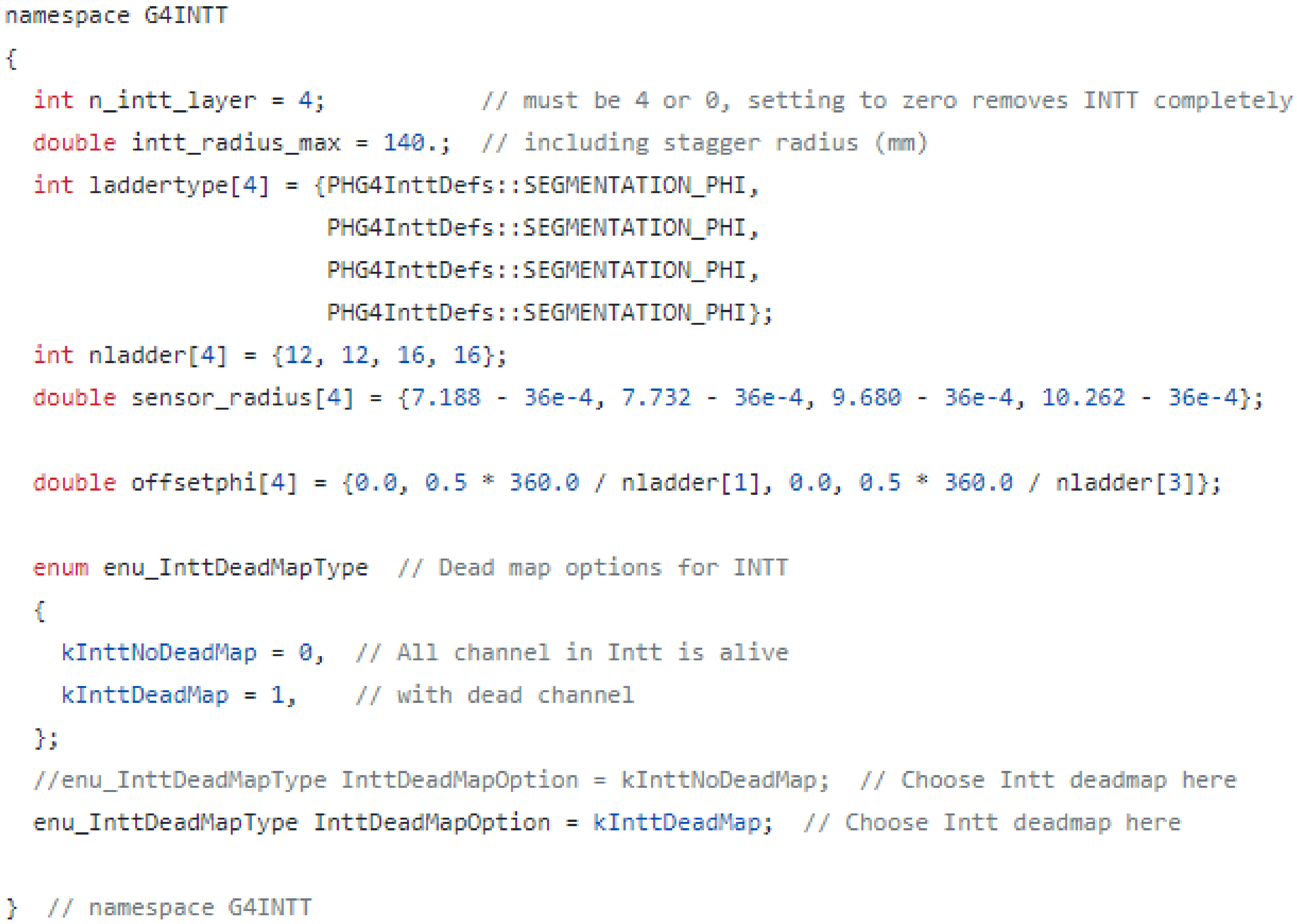
```
void G4Init()
{
  // Check on invalid combinations
  if (Enable::CEMC && Enable::CEMCALBEDO)
  {
      cout << "Enable::CEMCALBEDO and Enable::CEMC cannot be set simultanously" << endl;
      gSystem->Exit(1);
  }
  // load detector/material macros and execute Init() function

  if (Enable::PIPE) PipeInit();
  if (Enable::TrackingService) TrackingServiceInit();
  if (Enable::MVTX) MvtxInit();
  if (Enable::INTT) InttInit();
  if (Enable::TPC) TPCInit();
  if (Enable::MICROMEGAS) MicromegasInit();
  if (Enable::BBC) BbcInit();
  if (Enable::CEMCALBEDO) CEmcAlbedoInit();
  if (Enable::CEMC) CEmcInit();
  if (Enable::HCALIN) HCalInnerInit();
  if (Enable::MAGNET) MagnetInit();
  MagnetFieldInit(); // We want the field - even if the magnet volume is disabled
  if (Enable::HCALOUT) HCalOuterInit();
  if (Enable::PLUGDOOR) PlugDoorInit();
  if (Enable::EPD) EPDInit();
  if (Enable::BEAMLINE)
  {
    BeamLineInit();
    if (Enable::ZDC)
    {
      ZDCInit();
    }
  }
  if (Enable::USER) UserInit();
  if (Enable::BLACKHOLE) BlackHoleInit();
}
```



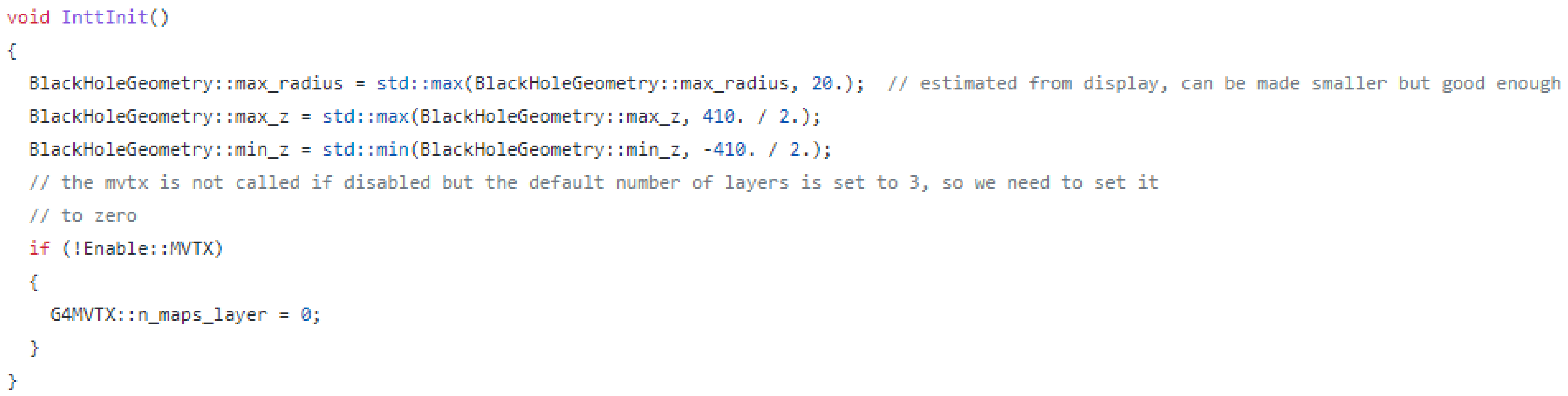
```cpp
int G4Setup()
{
  //---------------
  // Fun4All server
  //---------------

  Fun4AllServer *se = Fun4AllServer::instance();

  PHG4Reco *g4Reco = new PHG4Reco();
  g4Reco->set_rapidity_coverage(1.1);  // according to drawings
  WorldInit(g4Reco);
  if (G4P6DECAYER::decayType != EDecayType::kAll)
  {
    g4Reco->set_force_decay(G4P6DECAYER::decayType);
  }
```



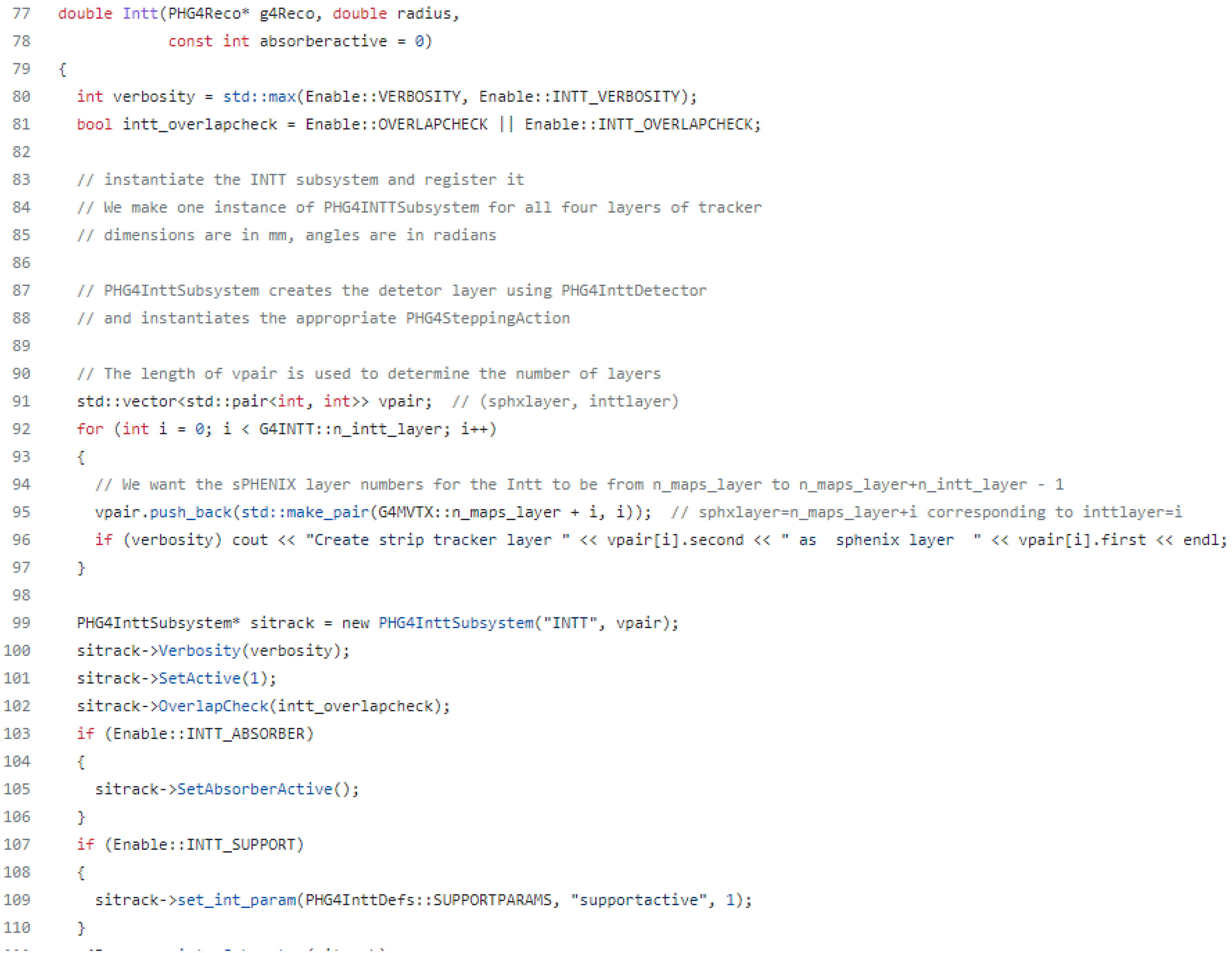
```cpp
// the radius is an older protection against overlaps, it is not
// clear how well this works nowadays but it doesn't hurt either
  double radius = 0.;

  if (Enable::PIPE) radius = Pipe(g4Reco, radius);
  if (Enable::TrackingService) TrackingService(g4Reco, radius);
  if (Enable::MVTX) radius = Mvtx(g4Reco, radius);
  if (Enable::INTT) radius = Intt(g4Reco, radius);
  if (Enable::TPC) radius = TPC(g4Reco, radius);
  if (Enable::MICROMEGAS) Micromegas(g4Reco);
  if (Enable::BBC) Bbc(g4Reco);
  if (Enable::CEMCALBEDO) CEmcAlbedo(g4Reco);
  if (Enable::CEMC) radius = CEmc(g4Reco, radius, 8);
  if (Enable::HCALIN) radius = HCalInner(g4Reco, radius, 4);
  if (Enable::MAGNET) radius = Magnet(g4Reco, radius);
  if (Enable::HCALOUT) radius = HCalOuter(g4Reco, radius, 4);
  if (Enable::PLUGDOOR) PlugDoor(g4Reco);
  if (Enable::EPD) EPD(g4Reco);
  if (Enable::BEAMLINE)
  {
    BeamLineDefineMagnets(g4Reco);
    BeamLineDefineBeamPipe(g4Reco);
```



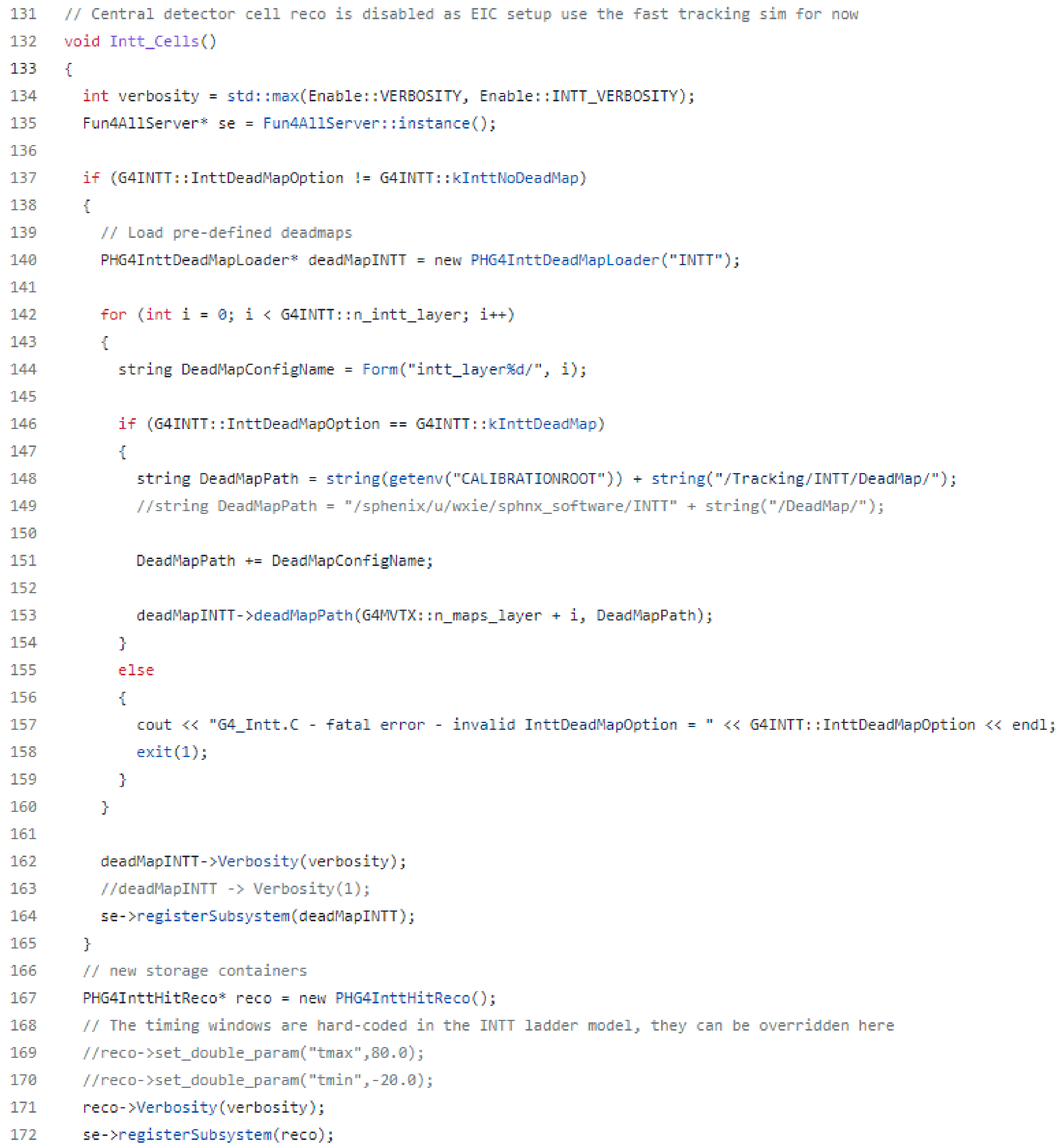
```
namespace G4INTT
{
  int n_intt_layer = 4;           // must be 4 or 0, setting to zero removes INTT completely
  double intt_radius_max = 140.;  // including stagger radius (mm)
  int laddertype[4] = {PHG4InttDefs::SEGMENTATION_PHI,
                       PHG4InttDefs::SEGMENTATION_PHI,
                       PHG4InttDefs::SEGMENTATION_PHI,
                       PHG4InttDefs::SEGMENTATION_PHI};
  int nladder[4] = {12, 12, 16, 16};
  double sensor_radius[4] = {7.188 - 36e-4, 7.732 - 36e-4, 9.680 - 36e-4, 10.262 - 36e-4};

  double offsetphi[4] = {0.0, 0.5 * 360.0 / nladder[1], 0.0, 0.5 * 360.0 / nladder[3]};

  enum enu_InttDeadMapType  // Dead map options for INTT
  {
    kInttNoDeadMap = 0,  // All channel in Intt is alive
    kInttDeadMap = 1,    // with dead channel
  };
  //enu_InttDeadMapType InttDeadMapOption = kInttNoDeadMap;  // Choose Intt deadmap here
  enu_InttDeadMapType InttDeadMapOption = kInttDeadMap;  // Choose Intt deadmap here

}  // namespace G4INTT
```



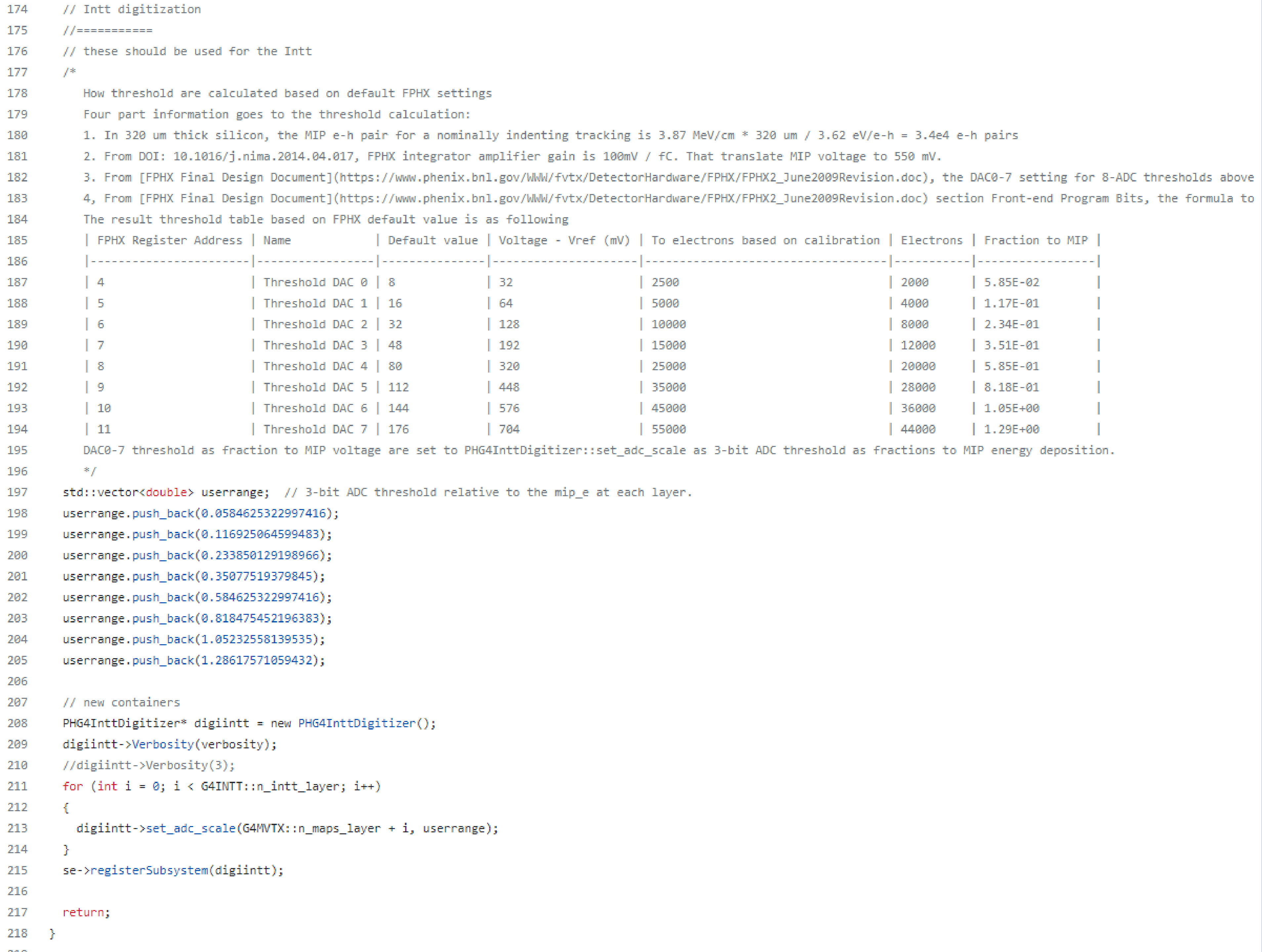
```
void InttInit()
{
  BlackHoleGeometry::max_radius = std::max(BlackHoleGeometry::max_radius, 20.);  // estimated from display, can be made smaller but good enough
  BlackHoleGeometry::max_z = std::max(BlackHoleGeometry::max_z, 410. / 2.);
  BlackHoleGeometry::min_z = std::min(BlackHoleGeometry::min_z, -410. / 2.);
  // the mvtx is not called if disabled but the default number of layers is set to 3, so we need to set it
  // to zero
  if (!Enable::MVTX)
  {
    G4MVTX::n_maps_layer = 0;
  }
}
```



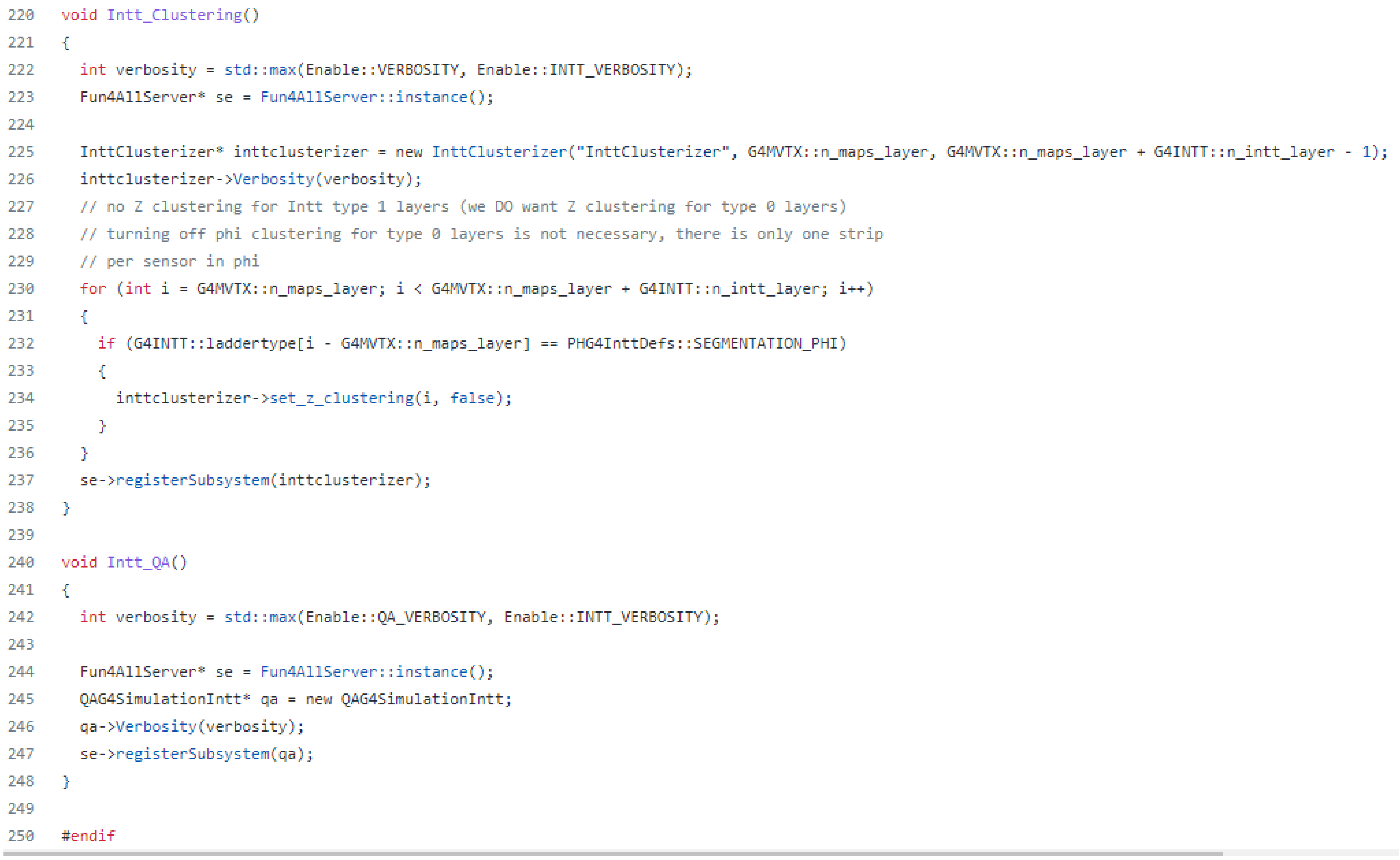
```cpp
double Intt(PHG4Reco* g4Reco, double radius,
            const int absorberactive = 0)
{
  int verbosity = std::max(Enable::VERBOSITY, Enable::INTT_VERBOSITY);
  bool intt_overlapcheck = Enable::OVERLAPCHECK || Enable::INTT_OVERLAPCHECK;

  // instantiate the INTT subsystem and register it
  // We make one instance of PHG4INTTSubsystem for all four layers of tracker
  // dimensions are in mm, angles are in radians

  // PHG4InttSubsystem creates the detetor layer using PHG4InttDetector
  // and instantiates the appropriate PHG4SteppingAction

  // The length of vpair is used to determine the number of layers
  std::vector<std::pair<int, int>> vpair;  // (sphxlayer, inttlayer)
  for (int i = 0; i < G4INTT::n_intt_layer; i++)
  {
    // We want the sPHENIX layer numbers for the Intt to be from n_maps_layer to n_maps_layer+n_intt_layer - 1
    vpair.push_back(std::make_pair(G4MVTX::n_maps_layer + i, i));  // sphxlayer=n_maps_layer+i corresponding to inttlayer=i
    if (verbosity) cout << "Create strip tracker layer " << vpair[i].second << " as  sphenix layer  " << vpair[i].first << endl;
  }

  PHG4InttSubsystem* sitrack = new PHG4InttSubsystem("INTT", vpair);
  sitrack->Verbosity(verbosity);
  sitrack->SetActive(1);
  sitrack->OverlapCheck(intt_overlapcheck);
  if (Enable::INTT_ABSORBER)
  {
    sitrack->SetAbsorberActive();
  }
  if (Enable::INTT_SUPPORT)
  {
    sitrack->set_int_param(PHG4InttDefs::SUPPORTPARAMS, "supportactive", 1);
  }
```



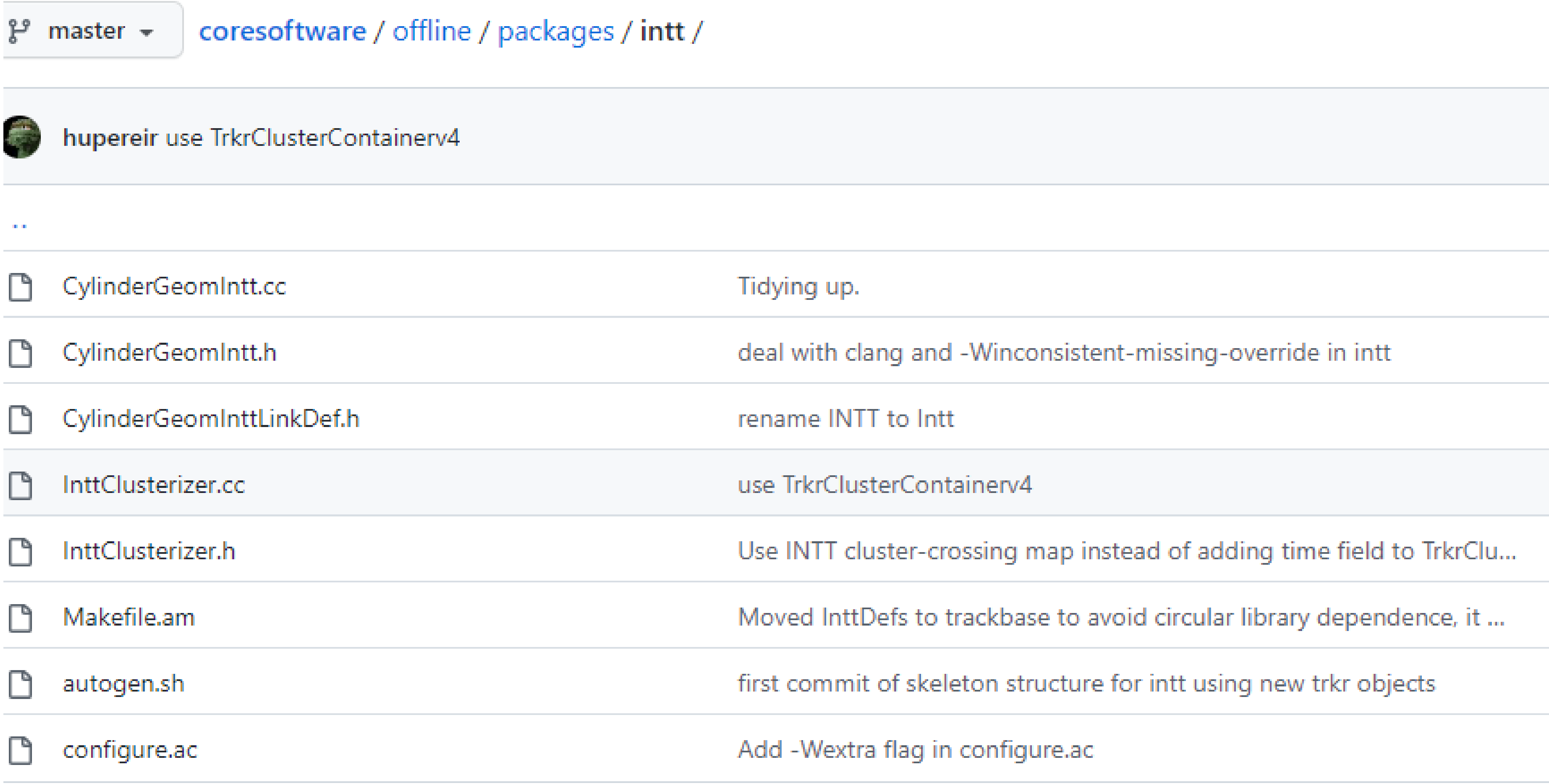
```
// Central detector cell reco is disabled as EIC setup use the fast tracking sim for now
void Intt_Cells()
{
  int verbosity = std::max(Enable::VERBOSITY, Enable::INTT_VERBOSITY);
  Fun4AllServer* se = Fun4AllServer::instance();

  if (G4INTT::InttDeadMapOption != G4INTT::kInttNoDeadMap)
  {
    // Load pre-defined deadmaps
    PHG4InttDeadMapLoader* deadMapINTT = new PHG4InttDeadMapLoader("INTT");

    for (int i = 0; i < G4INTT::n_intt_layer; i++)
    {
      string DeadMapConfigName = Form("intt_layer%d/", i);

      if (G4INTT::InttDeadMapOption == G4INTT::kInttDeadMap)
      {
        string DeadMapPath = string(getenv("CALIBRATIONROOT")) + string("/Tracking/INTT/DeadMap/");
        //string DeadMapPath = "/sphenix/u/wxie/sphnx_software/INTT" + string("/DeadMap/");

        DeadMapPath += DeadMapConfigName;

        deadMapINTT->deadMapPath(G4MVTX::n_maps_layer + i, DeadMapPath);
      }
      else
      {
        cout << "G4_Intt.C - fatal error - invalid InttDeadMapOption = " << G4INTT::InttDeadMapOption << endl;
        exit(1);
      }
    }

    deadMapINTT->Verbosity(verbosity);
    //deadMapINTT -> Verbosity(1);
    se->registerSubsystem(deadMapINTT);
  }
  // new storage containers
  PHG4InttHitReco* reco = new PHG4InttHitReco();
  // The timing windows are hard-coded in the INTT ladder model, they can be overridden here
  //reco->set_double_param("tmax",80.0);
  //reco->set_double_param("tmin",-20.0);
  reco->Verbosity(verbosity);
  se->registerSubsystem(reco);
```



```
  // Intt digitization
  //===========
  // these should be used for the Intt
  /*
     How threshold are calculated based on default FPHX settings
     Four part information goes to the threshold calculation:
     1. In 320 um thick silicon, the MIP e-h pair for a nominally indenting tracking is 3.87 MeV/cm * 320 um / 3.62 eV/e-h = 3.4e4 e-h pairs
     2. From DOI: 10.1016/j.nima.2014.04.017, FPHX integrator amplifier gain is 100mV / fC. That translate MIP voltage to 550 mV.
     3. From [FPHX Final Design Document](https://www.phenix.bnl.gov/WWW/fvtx/DetectorHardware/FPHX/FPHX2_June2009Revision.doc), the DAC0-7 setting for 8-ADC thresholds above
     4, From [FPHX Final Design Document](https://www.phenix.bnl.gov/WWW/fvtx/DetectorHardware/FPHX/FPHX2_June2009Revision.doc) section Front-end Program Bits, the formula to
     The result threshold table based on FPHX default value is as following
     | FPHX Register Address | Name             | Default value | Voltage - Vref (mV) | To electrons based on calibration | Electrons | Fraction to MIP |
     |-----------------------|------------------|---------------|---------------------|-----------------------------------|-----------|-----------------|
     | 4                     | Threshold DAC 0 | 8             | 32                  | 2500                              | 2000      | 5.85E-02        |
     | 5                     | Threshold DAC 1 | 16            | 64                  | 5000                              | 4000      | 1.17E-01        |
     | 6                     | Threshold DAC 2 | 32            | 128                 | 10000                             | 8000      | 2.34E-01        |
     | 7                     | Threshold DAC 3 | 48            | 192                 | 15000                             | 12000     | 3.51E-01        |
     | 8                     | Threshold DAC 4 | 80            | 320                 | 25000                             | 20000     | 5.85E-01        |
     | 9                     | Threshold DAC 5 | 112           | 448                 | 35000                             | 28000     | 8.18E-01        |
     | 10                    | Threshold DAC 6 | 144           | 576                 | 45000                             | 36000     | 1.05E+00        |
     | 11                    | Threshold DAC 7 | 176           | 704                 | 55000                             | 44000     | 1.29E+00        |
     DAC0-7 threshold as fraction to MIP voltage are set to PHG4InttDigitizer::set_adc_scale as 3-bit ADC threshold as fractions to MIP energy deposition.
     */
  std::vector<double> userrange;  // 3-bit ADC threshold relative to the mip_e at each layer.
  userrange.push_back(0.0584625322997416);
  userrange.push_back(0.116925064599483);
  userrange.push_back(0.233850129198966);
  userrange.push_back(0.35077519379845);
  userrange.push_back(0.584625322997416);
  userrange.push_back(0.818475452196383);
  userrange.push_back(1.05232558139535);
  userrange.push_back(1.28617571059432);

  // new containers
  PHG4InttDigitizer* digiintt = new PHG4InttDigitizer();
  digiintt->Verbosity(verbosity);
  //digiintt->Verbosity(3);
  for (int i = 0; i < G4INTT::n_intt_layer; i++)
  {
    digiintt->set_adc_scale(G4MVTX::n_maps_layer + i, userrange);
  }
  se->registerSubsystem(digiintt);

  return;
}
```



```
void Intt_Clustering()
{
  int verbosity = std::max(Enable::VERBOSITY, Enable::INTT_VERBOSITY);
  Fun4AllServer* se = Fun4AllServer::instance();

  InttClusterizer* inttclusterizer = new InttClusterizer("InttClusterizer", G4MVTX::n_maps_layer, G4MVTX::n_maps_layer + G4INTT::n_intt_layer - 1);
  inttclusterizer->Verbosity(verbosity);
  // no Z clustering for Intt type 1 layers (we DO want Z clustering for type 0 layers)
  // turning off phi clustering for type 0 layers is not necessary, there is only one strip
  // per sensor in phi
  for (int i = G4MVTX::n_maps_layer; i < G4MVTX::n_maps_layer + G4INTT::n_intt_layer; i++)
  {
    if (G4INTT::laddertype[i - G4MVTX::n_maps_layer] == PHG4InttDefs::SEGMENTATION_PHI)
    {
      inttclusterizer->set_z_clustering(i, false);
    }
  }
  se->registerSubsystem(inttclusterizer);
}

void Intt_QA()
{
  int verbosity = std::max(Enable::QA_VERBOSITY, Enable::INTT_VERBOSITY);

  Fun4AllServer* se = Fun4AllServer::instance();
  QAG4SimulationIntt* qa = new QAG4SimulationIntt;
  qa->Verbosity(verbosity);
  se->registerSubsystem(qa);
}

#endif
```

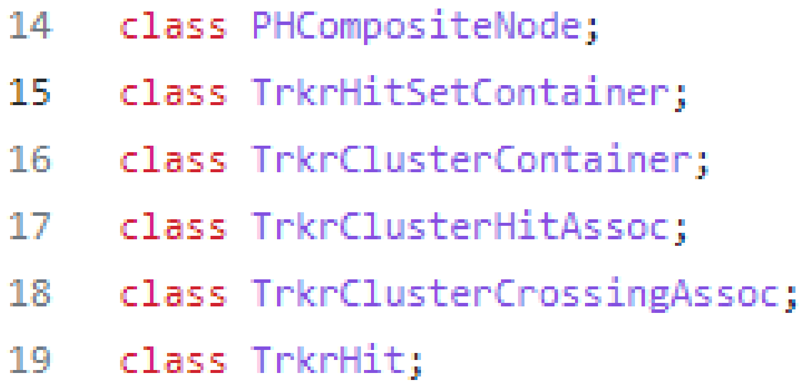


coresoftware / offline / packages / intt /

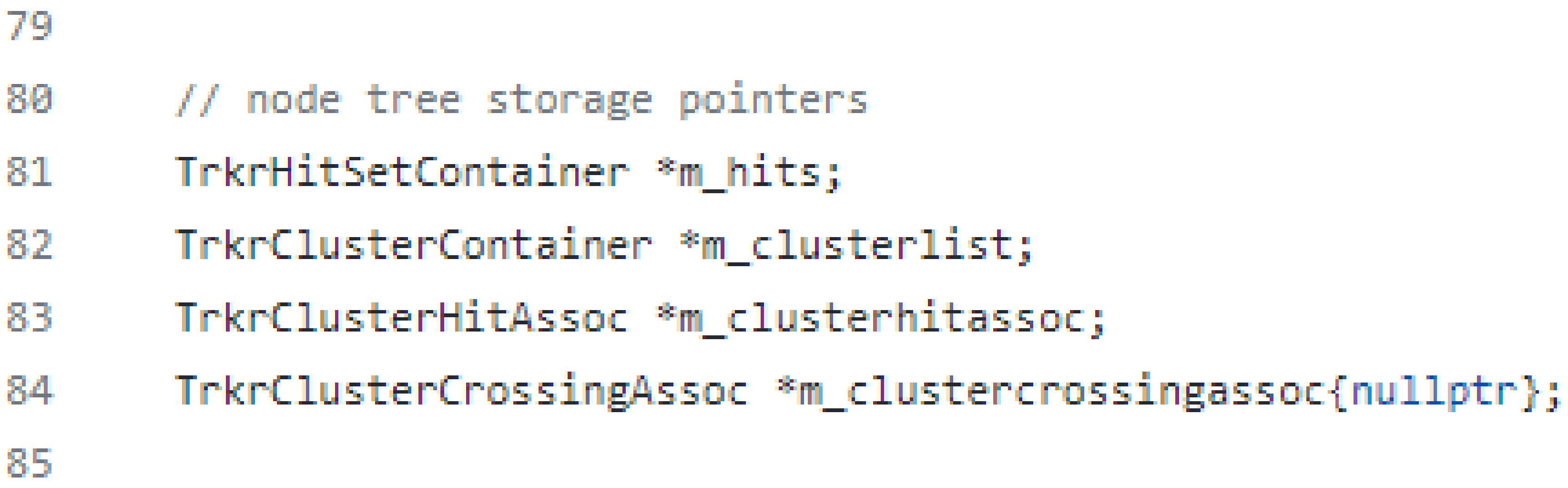
hupereir use TrkrClusterContainerv4

..

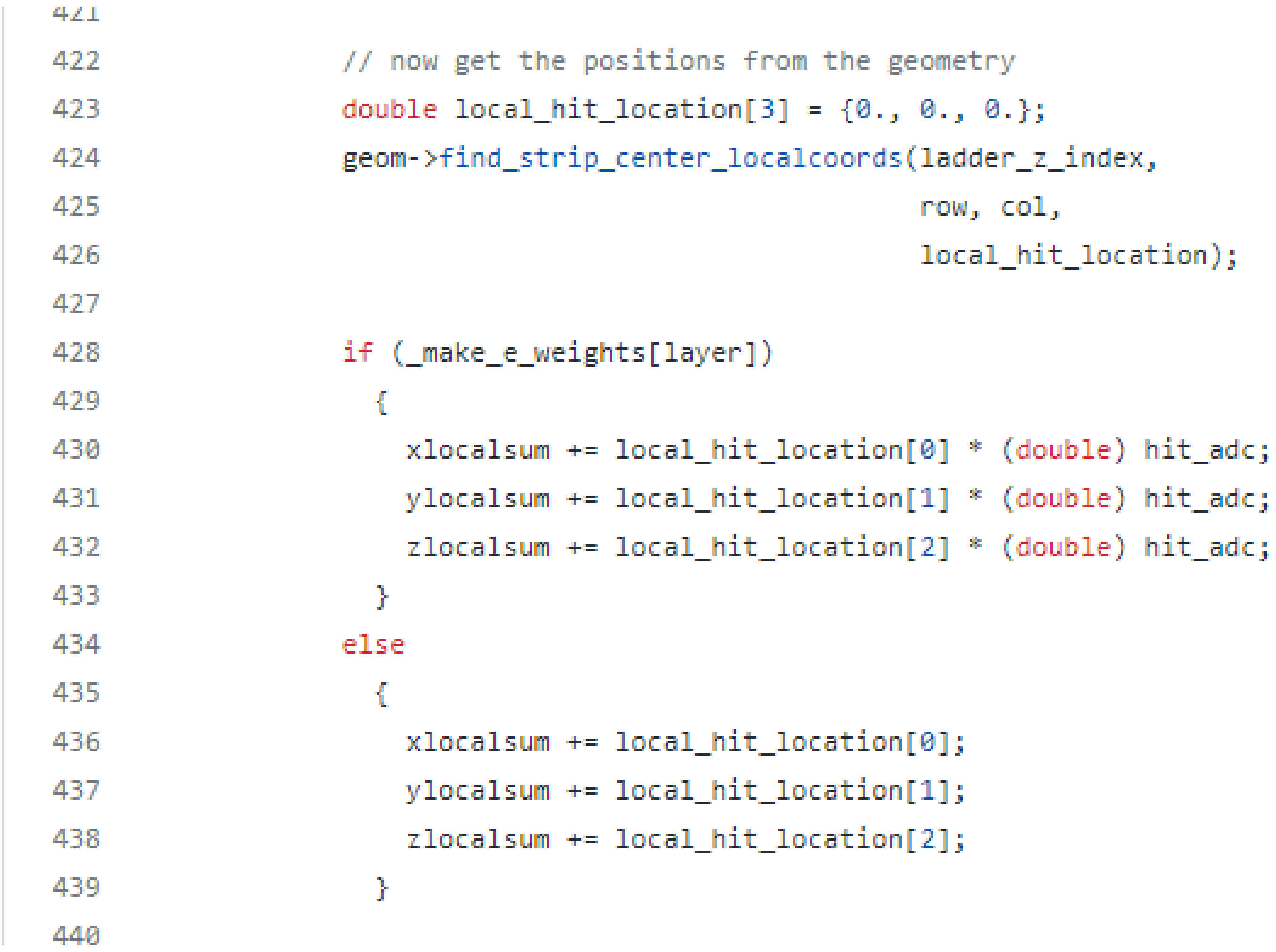
CylinderGeomIntt.cc	Tidying up.
CylinderGeomIntt.h	deal with clang and -Winconsistent-missing-override in intt
CylinderGeomInttLinkDef.h	rename INTT to Intt
InttClusterizer.cc	use TrkrClusterContainerv4
InttClusterizer.h	Use INTT cluster-crossing map instead of adding time field to TrkrClu...
Makefile.am	Moved InttDefs to trackbase to avoid circular library dependence, it ...
autogen.sh	first commit of skeleton structure for intt using new trkr objects
configure.ac	Add -Wextra flag in configure.ac



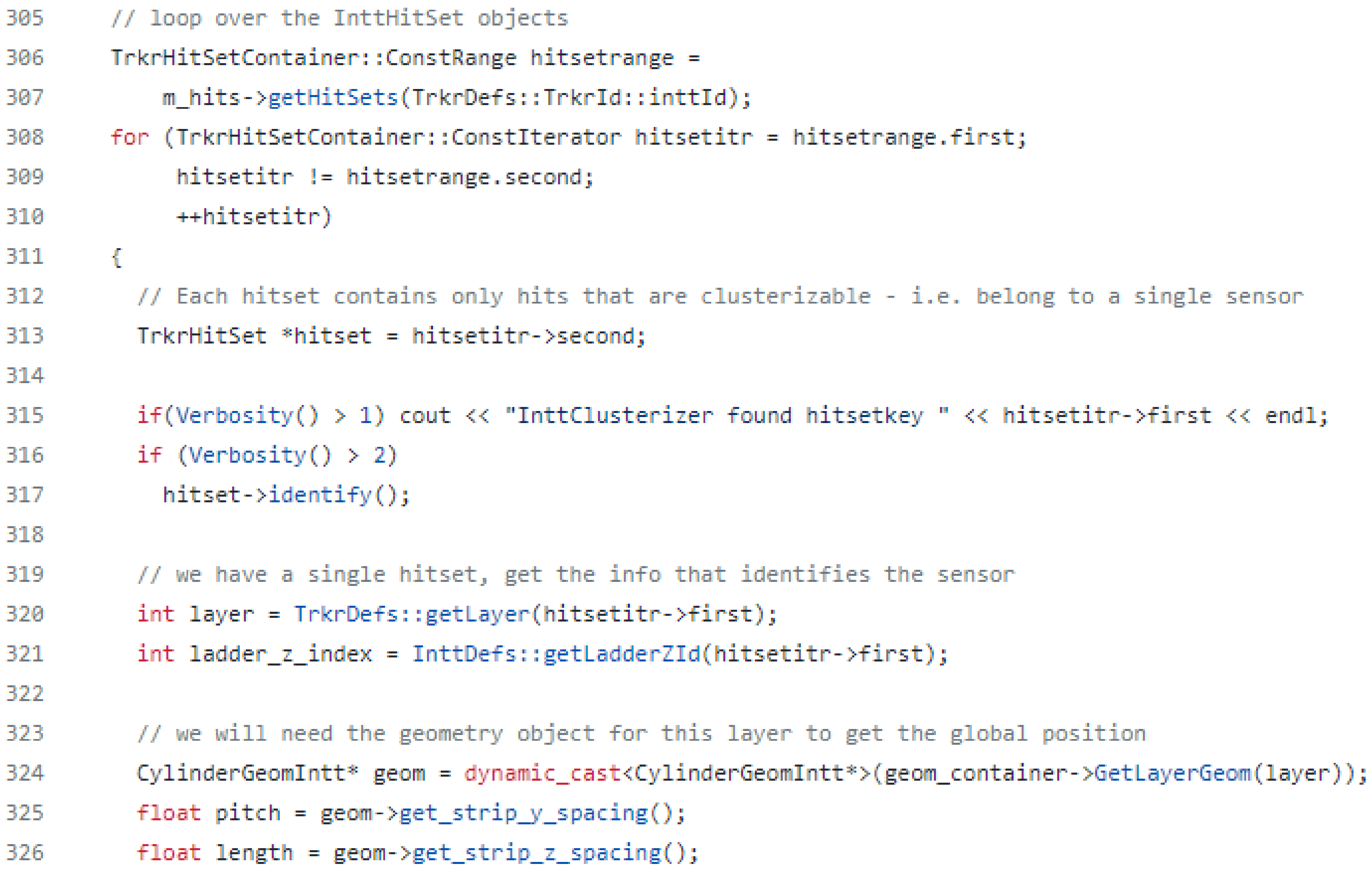
```cpp
class PHCompositeNode;
class TrkrHitSetContainer;
class TrkrClusterContainer;
class TrkrClusterHitAssoc;
class TrkrClusterCrossingAssoc;
class TrkrHit;
```



```cpp

    // node tree storage pointers
    TrkrHitSetContainer *m_hits;
    TrkrClusterContainer *m_clusterlist;
    TrkrClusterHitAssoc *m_clusterhitassoc;
    TrkrClusterCrossingAssoc *m_clustercrossingassoc{nullptr};

```



```
// now get the positions from the geometry
double local_hit_location[3] = {0., 0., 0.};
geom->find_strip_center_localcoords(ladder_z_index,
                                    row, col,
                                    local_hit_location);

if (_make_e_weights[layer])
  {
    xlocalsum += local_hit_location[0] * (double) hit_adc;
    ylocalsum += local_hit_location[1] * (double) hit_adc;
    zlocalsum += local_hit_location[2] * (double) hit_adc;
  }
else
  {
    xlocalsum += local_hit_location[0];
    ylocalsum += local_hit_location[1];
    zlocalsum += local_hit_location[2];
  }

```



```
// loop over the InttHitSet objects
TrkrHitSetContainer::ConstRange hitsetrange =
    m_hits->getHitSets(TrkrDefs::TrkrId::inttId);
for (TrkrHitSetContainer::ConstIterator hitsetitr = hitsetrange.first;
     hitsetitr != hitsetrange.second;
     ++hitsetitr)
{
  // Each hitset contains only hits that are clusterizable - i.e. belong to a single sensor
  TrkrHitSet *hitset = hitsetitr->second;

  if(Verbosity() > 1) cout << "InttClusterizer found hitsetkey " << hitsetitr->first << endl;
  if (Verbosity() > 2)
    hitset->identify();

  // we have a single hitset, get the info that identifies the sensor
  int layer = TrkrDefs::getLayer(hitsetitr->first);
  int ladder_z_index = InttDefs::getLadderZId(hitsetitr->first);

  // we will need the geometry object for this layer to get the global position
  CylinderGeomIntt* geom = dynamic_cast<CylinderGeomIntt*>(geom_container->GetLayerGeom(layer));
  float pitch = geom->get_strip_y_spacing();
  float length = geom->get_strip_z_spacing();
```