Development efforts of software framework to simulate DR calorimeter

Sanghyun Ko

Seoul National University

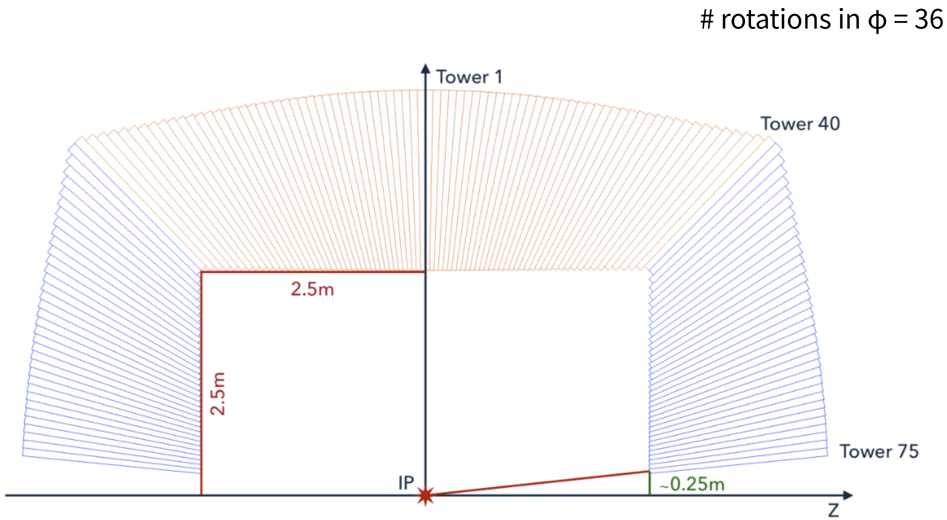
On behalf of the IDEA Dual-Readout Group

23 Jul 2020

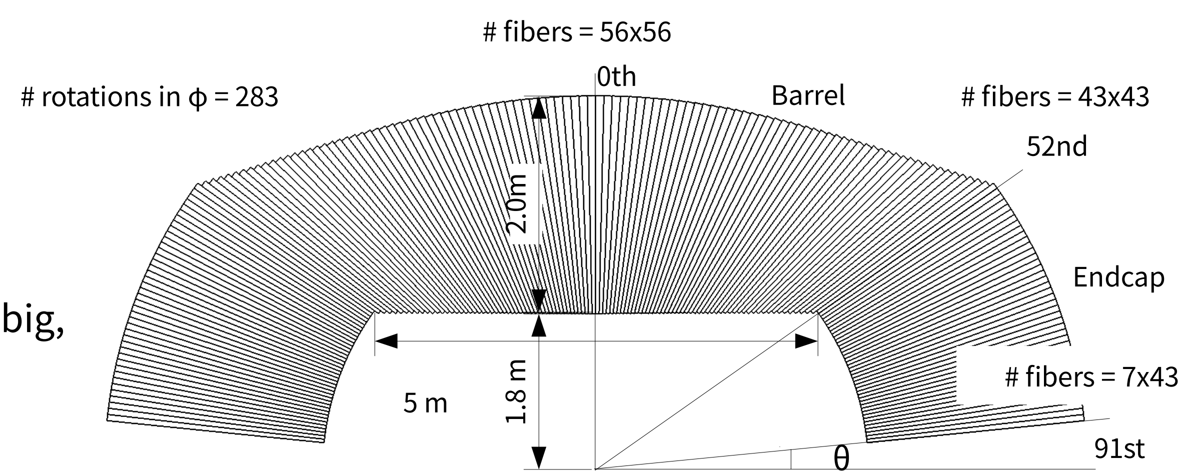
Standalone G4 simulation

Standalone application of G4 for dual-readout

- At the moment, standalone application of GEANT4 (with ROOT) is used for the simulation of dual-readout.
  - Standard & native way to do simulation study for detectors.
  - Historically, developed & evolved by two different groups of INFN & S. Korea.
- Excellent performances of DR calorimeter already presented by Iacopo yesterday [link].
- Presentation at FCCSW can be found at [link].



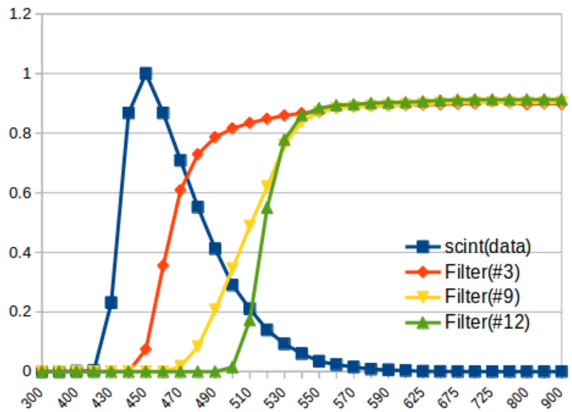
- INFN [github][Twiki]
  - Geometry adapted to the IDEA drift chamber.
- S. Korea [github]
  - Consistent with the one in CEPC CDR.
- Difference in the physics performance is not big, as already cross-checked several times internally [link].



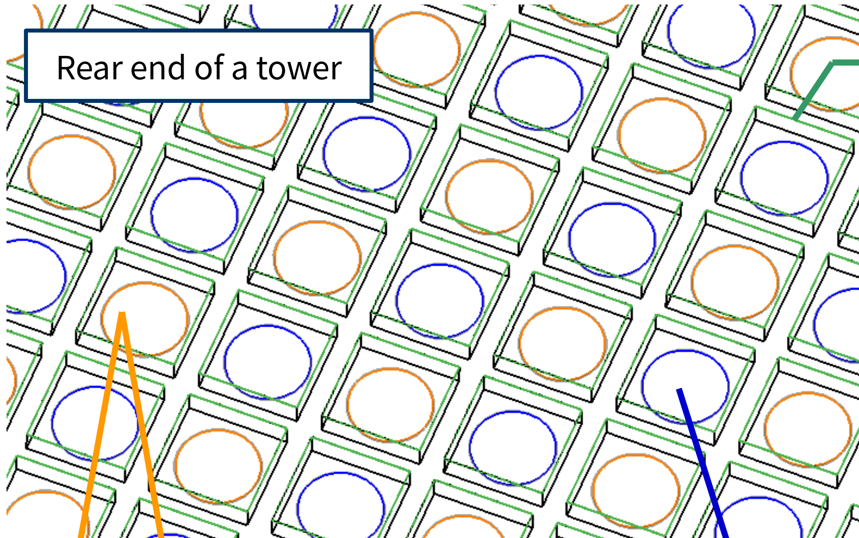
Required optical properties

GEANT4 setup – required optical properties for the simulation [Github]

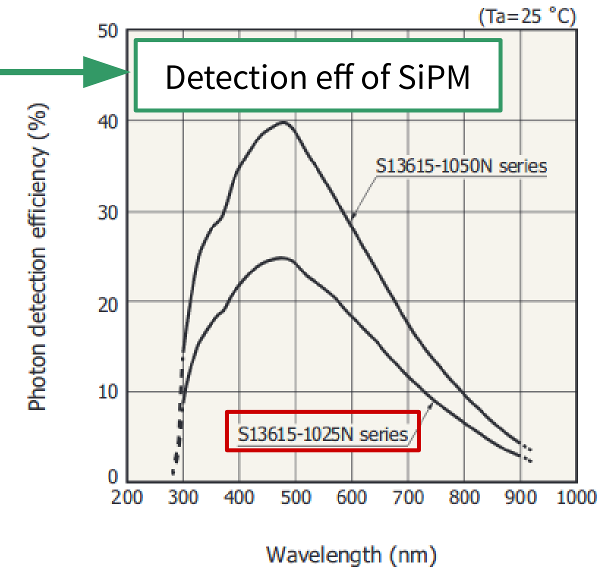
Transmission eff of filters



Rear end of a tower

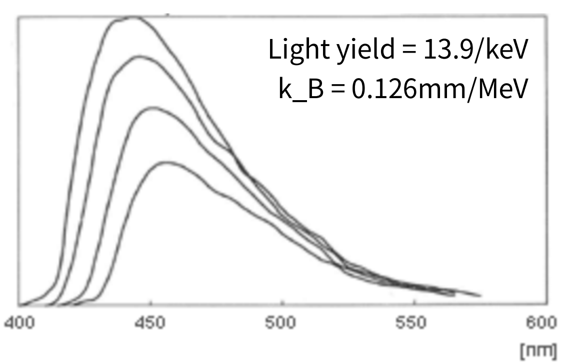


Detection eff of SiPM



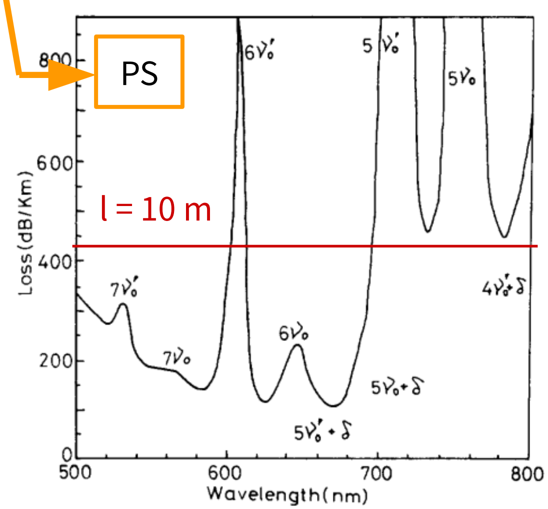
Attenuation loss diverges at 400nm → applied filter to S channel to mitigate it

Scintillation spectra of PS

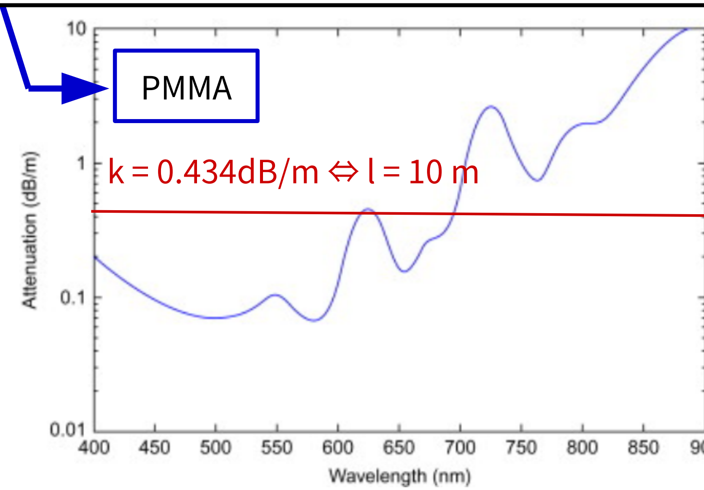


Attenuation loss of Polystyrene (PS) & PMMA

PS



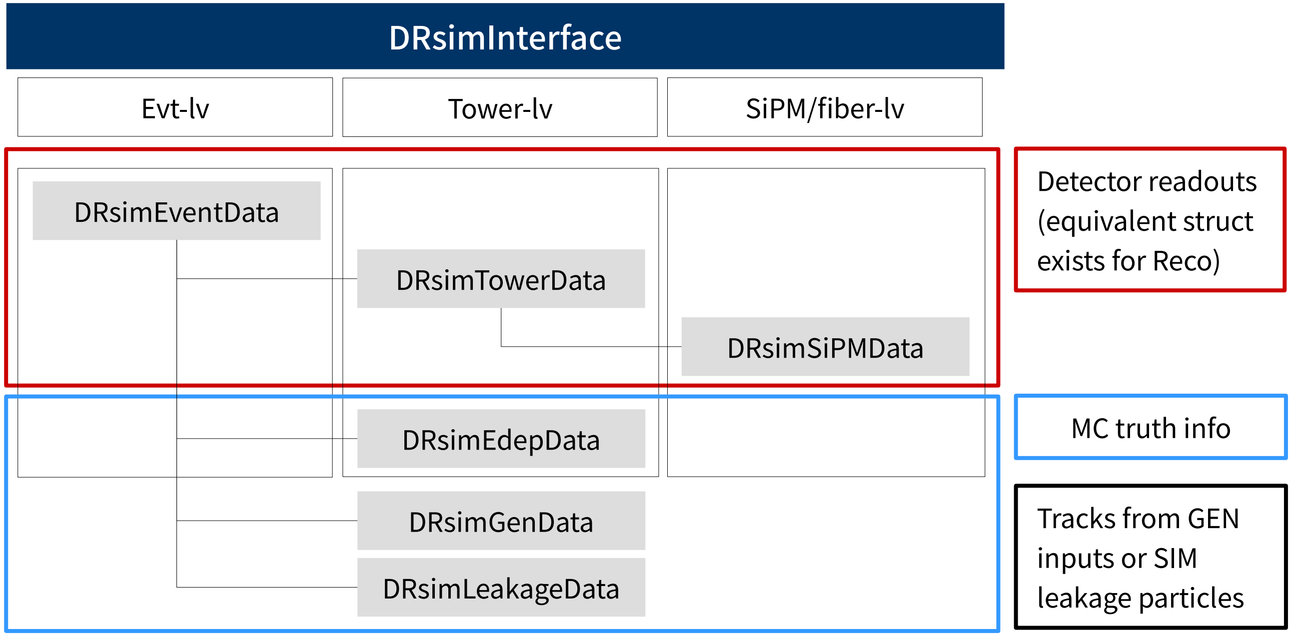
PMMA



Event Data Model

Event Data Model (EDM) of dual-readout calorimeter

- At the moment, the proto-EDM of dual-readout calorimeter simulation is constructed using the C++ class & structures [Github].
- Designed according to the geometrical hierarchy of DR calorimeter.
- I/O of a file is based on the TTree of ROOT [Github].
- Costs O(10 kb) – O(100 kb) per events.
- An interface to the common proto-EDM from the output of each of group is initialized (by Iacopo) [Github].




```
class DRsimInterface {
public:
  DRsimInterface();
  ~DRsimInterface();

  typedef std::pair<float,float> hitRange;
  typedef std::pair<int,int> hitXY;
  typedef std::map<hitRange, int> DRsimTimeStruct;
  typedef std::map<hitRange, int> DRsimWavlenSpectrum;
  typedef std::tuple<float,float,float> threeVector;

  struct DRsimSiPMData {
    DRsimSiPMData() {};
    virtual ~DRsimSiPMData() {};

    int count;
    int SiPMnum;
    int x;
    int y;
    threeVector pos;
    DRsimTimeStruct timeStruct;
    DRsimWavlenSpectrum wavlenSpectrum;
  };

  struct DRsimTowerData {
    DRsimTowerData() {};
    virtual ~DRsimTowerData() {};

    std::pair<int,float> towerTheta;
    std::pair<int,float> towerPhi;
    int numx;
    int numy;
    float innerR;
    float towerH;
    float dTheta;
    std::vector<DRsimSiPMData> SiPMs;
  };
```

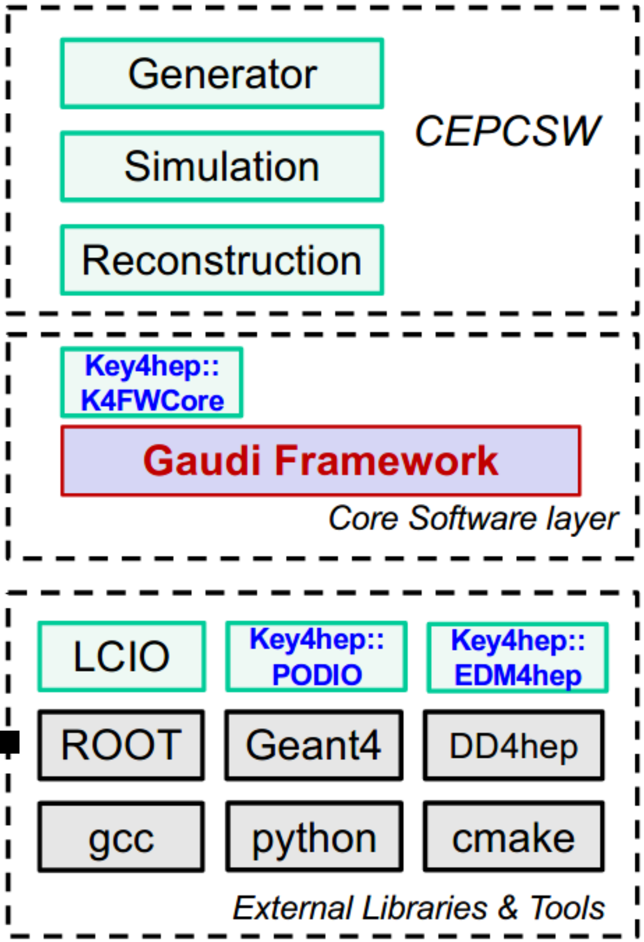
Migration to Key4HEP elements

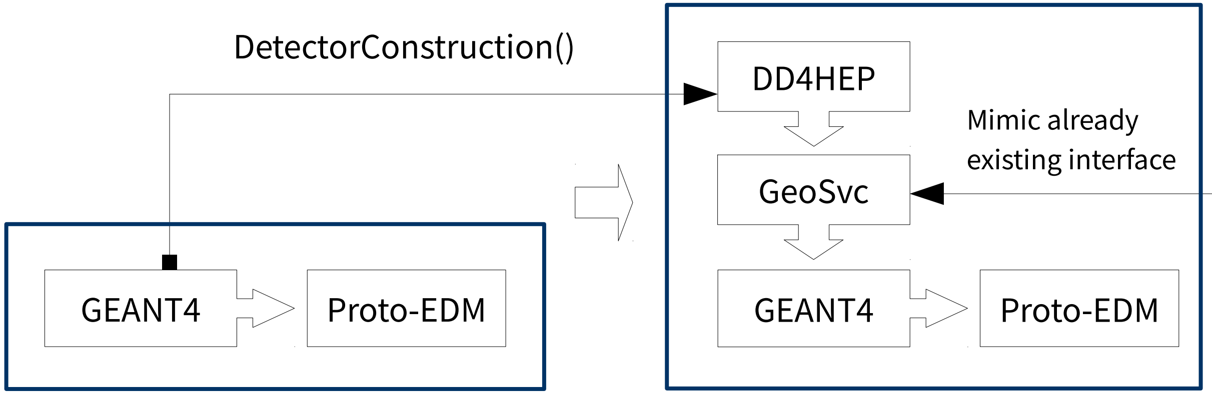
- Migration to centralized SW is a good opportunity to take a look into combined detector performances, e.g. tracker+calorimeter, ECAL+HCAL, .etc.
- However, migrating in a one big leap is not an ideal choice.
  - Core software is still rapidly evolving, may introduce extra efforts for integration & maintenance.
  - May need to test several options internally, usage of standalone SW is still too convenient to abandon.

→ Start by migrating to necessary subset of Key4HEP elements (e.g. DD4HEP, EDM4HEP) based on the current standalone SW is a natural choice.

- DD4HEP & GeoSvc interface will provide great convenience already for plugging-in other detector components compared to G4 standalone.

[link]





Migration efforts to DD4HEP

Initializing DD4HEP for dual-readout calorimeter

- Started migration effort to DD4HEP and initialized the DD4HEP code for dual-readout calorimeter [github] (at the development branch).
- Material properties (including optical properties) are translated to xml format.
- The convention of geometry in DD4HEP is very similar to that of GEANT4.
- Existing G4SensitiveDetector (SD) is translated into SD of DD4HEP [github], with DD4HEP segmentation & readouts to encode unique ID of readout systems into a single 64bit integer [github].

```xml
<readouts>
  <readout name="DRcaloSiPMreadout">
    <segmentation type="GridDRcalo"/>
    <!-- Mendatory to use the first 32 bits for tower infos & the last 32 bits for fiber/SiPM
    <id>system:2,eta:-8,phi:9,xmax:32:6,ymax:6,x:6,y:6,c:1,module:2</id>
  </readout>
</readouts>
```

```xml
<material name="DR_Polystyrene">
  <D value="1.032" unit="g/cm3"/>
  <composite n="19" ref="C"/>
  <composite n="21" ref="H"/>
  <property name="RINDEX" ref="RI_PS"/>
  <property name="ABSLENGTH" ref="AbsLen_PS"/>
  <property name="FASTCOMPONENT" ref="scintFast_PS"/>
  <constant name="SCINTILLATIONYIELD" value="13.9/keV"/>
  <constant name="FASTTIMECONSTANT" value="2.8*ns"/>
</material>

<matrix name="scintFast_PS" coldim="2" values="
  1.37760*eV 0.
  1.45864*eV 0.
  1.54980*eV 0.
  1.65312*eV 0.
  1.71013*eV 0.
  1.77120*eV 0.
  1.83680*eV 0.
  1.90745*eV 0.0003
  1.98375*eV 0.0008
  2.06640*eV 0.0032
  2.10143*eV 0.0057
  2.13766*eV 0.0084
  2.17516*eV 0.0153
  2.21400*eV 0.0234
  2.25426*eV 0.0343
  2.29600*eV 0.0604
  2.33932*eV 0.0927
  2.38431*eV 0.1398
```

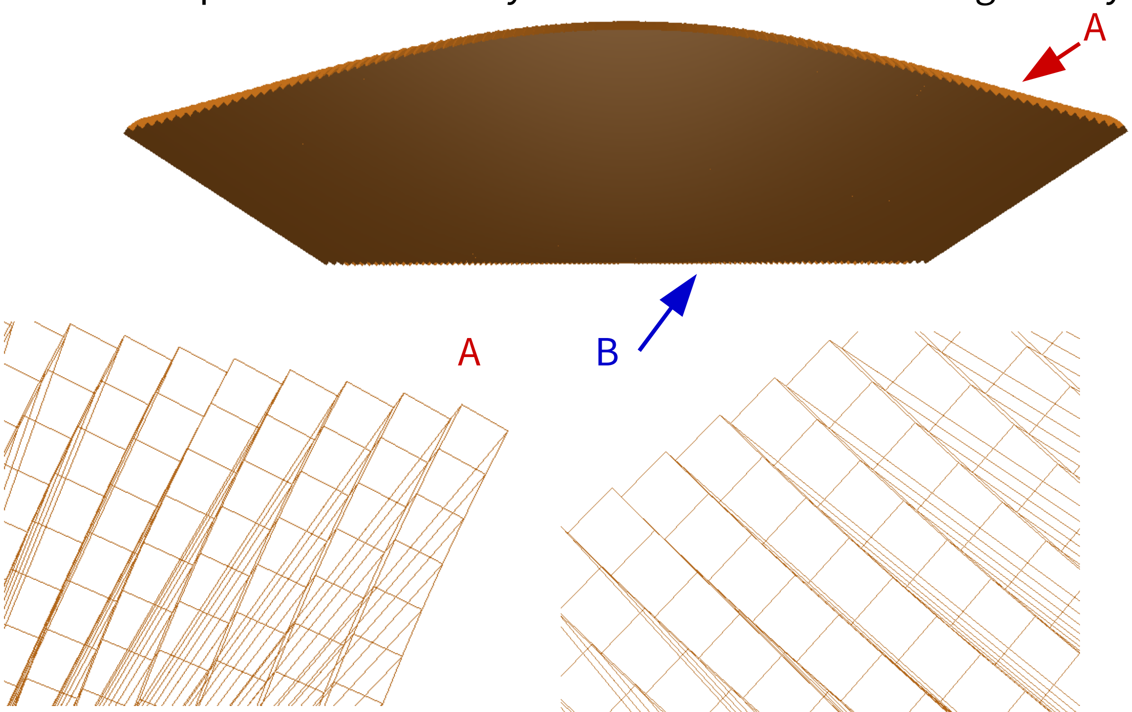
Analogous to G4Trap | to G4LogicalVolume | to G4VPhysicalVolume

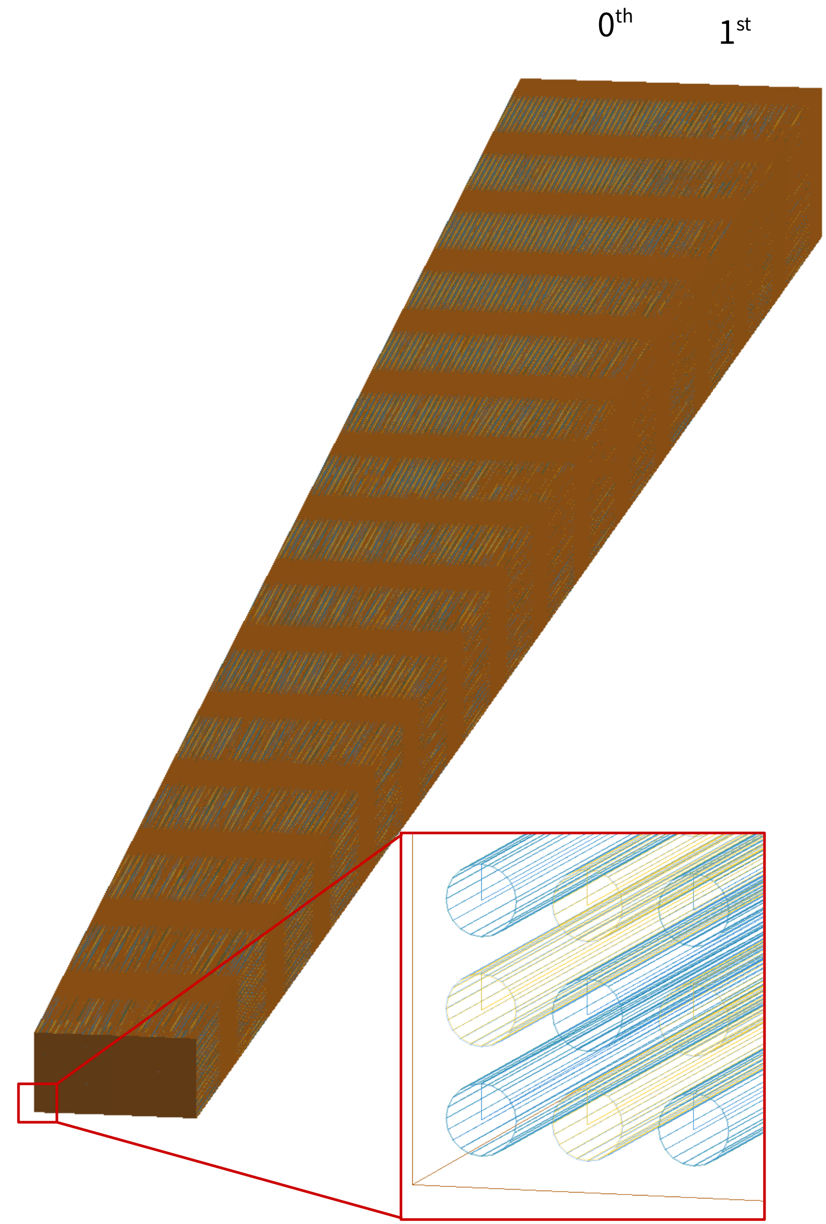
```cpp
dd4hep::Trap tower( fX_towerDim->height()/2., 0., 0., fParamBarrel->GetH1(), fParamBarrel->GetBl1(), fParamBarrel->GetTl1(), 0.,
                    fParamBarrel->GetH2(), fParamBarrel->GetBl2(), fParamBarrel->GetTl2(), 0. );

dd4hep::Volume towerVol( "tower"+std::to_string(towerNoLR), tower, fDescription->material(fX_towerDim->materialStr()) );
towerVol.setVisAttributes(*fDescription, fX_towerDim->visStr());

for (int nPhi = 0; nPhi < 30/*fX_towerDim->nphi()*/; nPhi++) {
  dd4hep::PlacedVolume towerPhys = fExperimentalHall->placeVolume( towerVol, towerNoLR*fX_towerDim->nphi()+nPhi, fParamBarrel->GetTransform3D(nPhi) );
  towerPhys.addPhysVolID("tower",towerNoLR*fX_towerDim->nphi()+nPhi);
}
```

Migration efforts to DD4HEP

Example of dual-readout calorimeter at DD4HEP

- Implemented barrel geometry with trapezoid copper towers & dual-readout fibers.
- Visualized with geoDisplay script of DD4HEP.
  - Barrel geometry
  - Copper towers (first two towers only in the barrel)
  - Cerenkov & scintillation fibres
- Caveat: Optical functionality of DD4HEP is not tested rigorously.



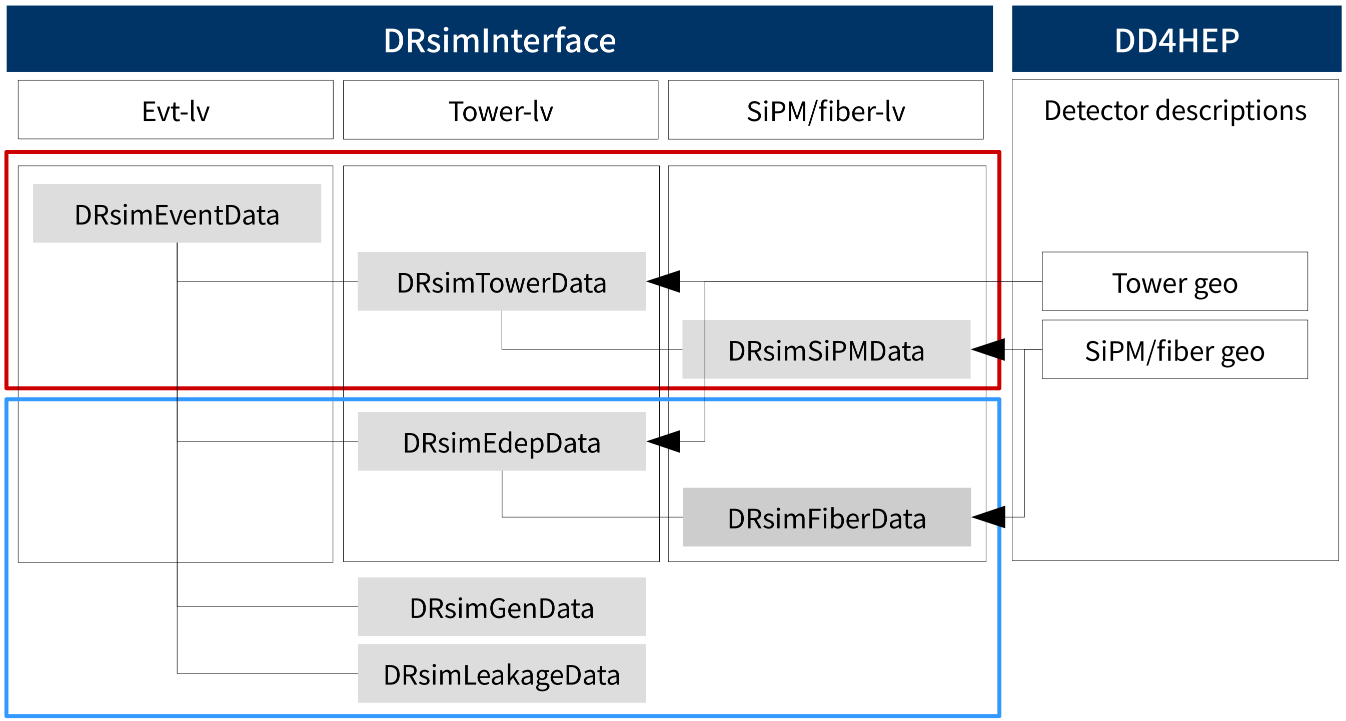


EDM with DD4HEP

Effect of DD4HEP to proto-EDM

- Dual-readout calorimeter has very complicated geometry with ~ O(100M) of channels.

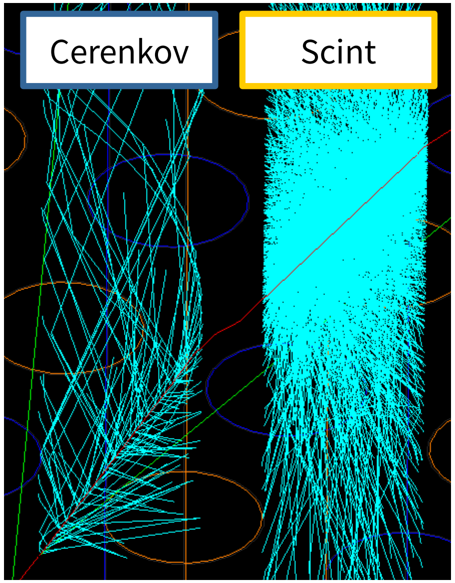
  i.e. geometry information (e.g. position of fibres/SiPMs) should not be repeated in every event.
- Instead of saving geometry information in EDM, geometry information will be loaded by DD4HEP using a unique readout ID with a encoded 64bit integer that contains all necessary informations to identify it.
- In this way DD4HEP can be interfaced to both of GEANT4 & reconstruction codes to utilize geometry information.

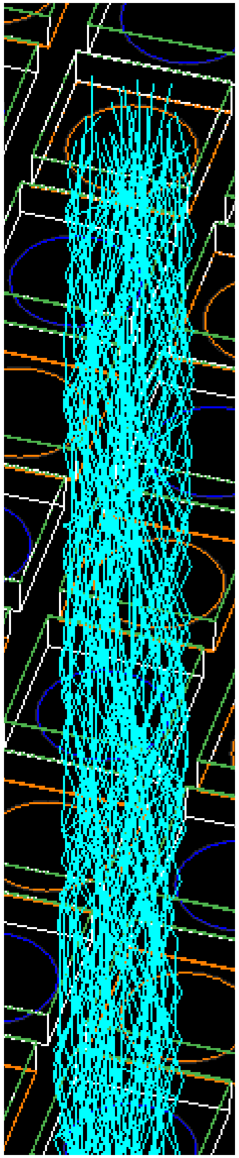


Speeding up optical photons tracking

Details needed to simulate dual-readout calorimeter

- Cerenkov & scintillation processes.
- Light attenuation of Polystyrene & PMMA.
- Transmission of optical surfaces, e.g. yellow filter, SiPM.
- Total internal reflection inside optical fibers.
  - Numerical aperture is important for the yield of Cerenkov channel.





CPU consumption for tracking optical photons

- A drawback for detailed simulation is CPU consumption caused by tracking optical photons.
- Single photon generates ~ O(10k) tracks for tracking, while there are ~ O(10k) optical photons per GeV of incident particle, results ~ O(100M) tracks per GeV.
- It takes 304 ± 88 min in average to produce an event! (20 GeV e-).

  → Developed a fast simulation module [Github] to compute efficiently while keeping the details.
- The module skips the intermediate tracking of optical photons and predicts the final position, time and momentum – by utilizing the fact that fibers are cylinder shape (i.e. facet normal always heads the center of the fiber).

  → Reduces CPU time to 4.62 ± 1.17 min per event (20 GeV e-) with identical results to full tracking.

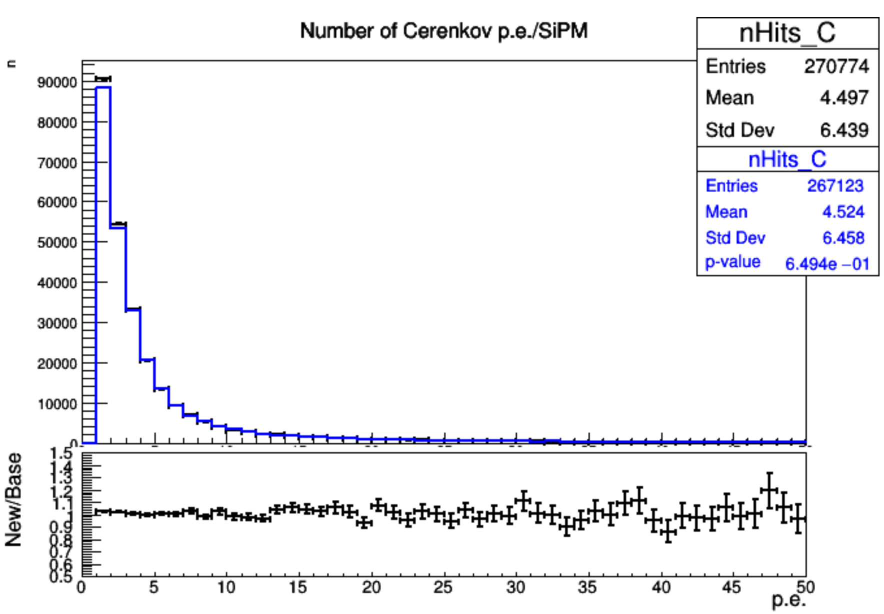
Speeding up optical photons tracking

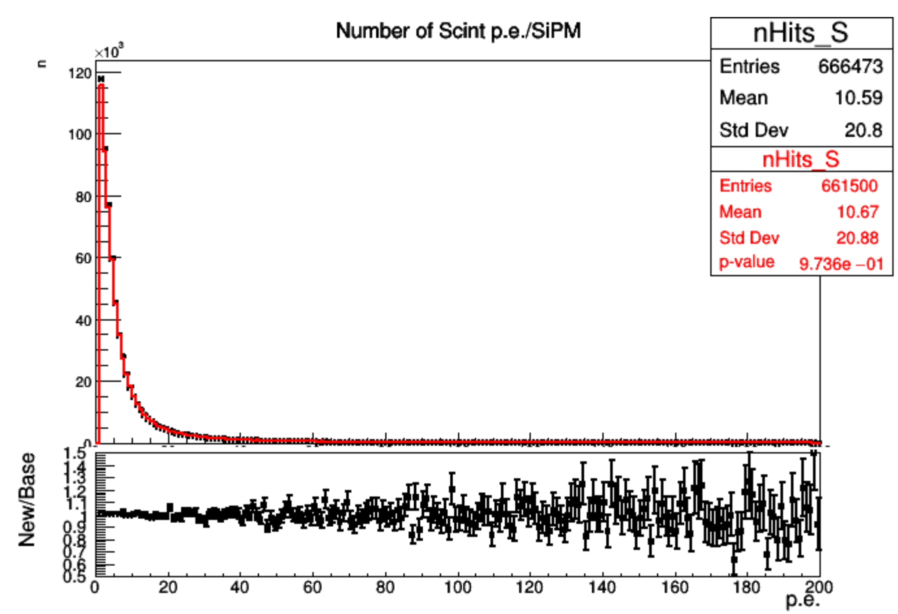
Comparison to full tracking & conventional fast simulation (GFlash)

- Role of the developed fastsim module is limited to managing optical photon transportation inside optical fibers.
- EM/hadronic physics is same to that of full simulation of G4, i.e. does NOT utilize any shower parameterization.

	GEANT4 fullsim	Fast Op transport	GFlash
Shower physics	Full tracking		Parameterization
Relative differences to GEANT4 fullsim	N/A	Optical photons inside optical fibers	EM/hadronic particles in the region of interest

- Comparison of Fast Op transport to full tracking shows good agreement.



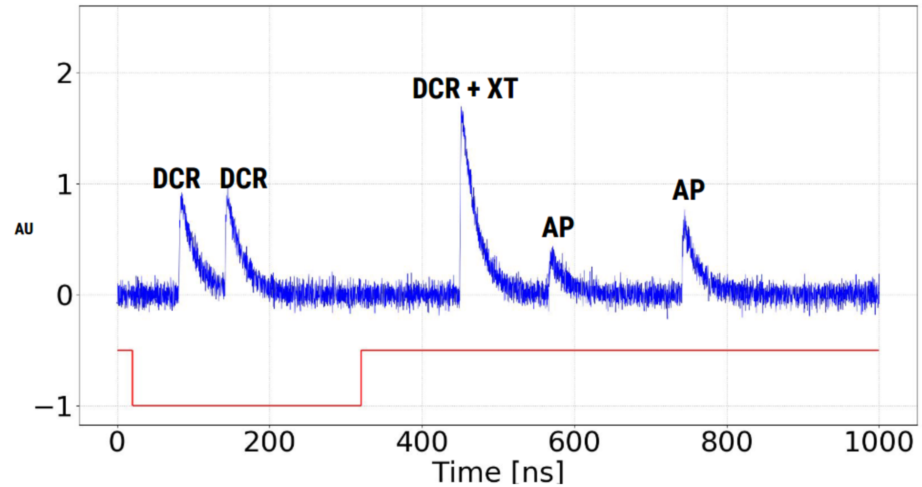


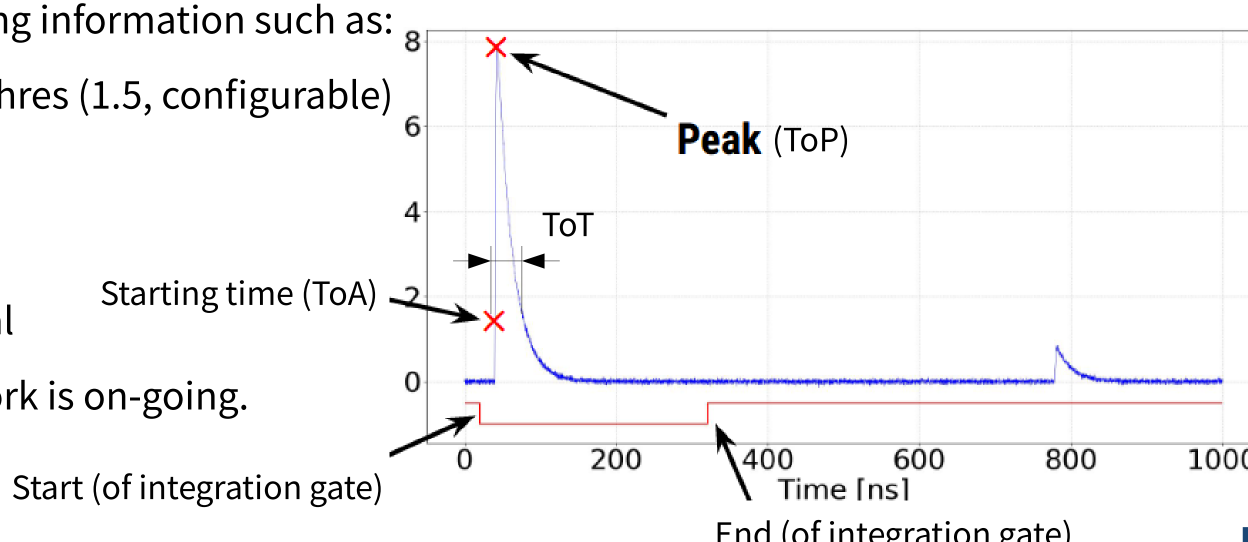
SiPM digitization

Code development on the SiPM digitization

By Edoardo Proserpio

- A realistic Python-based SiPM digitization code [github] has been developed and detailed studies on timing will be carried out with it.
- Detailed discussion can be found at [link1][link2].
- The digitization includes:
  - Electronic noise
  - SiPM non-linearity
  - Dark counts (DCR)
  - Crosstalk probability (XT)
  - After pulses (AP)
- It makes possible to extract timing information such as:
  - ToA – First time above the thres (1.5, configurable)
  - ToT – Time over thres
  - ToP – Time of the peak
  - or the whole digitized signal
- Integration effort to SW framework is on-going.





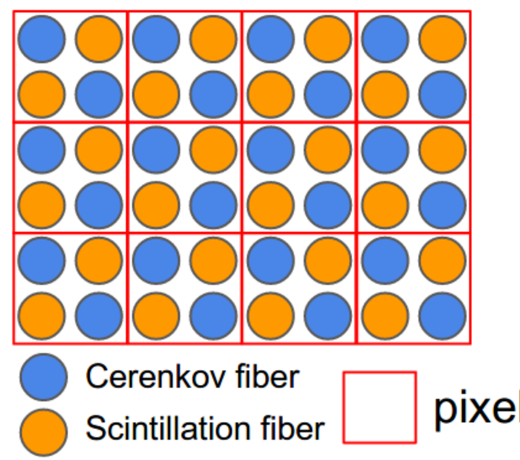
Particle ID

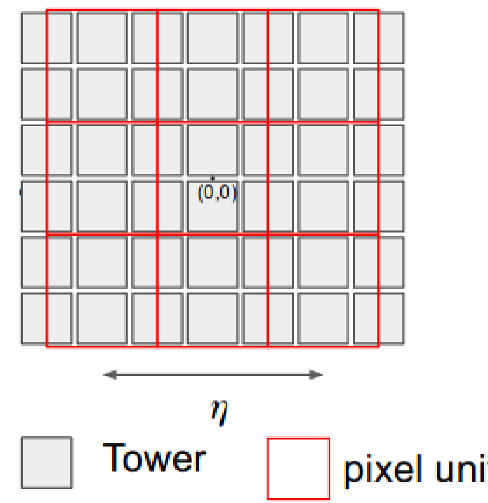
Particle identification using image-based ML

By Yunjae Lee

- Attempts to identify particles based on images (energy distributions) from dual-readout calorimeter.
- Images are constructed separately from scintillation & Cerenkov channel, respectively.
- Convolutional neural network (CNN) is used to discriminate images of different particles.

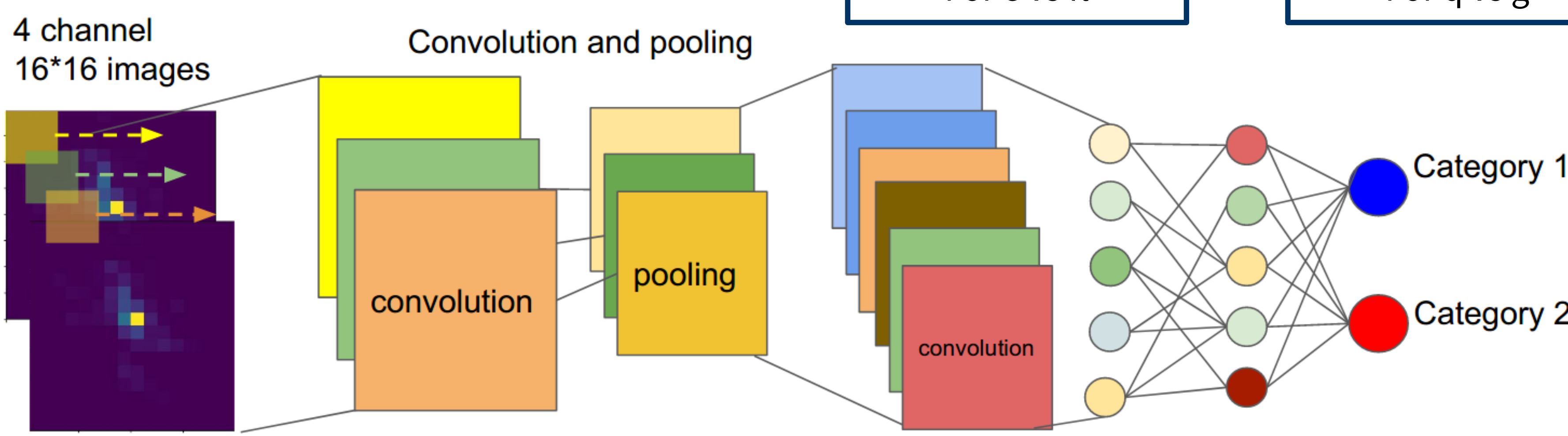
  Relatively simple (2 convolutional layers with 1 fully connected layer) model to prevent over-training.





For e vs π

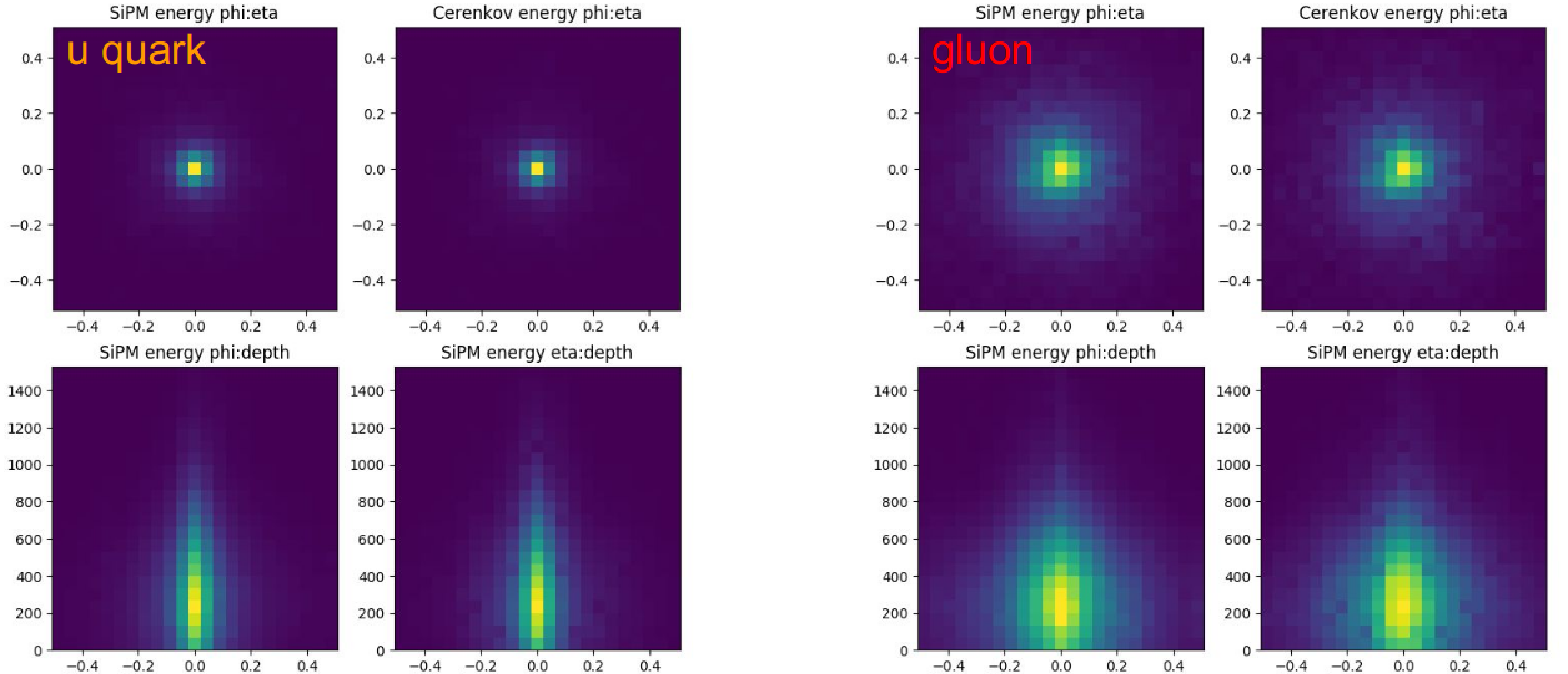
For q vs g



Particle ID

Averaged images of u-quarks and gluons

By Yunjae Lee

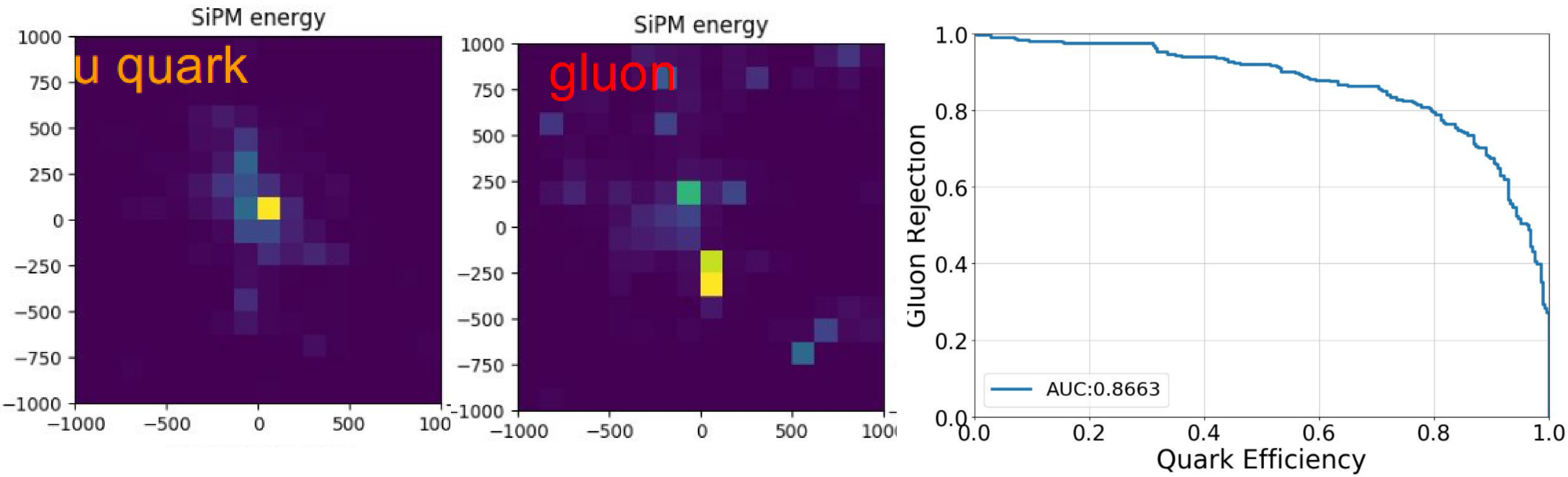


- Both u-quarks and gluons have same energy of 50 GeV.

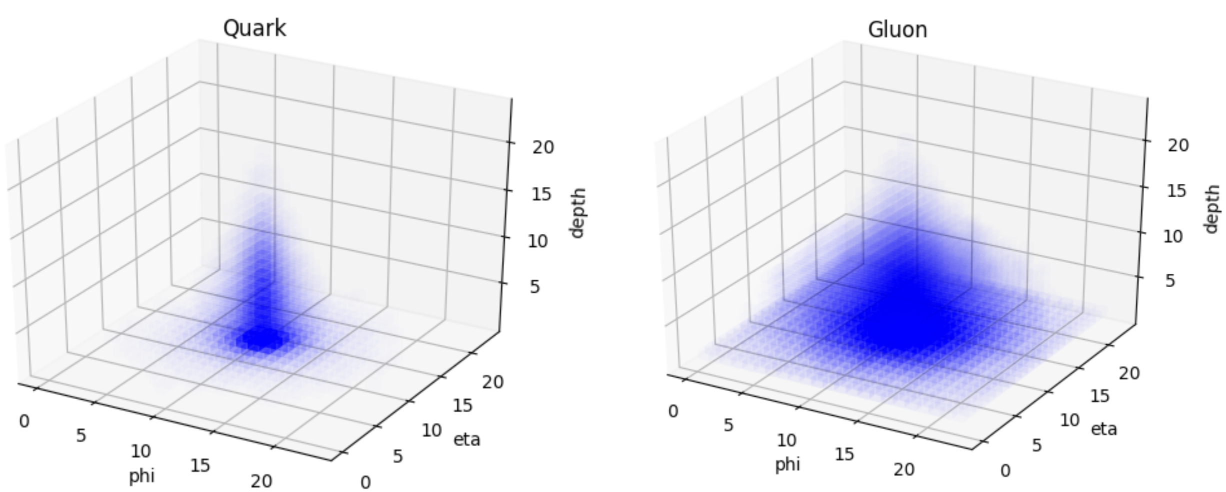
Particle ID

Discrimination power between u vs g

By Yunjae Lee



- Discriminating g from q (AUC 0.8663) based on single event image is not trivial unlike e- vs π+ (AUC 0.9996).
- Seeking for extra room for improvement, e.g. CNN using 3D voxel instead of 2D pixel images.

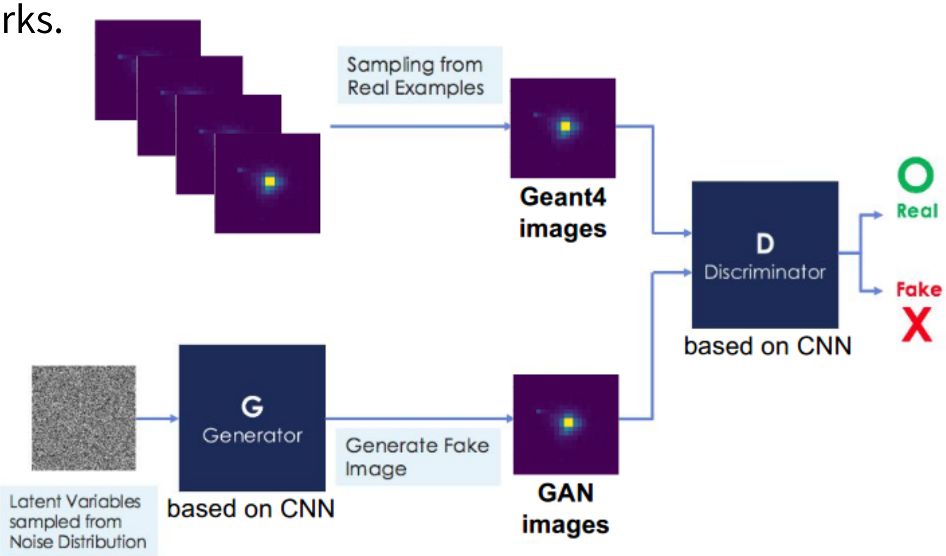


Replicating GEANT4 outputs

Replicating GEANT4 output using GAN

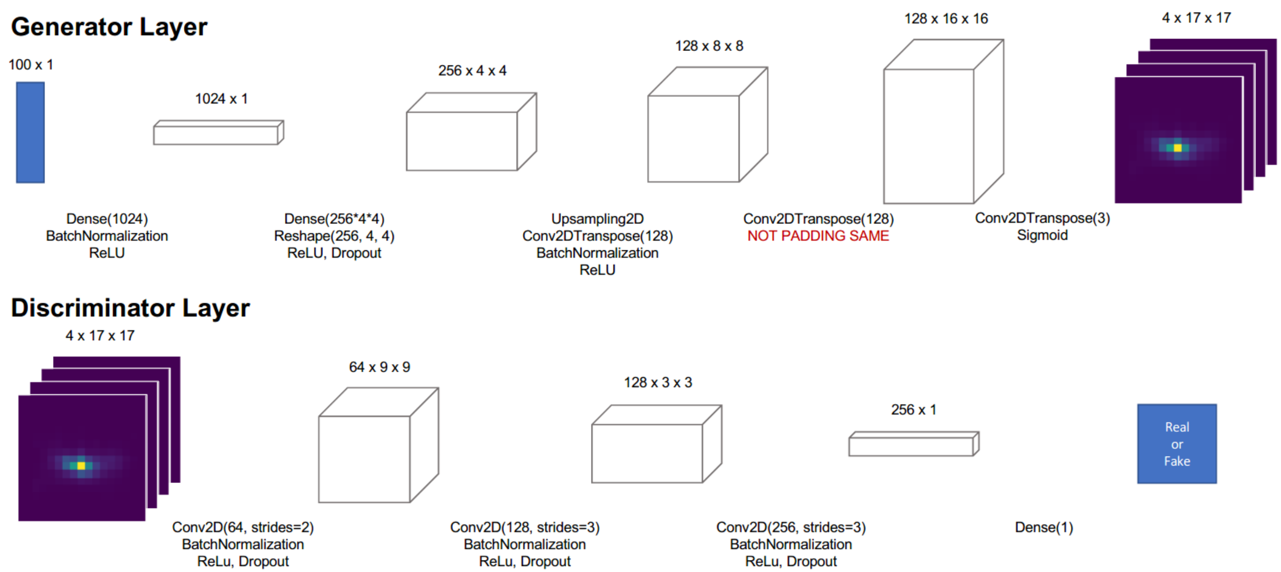
By Jongsuk Park

- Generative adversarial network (GAN) consists of two networks.
  - Generator, which generate fake images
  - Discriminator, which distinguishes real & fake images
- Generator and discriminator have adversarial relationship.
  - Generator tries to deceive discriminator.
  - Discriminator should distinguish fake & real images.



- If GAN is trained well for various scenarios, GAN can be used instead of full GEANT4 simulation.

  e.g. generating calorimeter images according to GEN information in a very short time compared to full simulation.



Summary

Development on the software framework for DR simulation

- At the moment, standalone GEANT4 application is used for the study, developed & evolved by two different groups of INFN and S. Korea for historical reason.
- Migrating to Key4HEP (and eventually centralized SW) will provide good opportunity, e.g. combined performance of detector components.
  - However, migrating in one big leap is not optimal choice.
  - Natural to start by migrating to necessary subset of Key4HEP elements (e.g. DD4HEP, EDM4HEP) based on the current standalone SW.
- Various development efforts are on-going regarding DD4HEP & EDM of DR calorimeter.

Code development efforts

- A G4VFastSimulationModel is developed to boost up speed of optical photon tracking.
- Python-based SiPM digitization code is developed, integration effort is on-going.
- Also studies on calorimeter imaged-based particle ID & replicating GEANT4's output using ML are on-going.

⋯ and a lot of opportunities are waiting on the software development!

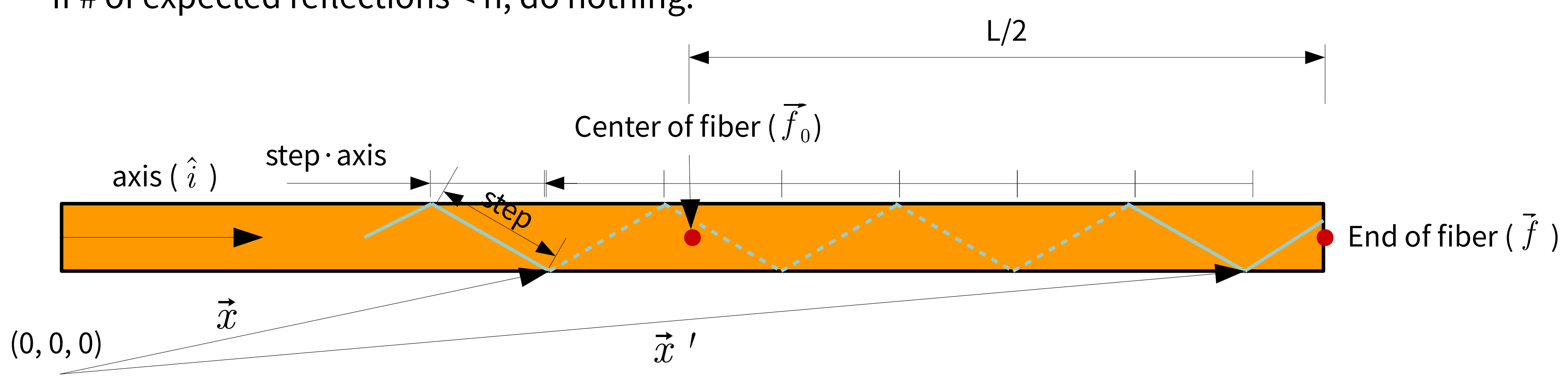
- Subscribe on egroups.cern.ch to idea-dualreadout@cern.ch
- Find meetings at Indico IDEA

Backups

Fast simulation

Estimating the target point of translation for fast optical photon transportation

- Based on the postulate that the step length of individual track remains same throughout whole transportation, the point of translation can be estimated easily.
- $\vec{f} = \vec{f}_0 + L/2\,\hat{i}$
- $\vec{f}_0$ & $\hat{i}$ can be obtained by G4TouchableHandle (touchable→GetHistory()→GetTopTransform().Inverse().TransformPoint/Axis(x, y, z))
- # of expected reflections = std::floor($\frac{(\vec{f} - \vec{x})\cdot\hat{i}}{\overrightarrow{step}\cdot\hat{i}}$)
- $\vec{x}\,' = \vec{x} + (\overrightarrow{step}\cdot\hat{i})\hat{i} \times$ # of expected reflections
- $t\,' = t$ + step/velocity $\times$ # of expected reflections
- User can require n times more total internal reflections by using (# of expected reflections – n).
  - n = 2 is sufficient to make sure everything works.
- If # of expected reflections < n, do nothing.



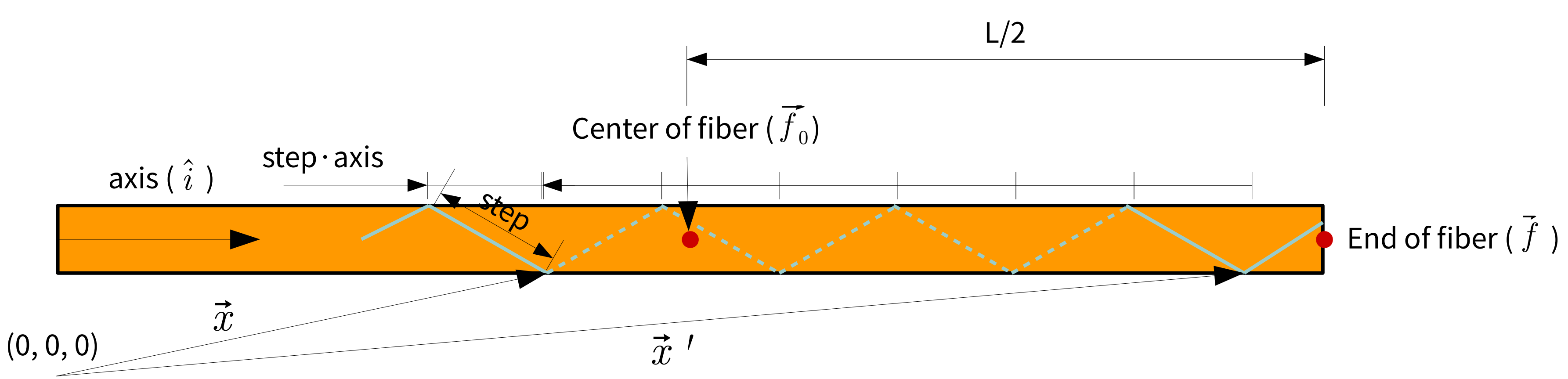
Checking absorption probability of an optical photon

- Skipping intermediate tracking of optical photon forces to check absorption probability by the model.
- In GEANT4, interaction probability with a matter of a particle is given as a 'lifetime' as a unit of interaction length.

  i.e., # of interaction length left = -std::log(G4UniformRand())
- The particle is killed when the travel length exceeds # of interaction length left.

- For a fast transported optical photon, absorption can be checked via
  - # of expected reflections × steplength / attenuation length > # of interaction length left
- Attenuation length of a material can be accessed using G4MaterialPropertyTable.

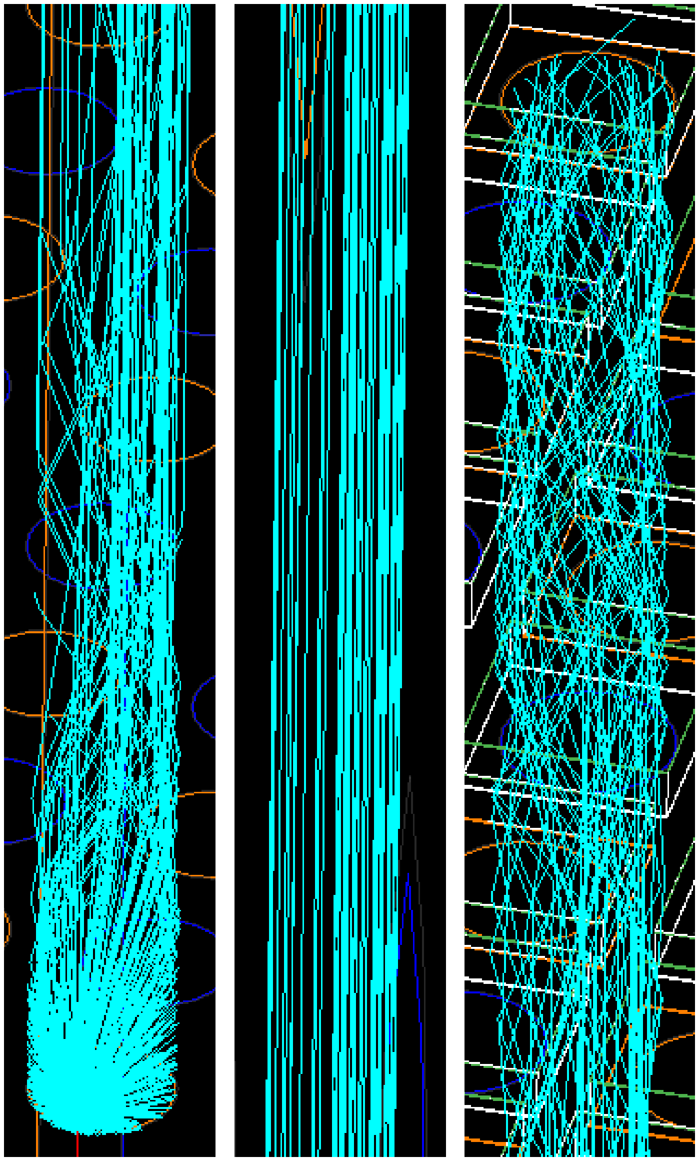
  matPropTable→GetProperty(kABSLENGTH)→Value(momentum)



Fast simulation

Demonstration of fast optical photon transportation

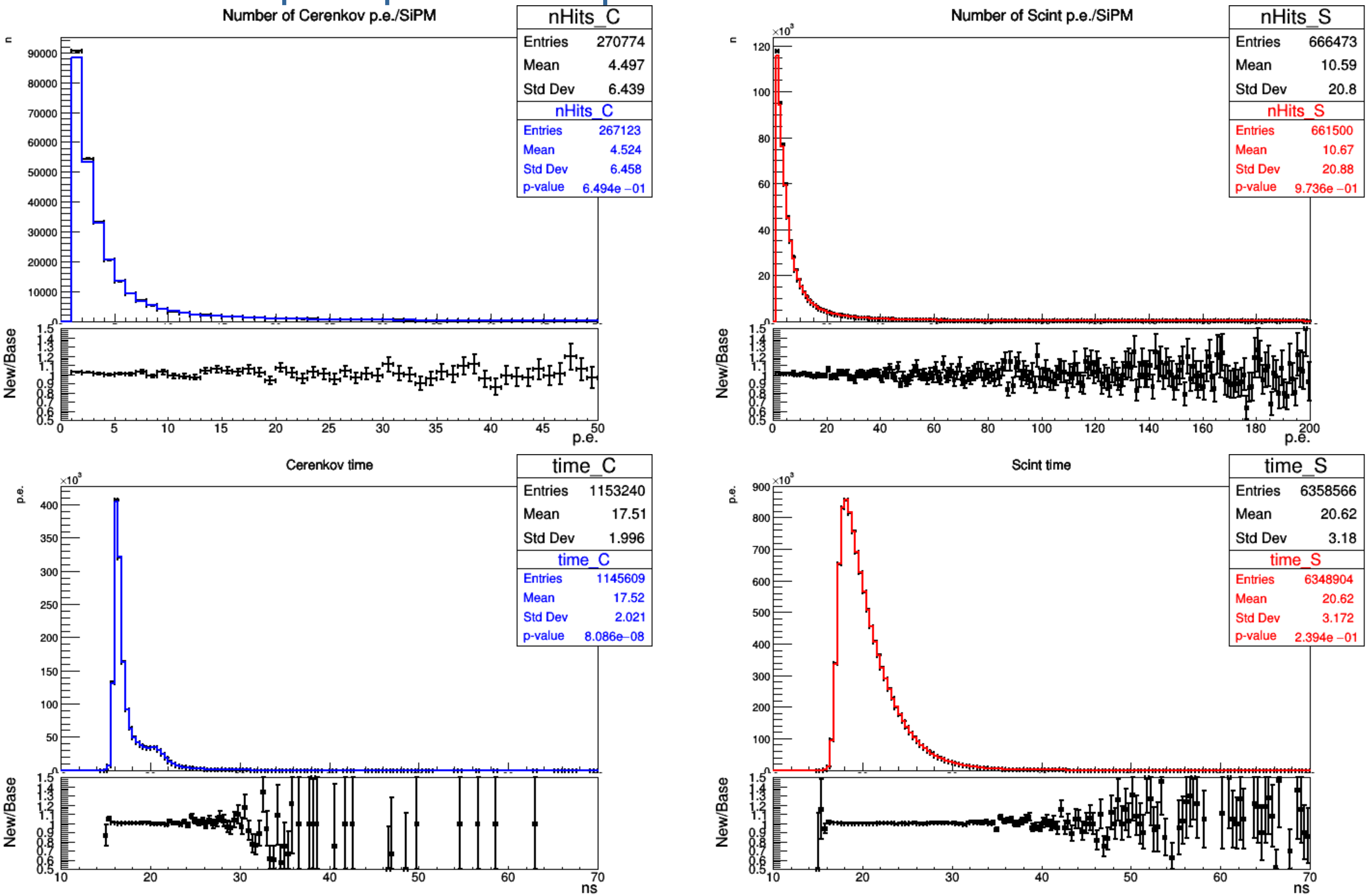
- Visualized translating tracks of optical photons from scintillation process to the end of the scintillation fiber.
- A 100 keV electron is shot to a scintillation fiber.
- The idea of translating a track using G4VFastSimulationModel works.



Validation of fast optical photon transportation

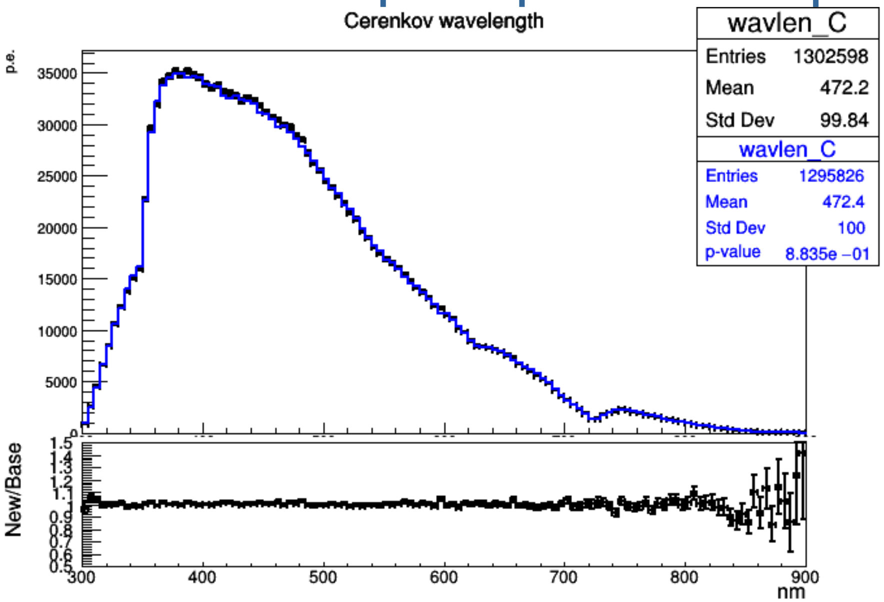
- To validate that it reproduces the result of full tracking well at the energy scale of interest, compared the distribution of optical photons detected at sensitive detector (SiPM) using 1000 of 20 GeV electron events.
- Distributions of interest, for each channel (S & C)
  - # of detected optical photons / SiPM
  - # of detected optical photons vs wavelength
  - # of detected optical photons vs global time

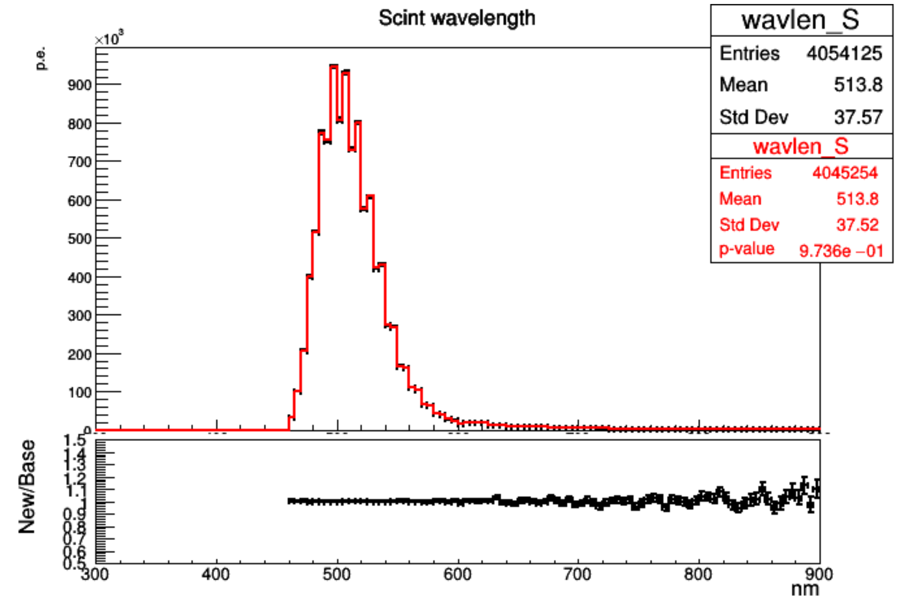
Validation of fast optical photon transportation



Fast simulation

Validation of fast optical photon transportation





Improvement in CPU consumption using fast optical photon transportation

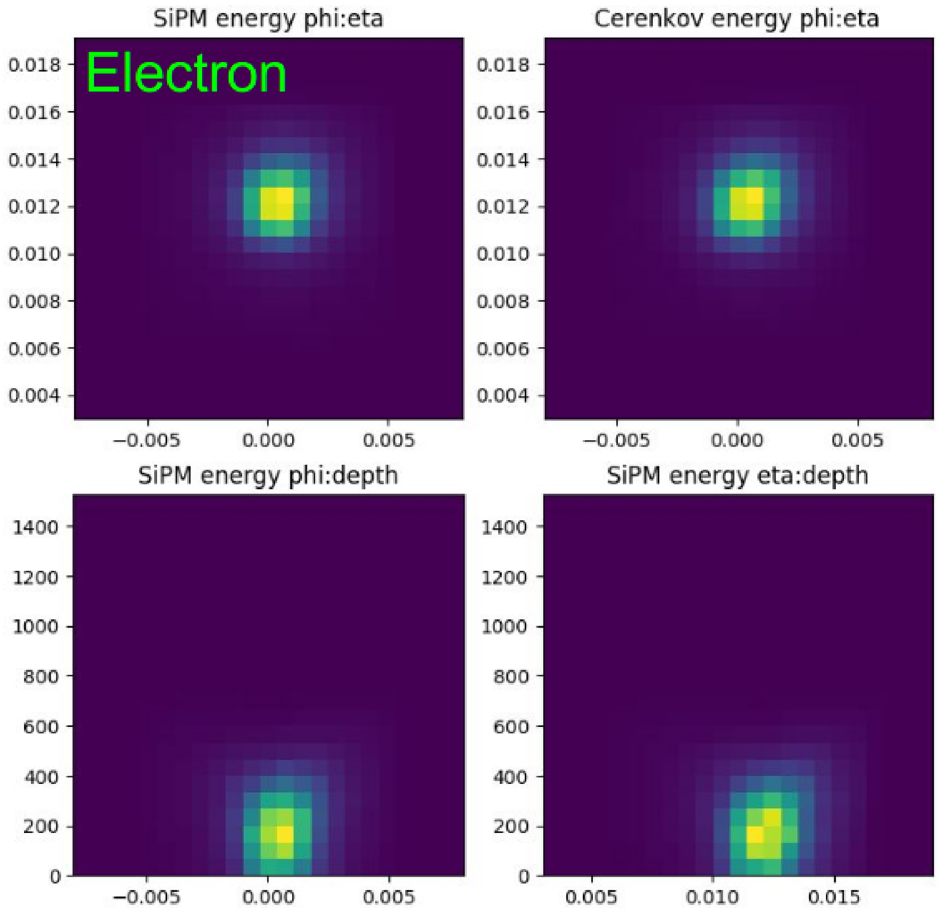
- It takes 4.62 ± 1.17 min in average to produce an event (tested with 1000 of 20 GeV electron events).
- While it was 304 ± 88 min when using full tracking with the same server.
- Almost ~ 70 times faster than full tracking!

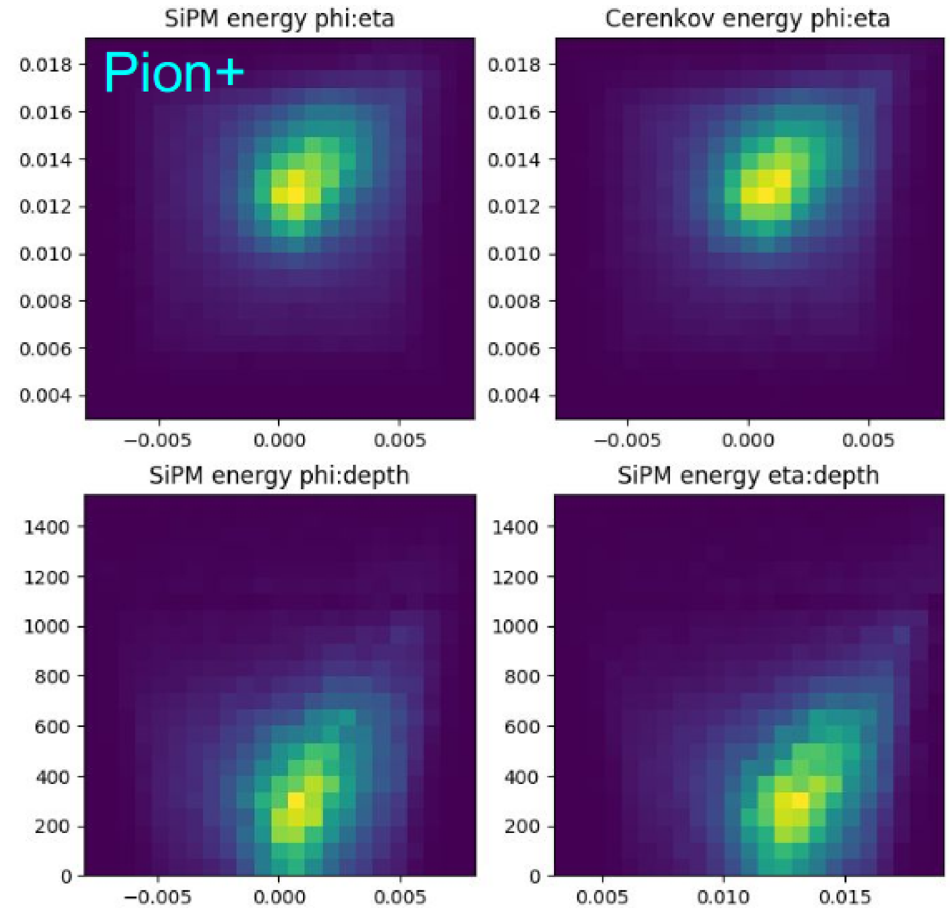
- Initial proposal of the idea was presented at GEANT4 R&D meeting [link][Github].
- Planning to promote the development as a generic plug-in or module of GEANT4 for optical fiber simulation under the supervision of GEANT4 experts.

Particle ID

Averaged images of electrons and pions

By Yunjae Lee



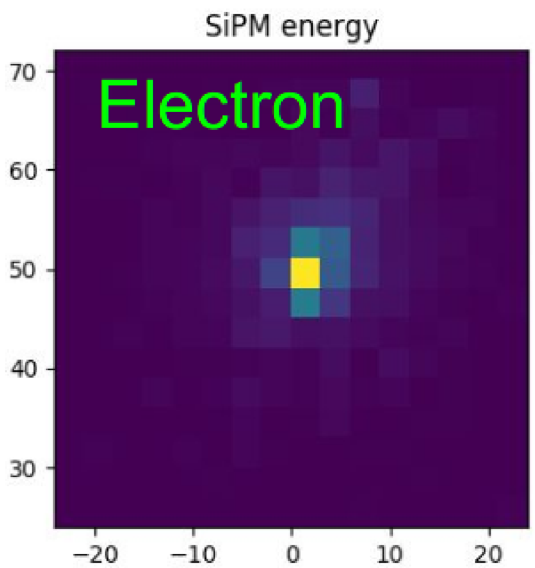


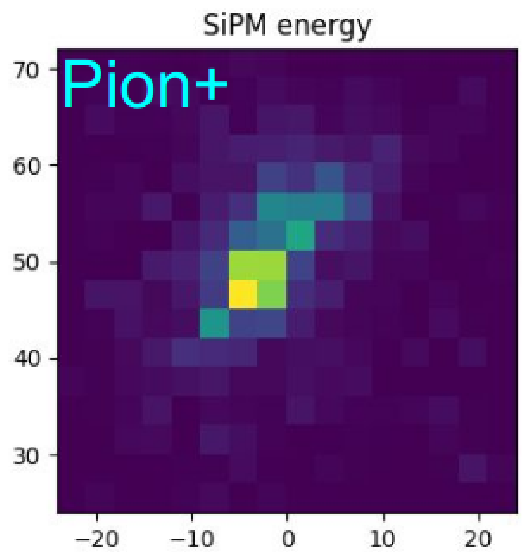
- Both electrons and pions have same energy of 20 GeV.
- Note that the beam entered the calorimeter with inclination in η direction.

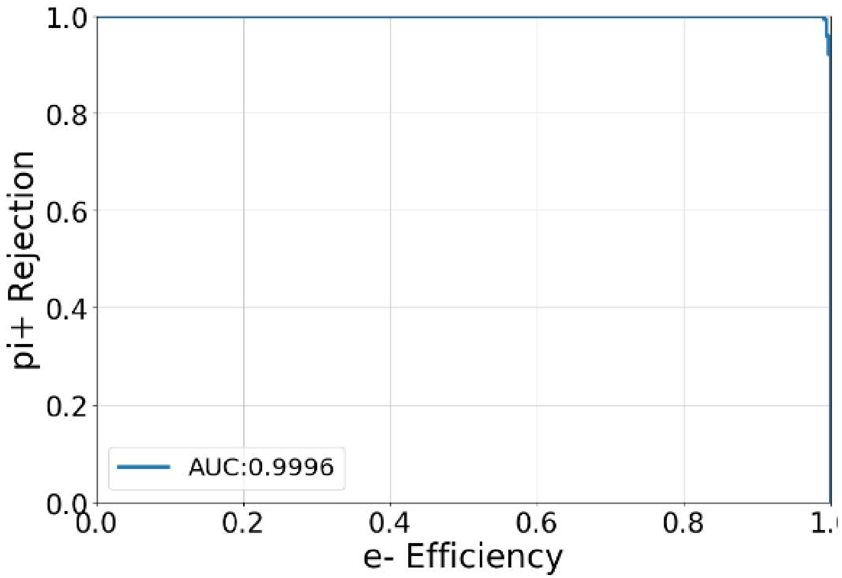
Particle ID

Discrimination power between e- vs π+

By Yunjae Lee







- The image of single electron vs pion already shows obvious difference.
- CNN is able to distinguish electrons and pions almost certainly, with area under curve (AUC) 0.9996.
  (roughly corresponds to 99.98% electron efficiency at 99.98% pion rejection.)

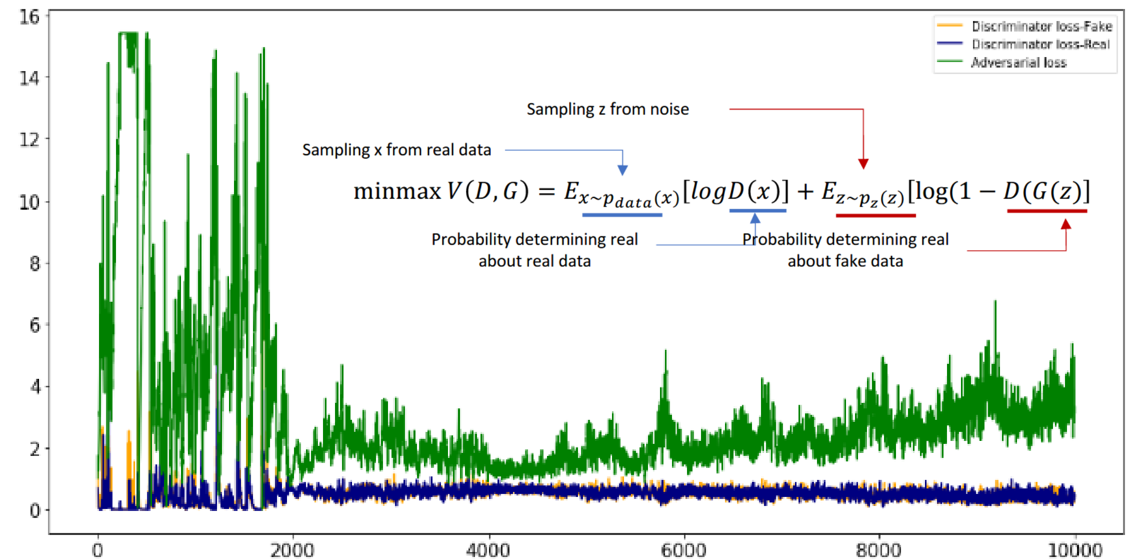
Replicating GEANT4 outputs

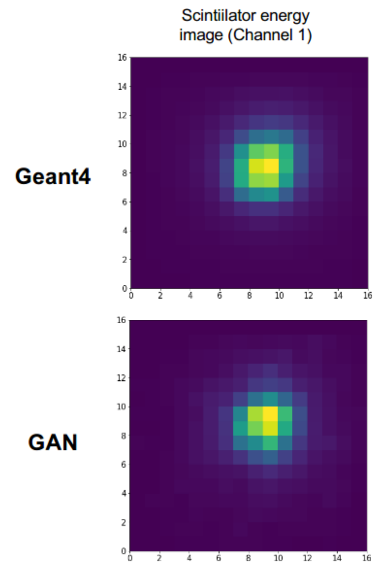
Initial performance of GAN using 20 GeV e-

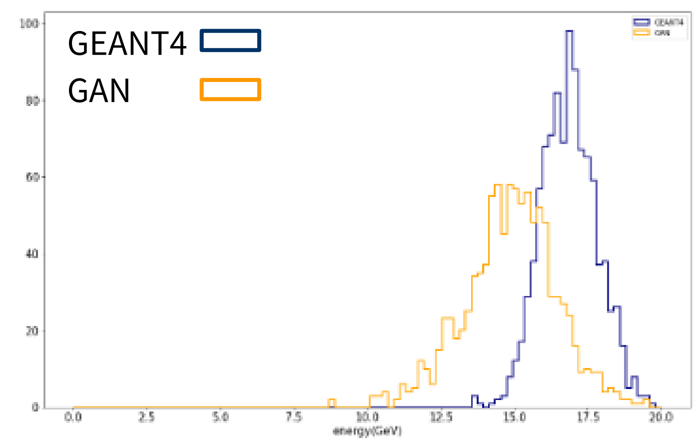
By Jongsuk Park

- Tested the performance of GAN using 1000 events of 20 GeV electrons.
- Minimax algorithm – generator and discriminator have adversarial relationship in the loss function.
- For a image of the calorimeter, GAN started to mimic GEANT4 outputs.

- However, in the view of actual energy distribution, GAN has still need to be improved significantly.
- Training using large statistics with parameterized energy from fast optical photon simulation in the pipeline.







Scintiilator energy distribution (Channel 1)

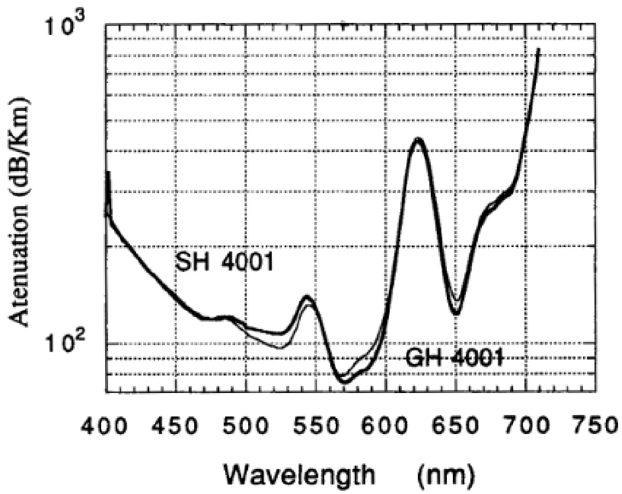
Material properties

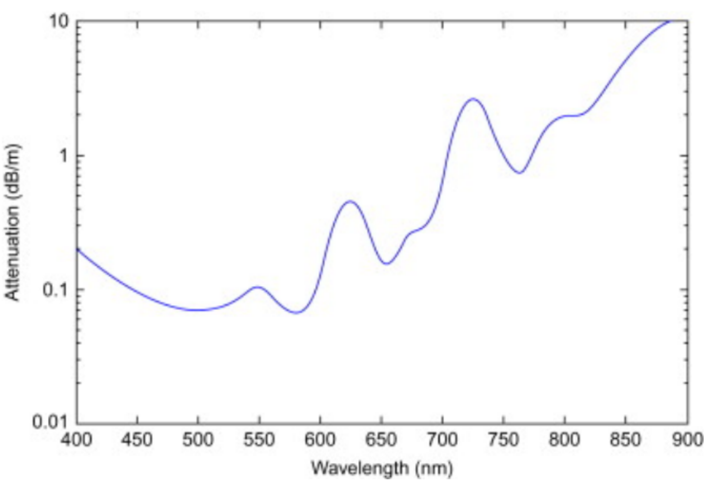
Photon energy

- The energy window of optical photons is set to 900-300 nm (1.37760-4.13281 eV) with 25 nm step.

PMMA

- RI
  - refractiveindex.info (G. Beadie, M. Brindza, R. A. Flynn, A. Rosenberg, and J. S. Shirk. Refractive index measurements of poly(methyl methacrylate) (PMMA) from 0.4-1.6µm, Appl. Opt. 54, F139-F143 (2015))
- Attenuation
  - sciencedirect (Silvio Abrate, Handbook of Fiber Optic Data Communication (4th Ed.), 2013)
  - Eska POF manufacturer





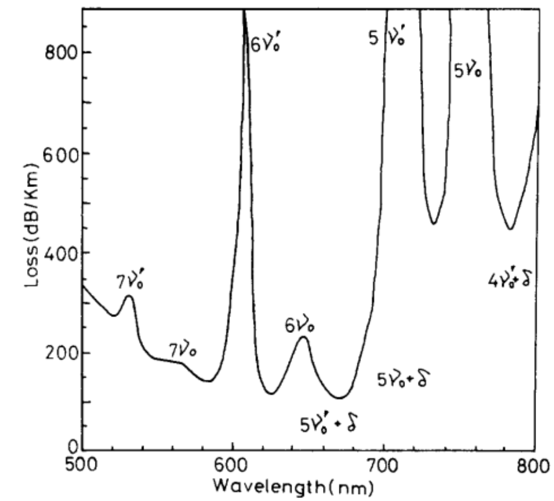
Material properties

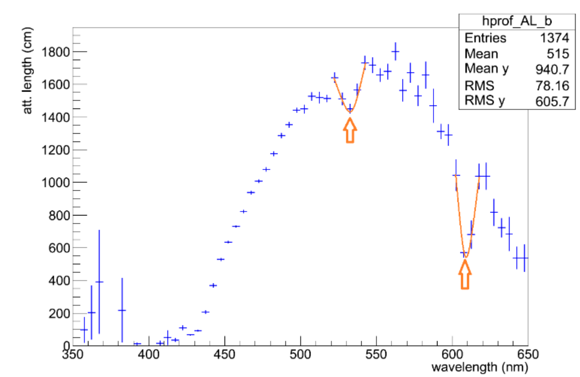
Fluorinated polymer

- RI
  - RD52 paper (N. Akchurin, et al., Nuclear Instruments and Methods in Physics Research, A762 (2014), pp. 100-118.)
  - Set to single value (1.42).

Polystyrene

- RI
  - refractiveindex.info (N. Sultanova, S. Kasarova and I. Nikolov. Dispersion properties of optical polymers, Acta Physica Polonica A 116, 585-587 (2009))
- Attenuation
  - J. Applied Physics (T. Kaino, M. Fujiki, and S. Nara, Low-loss polystyrene core-optical fibers, Journal of Applied Physics 52, 7061 (1981))
  - LHCb-PUB-2015-011, 012 (SCSF-78 LHCb Sci-Fi tracker R&D TDR)
  - kuraray scintillating fiber manufacturer (SCSF-78)

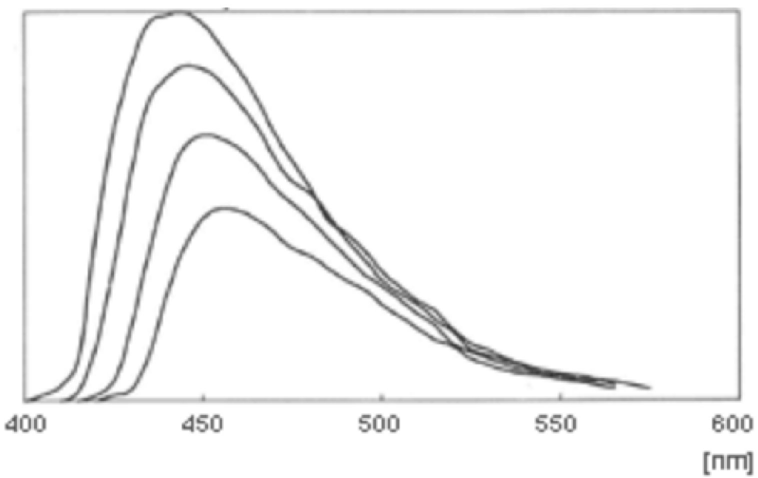




Material properties

Polystyrene

- Emission spectrum, decay constant
  - kuraray scintillating fiber manufacturer (SCSF-78)
  - Decay constant = 2.8 ns
- Birks constant
  - k_B = 0.126 mm/MeV

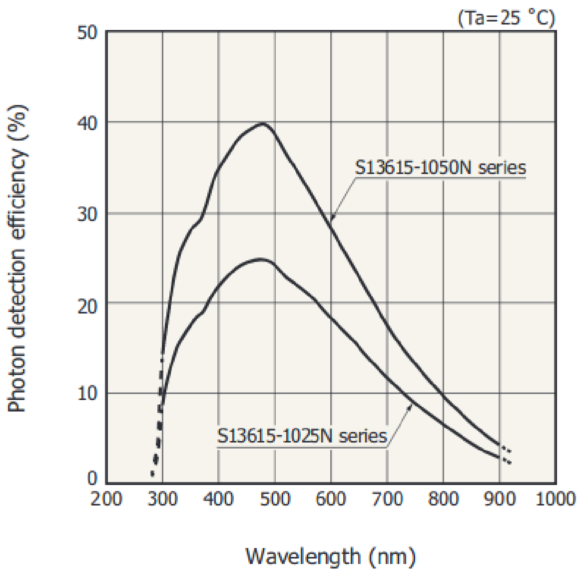


Glass, Air

- RI
  - 1.52, 1.0
- Attenuation
  - 420 cm, N/A

PDE (Photon Detection Efficiency)

- Hamamatsu S13615-1025N series



Title

Text

$$formula$$