

“RandomNoise” Implementation for trackers in ElCrecon

Reconstruction Group Meeting
11. 08. 2025 (MON)

Minjung Kim (UC Berkeley & LBNL)

Motivation


📡 Electronics noise is unavoidable in silicon detector readout and can lead to fake hits.

🎯 To realistically evaluate reconstruction performance, we model noise hits.

🔍 This allows for direct comparison with ideal (noise-free) simulations, estimating impact of fake hits.

🔧 We provide a unified noise injection method for silicon tracker subsystems (BVTX/BTRK/ECTR), which can be expanded to other readout systems.


Concept

 **Inputs:** EventHeader (ensures reproducibility of random generation) + detector geometry directly subscribed

 **Noise generation:**

 Poisson-distributed number of hits per system

 Uniform spatial distribution within modules.

 Optional time spread (digitization-dependent; for the future with TimeFrame rather than Event)

 **Outputs:** noise-only collection

+ Merging: CollectionCollector combines original raw hits and noise hits to create RawHitsWithNoise.

Implementation Overview

Algorithm Flow:

1. RandomNoise Algorithm

 Caches geometry and readout info in `init()`.

 Initializes RNG with seed from `EventHeader`.

 Generates noise hits.

2. CollectionCollector

 Merges original raw hits and noise-only hits.

3. Factory Integration

 **+** Adds noise factories and merged collections per detector plugin.

Detailed RandomNoise Algorithm (1)

Initialization (init() method)



Loads detector geometry from DD4hep

- Uses `algorithms::GeoSvc::instance().detector()` to obtain a pointer to the global `dd4hep::Detector` instance.
- If the geometry service is missing, logs an error and stops initialization.



Identifies readout segmentation type and extracts sensitive cell IDs



Stores module boundaries and channel counts for later noise hit generation

Detailed RandomNoise Algorithm (1)

Initialization (init() method)



Loads detector geometry from DD4hep



Identifies readout segmentation type and extracts sensitive cell IDs

- Resolves the readout by name (`m_cfg.readout_name`) from the loaded geometry.
- Iterates through all detector elements (`m_dd4hepGeo->detectors()`), retrieves their SensitiveDetector, and checks if their associated readout matches the requested one.
- Adds matching DetElement objects to `m_targetDets` for later processing.



Stores module boundaries and channel counts for later noise hit generation

Detailed RandomNoise Algorithm (1)

Initialization (init() method)



Loads detector geometry from DD4hep



Identifies readout segmentation type and extracts sensitive cell IDs





Stores module boundaries and channel counts for later noise hit generation

- For each target detector element, calls **ScanDetectorElement()** to recursively traverse the geometry tree and collect **ID paths (volume ID maps)** down to the deepest sensitive volumes.
- Calls **ScanComponent()** to determine **segmentation bounds (min/max values for each ID field)** by probing geometry cells via segmentation-specific scanning functions (Cartesian, Cylindrical, Polar, or generic Monte Carlo sampling).
- Caches both idPaths and bounds in m_idPathsCache and m_boundsCache so the event loop (process()) can generate noise hits efficiently without re-scanning geometry each event.




Detailed RandomNoise Algorithm (2)

Noise Hit Generation:

Random Engine Setup:

-  Uses EventHeader information (run number, event number) to generate a unique, reproducible seed.
-  Creates a local `std::mt19937` engine per event to ensure thread safety.


Noise Hit Generation:

-  Determines the number of noise hits from a Poisson distribution using `n_noise_hits_per_system` parameter.
-  Randomly selects modules and positions within those modules; duplicate check
-  Assigns amplitude, time, and cell IDs consistent with detector readout mapping.

Detailed RandomNoise Algorithm (3)

Output Handling:

 Fills EDM4hep **RawTrackerHit** objects for each noise hit.

 Sends noise-only collection downstream; CollectionCollector merges noise with physics hits later on.

Names & Parameters

Detector	Noise Factory (collection) Name	Main Parameters
BVTX	SiBarrelVertexNoiseRawHits	addNoise, n_noise_hits_per_system
BTRK	SiBarrelNoiseRawHits	addNoise, n_noise_hits_per_system
ECTRK	SiEndcapTrackerNoiseRawHits	addNoise, n_noise_hits_per_system

- 📌 All use **EventHeader** as input tag
- 📌 Relevant collections added in JEventProcessorPODIO
- 📌 n_noise_hits_per_system (per event at this moment)

For the future with TimeFrame, noise rate (in Hz) with time spread would be ideal

Summary and outlook

🏁 Implementation of reproducible noise injection algorithm; added for BVTX, BTRK, and ECTRK. Also applicable to other readout systems.

🏁 Most of suggestions/feedback for PR#1988 implemented! *Thanks for reviewing!*

🕒 Full usage will be shown in tracking meeting; impact of noise on tracking performance.

🕒 Updated required for upcoming TimeFrame based data structure

? Exclude duplicate noise hit?

? More feedback/suggestions? i.e. amplitude?

Summary and outlook

- 🚩 Time injection algorithm; added for BVTX, BTRK, and
- 🏁 Most of
- 🕒 Full usage will be shown
- 🕒 Updated required for upcoming Timer
- ? Exclude duplicate noise hit?
- ? More feedback/suggestions? i.e. amplitude?

Last minute bug found :(

Thanks for reviewing!

Smoothing for seed 47491 and

WARNING: vol_id (xxxx) not

Get Back to you sooooooon in next commit!