Ideas for a new 3D LAr Pixel Simulation

Brett Viren

November 12, 2024

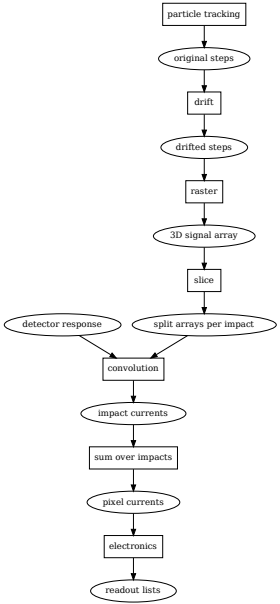
Overview

- Present the basic problem, data domains and calculation overview.
- Required computation resource estimation and optimization.
- Various issues

Major components and data

Basic operations:

- particle **tracking** → **original steps**
- **drift** → **drifted steps**
- **raster** steps → **signal voxels**
- **partition** into per-impact **array slices**
- **response** to signal → **pixel impact currents**
- **sum** over impact slices → **pixel currents**
- **electronics** → **readouts**



The core calculation

$$M = R * S + N$$

- M the "measurement" represents a family of **induced current waveforms** input to **pixel readout electronics**.
- S the "signal" represents the distribution of **ionized electrons** after **drifting to the response plane**.
- R the "response" operator **transforms** S to the domain of M. With pixels, it is "simply" the **field response**. With wires, "electronics response" is also included. In both cases, the operation is **convolution and sum**.
- N some generator of **noise** in the measurement domain (will ignore this today).

Data forms

$$M = R * S + N$$

Define **multi-dimensional array** representation with values **binned** on **uniform grids**.

- A 3D "voxel" defined as (2D transverse spatial bin) $\times$ (1D longitudinal / drift / time bin).
- S gives ionization charge per voxel,
- M gives induced charge per voxel,
- R transforms from the space of S-voxels to the space of M-voxels.

Choose use common sizes for S and M 3D voxels. For example:

- $V_y \times V_z = (0.5\,\mathrm{mm})^2 = (5\,\mathrm{mm}/10)^2$ spanning a **sub-pixel** area.
- $V_x \equiv T = 100\,\mathrm{ns} \equiv D/v = \frac{0.16\,\mathrm{mm}}{1.6\,\mathrm{mm}/\mu\mathrm{s}}$ longitudinal / drift / time spanning an FR **step**.

S and M spaces need not be so coincident. We may need to break this symmetry in the name of optimizing.

Response voxelization

Field response calculations give us

- **instantaneous current** at **discrete steps** along a **single path** starting at a **point**.

Voxelization requires **bins** and we must:

- Form an average/integral of the response values over the sub-pixel area.
- Integrate over time step and go from induced **current** to induced **charge**.
  - Practically, this requires simply multiplying: $Q = I \cdot T$.

We now assume R is voxelized in some manner.

- Doing this efficiently is a core problem.

A drawing would be nice here…

Interlaced partitioning

Up to now we think of S and R defined on a large, fine-grained, sub-pixel grid spanning pixels.

- But, M must be defined on a full-pixel grid and each pixel in fact **sums** over their area.

The domain for S and R is actually an **interlacing** of a **partitioning** of the sub-pixel grid.

- We partition (slice) signal arrays: $S \to S_p$ for each partition p and *etc* for R.

Each partition selects a 2D array of a family of 1D waveforms corresponding to common relative sub-pixel locations.

A drawing would be nice here…

Response operator

$$M = R * S + N$$

- So, $R*$ operator becomes a sum of 3D convolutions, each over the one impact partition.
  - $R * S \rightarrow \{(R_p * S_p)\}$
  - $Q = \sum_p (R_p * S_p) \Rightarrow$ 1D waveforms $\times$ 2D pixel grid.

Electronics readout

This requires a straight-forward model for the pixel electronics.

$$E(\{Q(t)\}) \rightarrow \{(q_i, t_i), ...\}$$

Given an induced charge waveform input at one pixel, return a list of q/t readout pairs.

Approximate array sizes

Response domain

- 2D grid of **response pixels** spanning all response path starting locations.
  - $N_{res,y} \times N_{res,z} = 9 \times 9$.
- 1D grid of **response ticks** spanning response path steps.
  - $N_{res,t} = 1000$.

Signal and measurement domains

- 2D grid of **measurement pixels** in one detector "unit".
  - $N_{pix,y} \times N_{pix,z} = 100 \times 100$.
- 1D array of **measurement ticks** spanning the max drift time.
  - $N_{tick} = 10000$.

Impact partitioning

- 2D grid of **sub-pixels per pixel** matching response path **binning** (not points).
  - $N_{imp,y} \times N_{imp,z} = 10 \times 10$.
- A **universal clock** of drift period $T = 100\,\mathrm{ns} \equiv D/v = \frac{0.16\,\mathrm{mm}}{1.6\,\mathrm{mm}/\mu\mathrm{s}}$

Array sizes

- $N_R = (N_{imp,y} \times N_{imp,z})(N_{res,t} \times N_{res,y} \times N_{res,z}) = (10 \times 10)(1000 \times 10 \times 10) = 10^7$
- $N_S = (N_{imp,y} \times N_{imp,z})(N_{tick} \times N_{pix,y} \times N_{pix,z}) = (10 \times 10)(10000 \times 100 \times 100) = 10^{10}$
- $N_M = (N_{tick} \times N_{pix,y} \times N_{pix,z}) = (10000 \times 100 \times 100) = 10^8$

Each 3D convolution:

- Size: $(1000 \times 10 \times 10) * (10000 \times 100 \times 100) \rightarrow (10000 \times 100 \times 100)$
- Must sum each 400 MB result over (10×10) convolutions.
  - If (10×10) convolutions are serial, can accumulate in fixed 400 MB.
  - If convolutions are in parallel we must store 40 GB before the sum.
    - ⋆ Or perhaps can make store the 400 MB arrays sparsely.

Some timing estimates

Using one signal partition $S_p = (10000 \times 100 \times 100) = (2 \cdot 5)^8$ (small primes)

operation	DFT	mult	add
CPU-ms	548	31.1	32.2
GPU-ms	8.36	0.85	0.84

- Time in millisecond, excludes any CPU/GPU transfer
  - Round trip CPU-GPU-CPU requires 53 ms
- PyTorch and `ipython`'s `%timeit` and `torch.set_num_threads(1)`
  - Care taken to `detatch()` from autograd.
- Both CPU and GPU results utilize 1 CPU core. GPU sees 100% util.

Some memory estimates

Voxelize steps into full S then partition $S \to \{S_p\}$

- Easiest but requires 40 GB array

Voxelize steps into per-step arrays, partition and accumulate into S_p

- Assume raster each step to in $(5\,\mathrm{mm})^3 = (10 \times 10 \times 10) + (1 \times 1 \times 1)$ location.
- With $\mathcal{O}(10^6)$ steps, 1 GB memory needed. Could easily remain in GPU RAM.

Voxelization time

Voxelization is a many (steps) to many (voxels) accumulation problem.

- No simple SIMD type algorithm exists.
- HEP-CCE/PPS discovered this when accelerating WCT
  - A divide-and-conquer method was developed.
- Some “real” work needed to estimate how much step costs here.
  - External choice can impact this, eg **chunking**.

Chunking

Break up the problem domain, solve problem on smaller domains, connect the results.
For example,

- $(10, 10, 10) = 1000$ chunk c problem signal $S^c = (10, 10) \times (1000, 10, 10))$
- Size: $10^{10} \rightarrow 10^7$ or 40 GB $\rightarrow$ 40 MB.

Plus extra memory and compute to assure no cyclic artifacts

- Enlarge convolution to be size of S^c + size of R.
- Apply the overlap-add or overlap-save alg to add this enlargement to proper neighbors.

More voxelization issues

The “Gaussian hotdog”

- A 3D Gaussian $(\sigma_L, \sigma_T, \sigma_L)$ with $\sigma_x(\Delta t)$ is convolved with a 3D step line.
- Consider binning Δt coarsely and pre-tabulating Fourier representation of each Gaussian.
- The “hotdog” is then formed by multiplication with Fourier representation of a step line.

Reduce steps to the usual “depos”

- The min step size in G4 is $\mathcal{O}(1\,\mathrm{mm})$ and nominal voxel is $(0.5\,\mathrm{mm})^3$
  - Maybe step line shape is not important?
- Or, perhaps bin possible step directions to make lookup of Fourier representation.
  - This meshes well with the hotdog optimization above.

Time bases

We chose a universal $T = 100\,\mathrm{ns}$ which already drives problem size. But,

- Characteristic diffusion time is $\mathcal{O}(1\,\mu\mathrm{s})$.
- FRs need $T_{FR} = 50\,\mathrm{ns}$ at least to avoid being undersampled.
- Real electronics clock is $100\,\mathrm{ns}$.
  - But, is there an effective low-pass filter cutoff at lower frequency?

Consequences / options:

- Double the problem size with a universal $T = 50\,\mathrm{ns}$
- Keep $T_{FR} = 50\,\mathrm{ns}$ but do proper resampling to T as part of FR convolution.
  - Ie, we think of diffusion as a low-pass filter.
- Choose $T > 100\,\mathrm{ns}$ if downstream electronics has an effective low-pass filter.