Differentiable Water Cherenkov Simulator Plans

Omar Alterkait

CIDeR-ML General Meeting

11/20/23

Outline

- This work will be in collaboration with Cesar
- Start by giving a plan on the differentiable simulator
- Introduce Taichi-Lang

Why a Differentiable Detector Simulator?

- Gradients using automatic differentiation
- Enables more precise sensitivity analysis and optimization of detector parameters
- Enhances reconstruction ML model training and inference efficiency
- Reduces reliance on extensive MC simulations
- Allows continuous, data-driven calibration, aligning the simulator closely with real detector behavior
- With modern libraries, we can parallelize this process (especially for GPUs) and run it faster

Steps Needed for WC Diff

- Have to decide what representation we want the data to be in (voxel or not)

Generate Photons

- Cut up the track into smaller segments that fill up the voxel
- Generate photons proportionally to the length of the track inside that voxel segment
- Photon direction is sampled from a cone with the appropriate Cherenkov angle

Propagate Photons

- While the photons are in the medium, propagate them over time
- Include effects such as scattering and attenuation
- When the photons hit a PMT or a wall, there is a chance of reflection or loss

Detect Photons

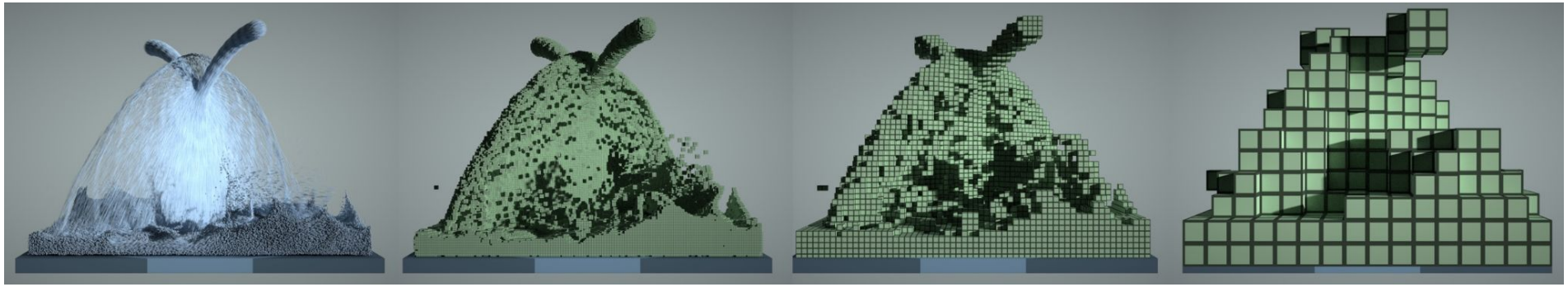
- Have some detection based on the angle of interaction of the photon with the pmt
- Will start off with just summing all the detected photons that are not reflected or list, but could include more effects in the future

Choice of Implementation Language

- We need a language that enables automatic differentiation
- Need to decide on an efficient way to implement it
- Considering that data is very spatially sparse, a sparse representation of the data would be useful.
- Some common ML languages used (jax, torch) are useful, but don't have full support for sparse rendering and propagation
- This is where Taichi-Lang comes in

Taichi-Lang

- https://www.taichi-lang.org/
- High performance parallel computing language in python that builds code for either CPUs/GPUs. Performance comparable to fine tuned cuda implementations
- Has spatially sparse data structures for efficient implementation. Their code is so fast, that they have shown real-time fluid simulation.
- Includes automatic differentiation that is faster than Pytorch/Jax.



Next Steps

- Look into taichi and its documentation a little more
- Work with Cesar, and implement the parts discussed above
- Try to get some parts done before the collaboration meeting. But if not, this differentiable simulator should be mostly implemented by the end of the collaboration meeting.

Questions