CIDER Weekly Updates

Omar Alterkait

Graduate Student, Tufts University

11/8

- During the workshop, made the photon_propagation fast. But this meant it was not differentiable.
- The quality of gradients we got from the simulation had some problems.
- So I wanted to look into making propagate_photons differentiable and fast

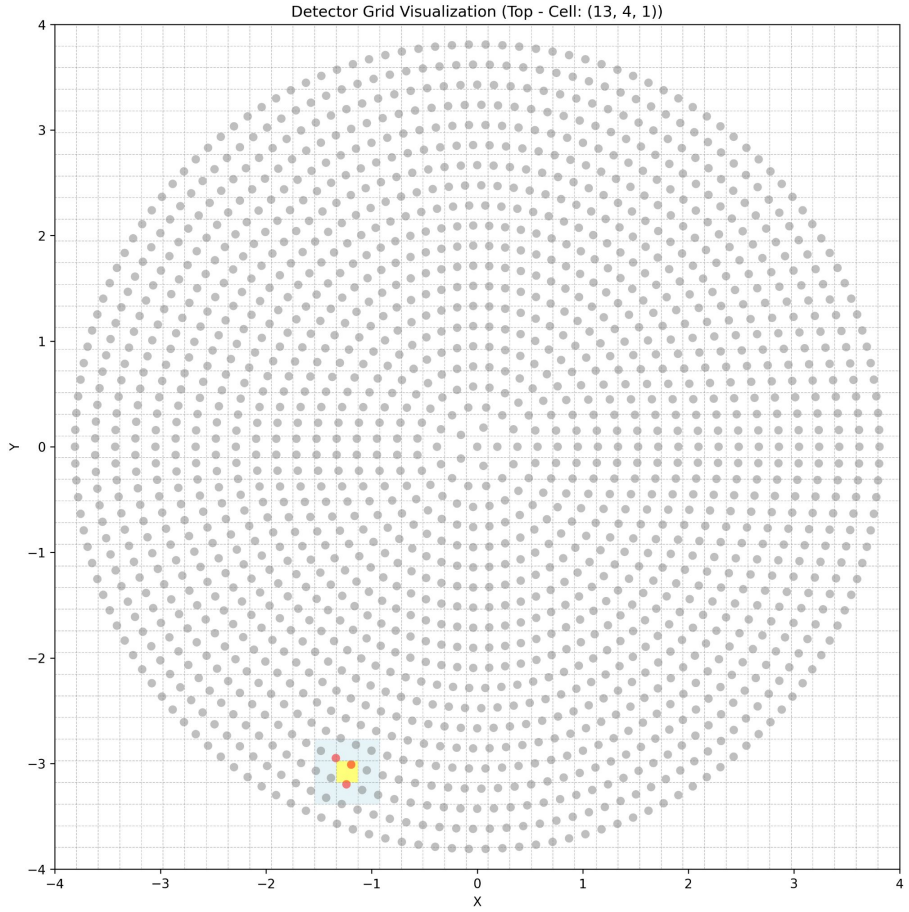
```python
import jax
import jax.numpy as jnp
from functools import partial

def create_event_simulator(propagate_photons, Nphot, NUM_DETECTORS, detector_points, full_output=True):
    @jax.jit
    def _simulate_event_core(params, key):
        cone_opening, track_origin, track_direction, initial_intensity = params

        # Normalize track direction
        track_direction = track_direction / jnp.linalg.norm(track_direction)

        # Generate photons
        photon_directions, photon_origins = differentiable_get_rays(track_origin, track_direction,
         cone_opening, Nphot, key)

        # Propagate photons (non-differentiable)
        propagation_results = jax.lax.stop_gradient(propagate_photons(photon_origins, photon_directions))

        # Extract results
        hit_indices = propagation_results[:, 0].astype(jnp.int32)
        travel_times = propagation_results[:, 1]
```

To do this, we use the grid system like before to filter which PMTs to check, and then do the calculations with all of them instead of just keeping the closest.

This was a little problematic to try and get. But I was able to do it

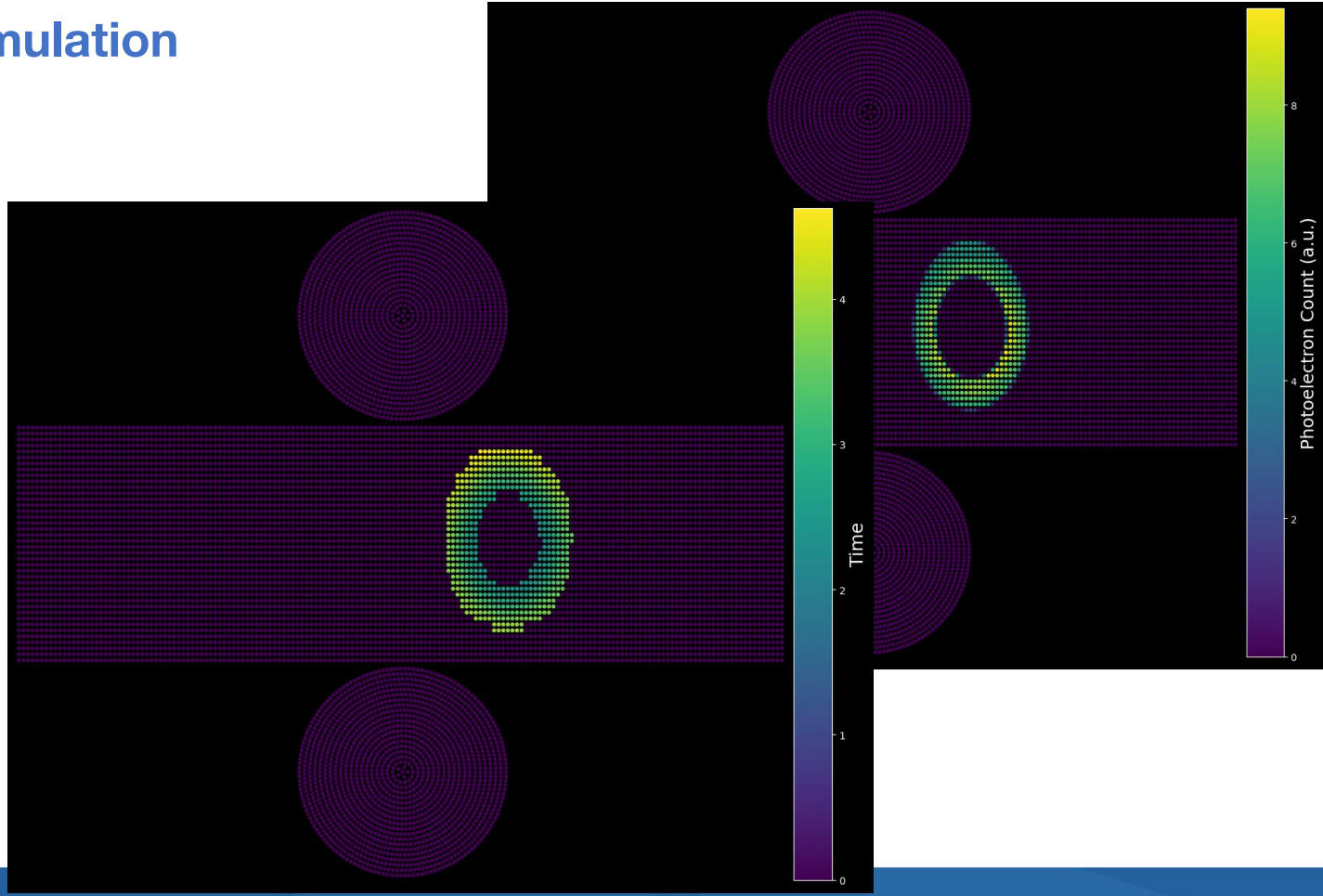
So now we have a temperature parameter to control how much smoothing we have (lower temperature means more neighboring PMTs get some charge as well)

We now get a tradeoff between speed and how many pmts we check



Results of new simulation

With high temperature, we get the same results as before. Speed is still a few ms for the calculation

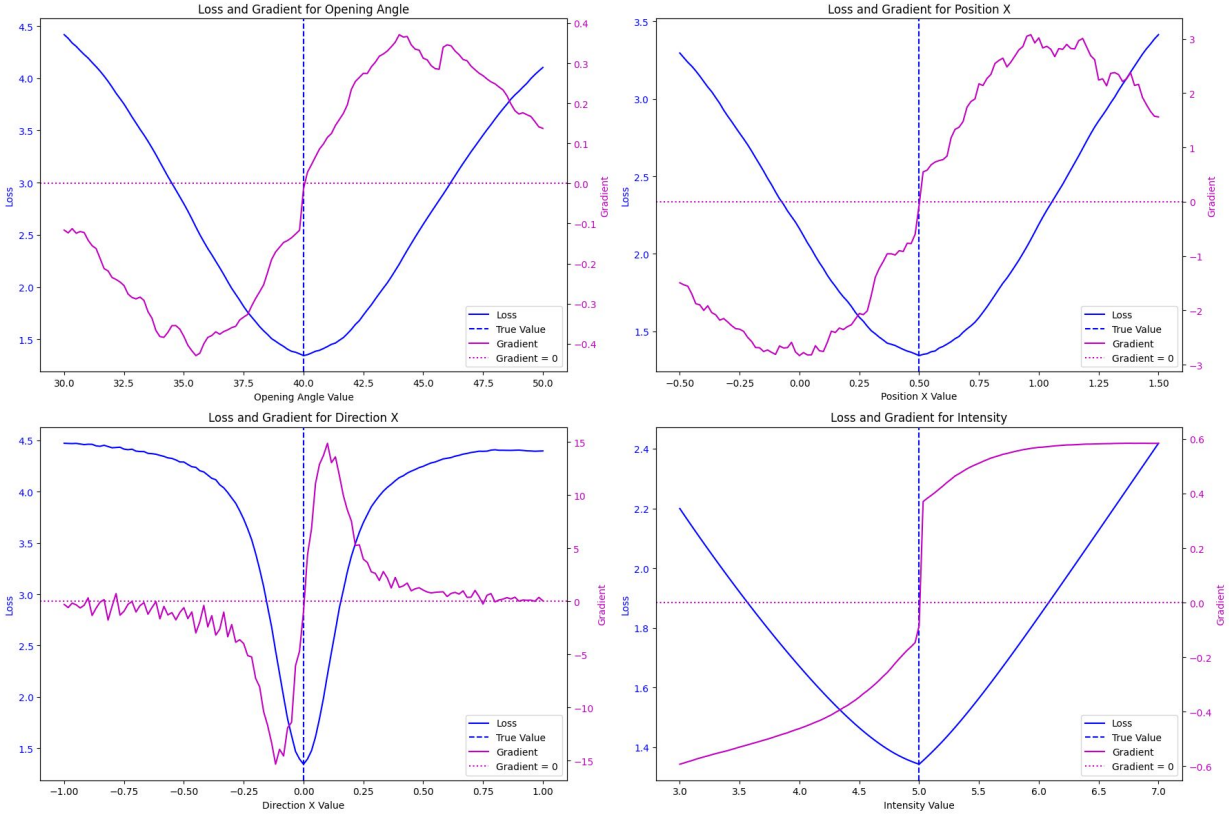


Gradient Results

We get good gradient results. These are all results without time.

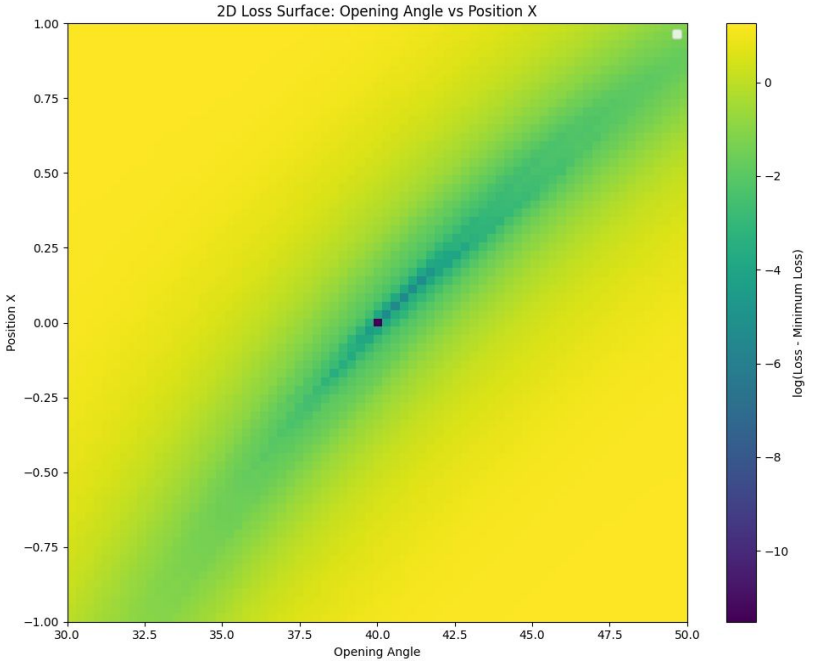
With time, there are some slight problems, like before of doing two things:

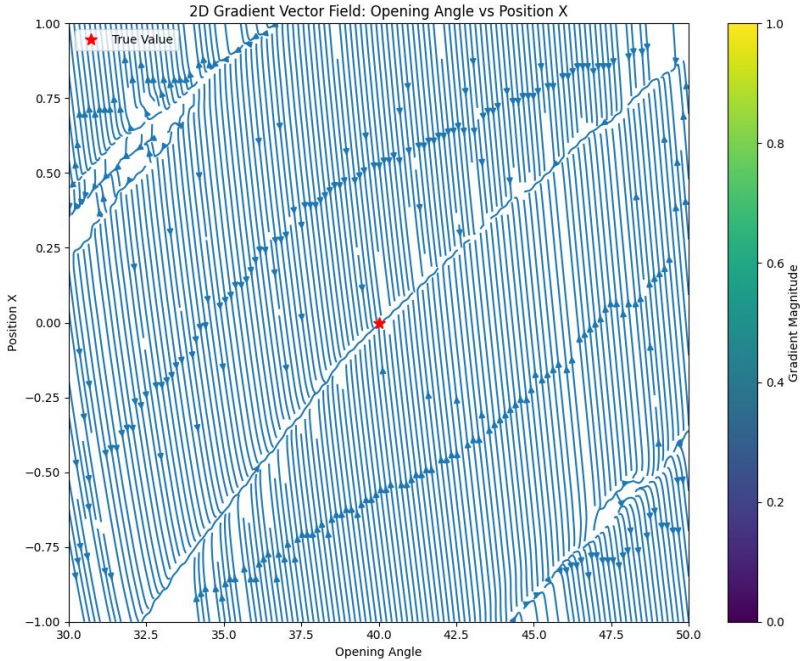
- First biases the opening angle gradient for some reason
- Some problems with breaking degeneracy



Gradient Valley without time

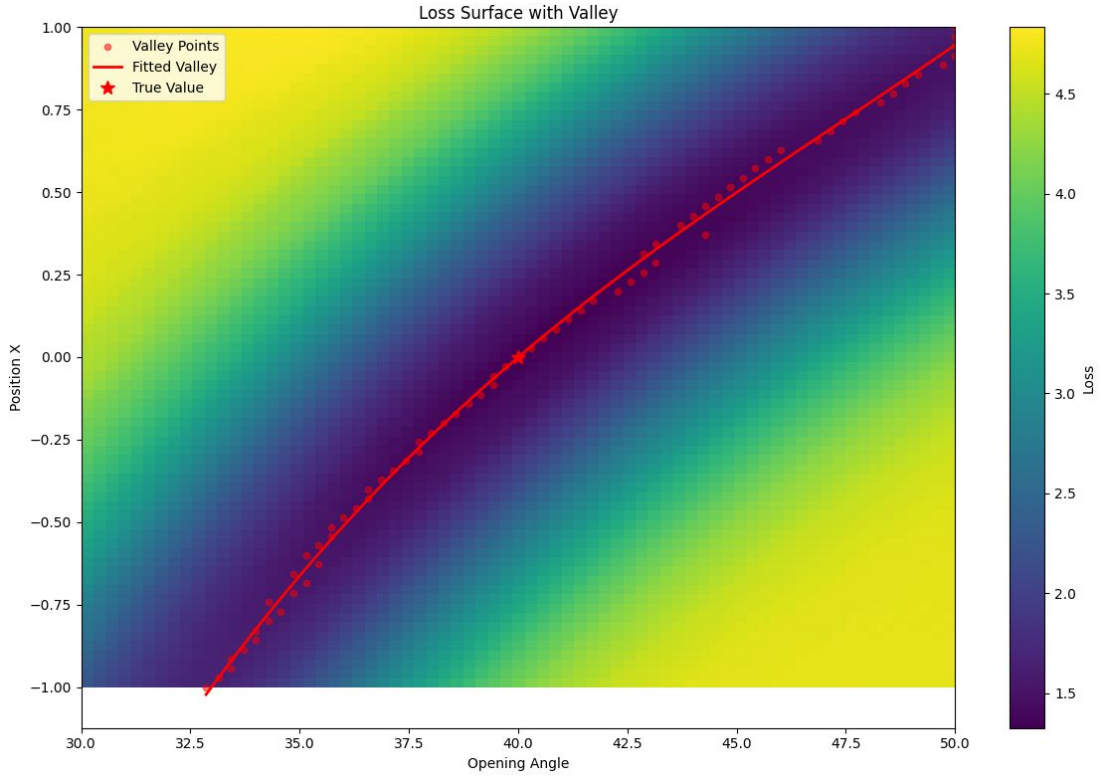
Time is used indirectly since we have the final position





Checking gradient values in the loss valley

I made a fit for the loss in this valley, and then saw what the gradient values are in this valley (along the direction of the valley)



This is that but changing sigma values. This sigma is the kernel size in the loss for comparing PMTs to PMTs