Signal Processing Refactoring/Translation

Jake Calcutt
08/21/2024

Introduction

The current OmnibusSigProc code is a monolithic implementation of the signal processing

- Encodes everything including 2D deconvolution, filtering for ROI finding, ROI finding itself
- Lots of opportunities for clarifications & optimizations

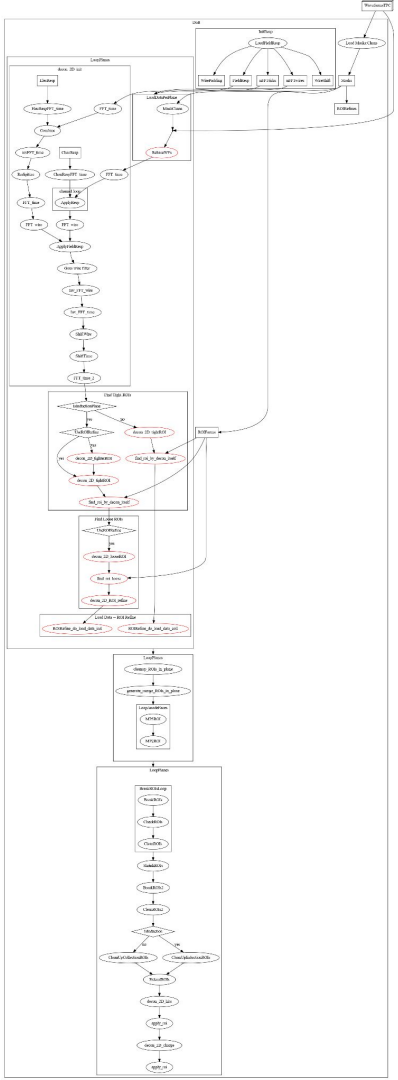
OmnibusSigProc Diagram (WIP)

Lots of nested routines, black box classes (ROI_form/ROI_refinement) that perform unexpected operations on the data

- Antithetical to larger wire-cell toolkit strategy of making use of graphs for flexibility
- Hard to extend

Suboptimal practices (recalculating the field/elec. response each frame/TPC)

- Could be done once and stored in memory



Ideas

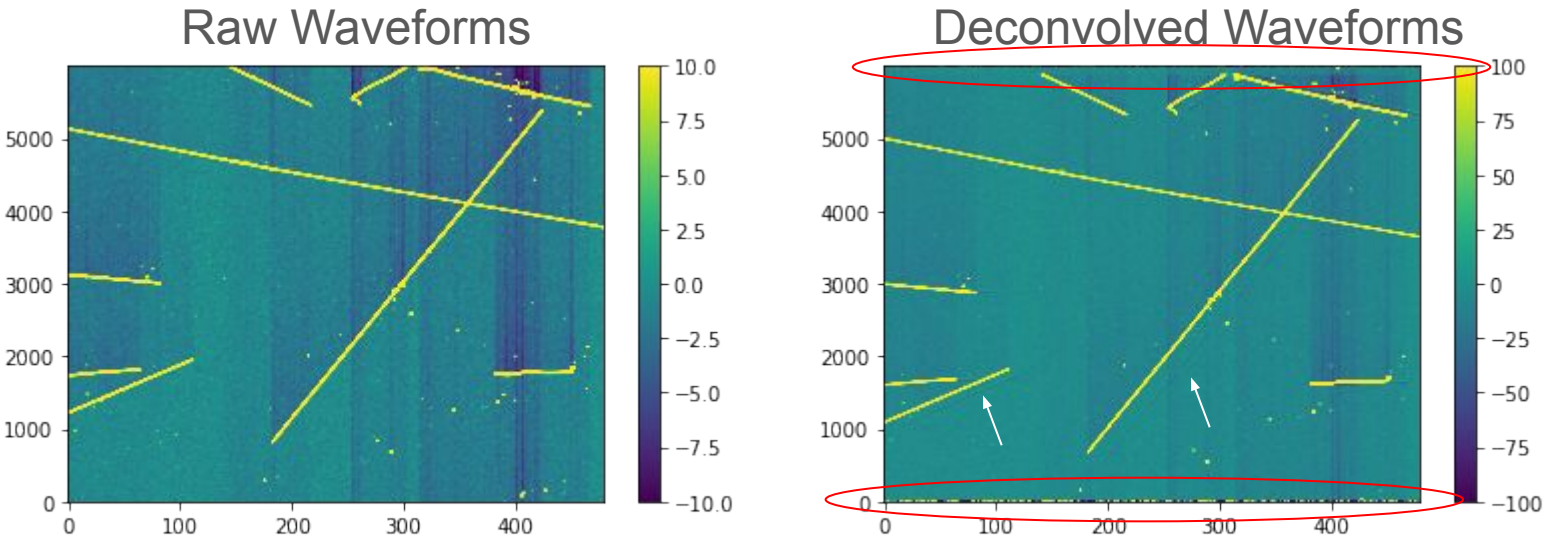
Brett and I have discussed options to address these issues

1. Refactor in C++ & replace Eigen arrays with some C++ tensor implementation (i.e. ArrayFire)
2. Implement in python and wrap in C++ to integrate into the toolkit
    a. I.e. implement in pytorch and link with libtorch
        i. Similar to what is done with DNN ROI finding (I think – please correct me if I'm wrong)
        ii. Get tensor operations for GPU utilization, but do we have to be careful about all the backprop overhead?
            1. Maybe just torch.no_grad() is enough

Pytorch implementation Attempt

See jupyter notebook for demonstration

Tested on PDSP simulation



Next Steps

Understand what is causing the strange artifacts to appear in the deconvolution

Extend functionality to ROI finding

Integrate into toolkit and test performance vs OmnibusSigProc

- On both CPU & GPU