Detector Protection
Ice, Salt, and Mesh

Barron Montgomery[1,2], Christopher O'Grady[2], Chuck Yoon[2], Mark Hunter[2+]

[1]Physics Department, Material Science and Engineering Department, Stanford University, 450 Serra Mall, Stanford, CA 94305, USA

[2]Linac Coherent Light Source, SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA 94025, USA.

[+]Contact: mhunter2@slac.stanford.edu

Introduction

When working with million dollar technology it's important to also invest in proper and secure protection.

More recently, our detectors have been undergoing damage from unforeseen formations of ice, salt, and mesh. All of these formations diffract the light from the beam to extreme intensities, damaging our detector.

The current means of protection for the detector is a simple algorithm which looks for pixels of an intensity higher than 5000 ADU. It has served well as a basic level of protection, but has not been able to account for the aforementioned threats. Here is our progress for developing a stronger means of Detector Protection.

Past Research

After studying many logged events with ice, salt, and mesh, I noticed that ice would remain visible on the CsPad for multiple events before disappearing. Based on this observation, I wrote a python script which searched for pixels with an intensity above a certain threshold (identifying the ice) and then analyzed subsequent events to see if those same pixels remained above the threshold. This would be good indication of ice formation.

The algorithm turns a 3-Dimensional image of an event into a binary grid: changing the intensity of each pixel to 1 or 0 depending on if that pixel was above the intensity threshold or not. It then multiplies repeated pixels in subsequent binary images to see if the pixel stayed above threshold for 2 (or more) events.

Although this algorithm proved it's worth, we soon learned that ice actually moves very subtly around the grid, thus rendering the repeated pixel hit algorithm unreliable to catch every ice event.

We have decided to combine multiple algorithms (possibly including the subsequence algorithm) in order to fully protect the detector.

-Existing above-threshold algorithm
-Geometry-based radial integration algorithm
-Subsequence algorithm
-Black-Hole algorithm

The current script I'm working on uses flood-fill techniques to identify 'Black Holes' as seen in figure 3. These black holes are portions of the grid where the intensities of the pixels are greater than what the detector is capable of outputting.

Data

Figure 1: Extreme case of ice rings forming on the DsCsPad

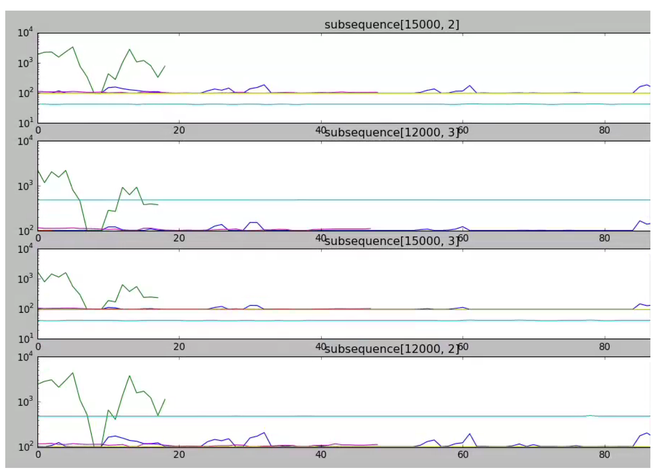


Figure 2: Data from subsequence algorithm: [threshold, n of events]

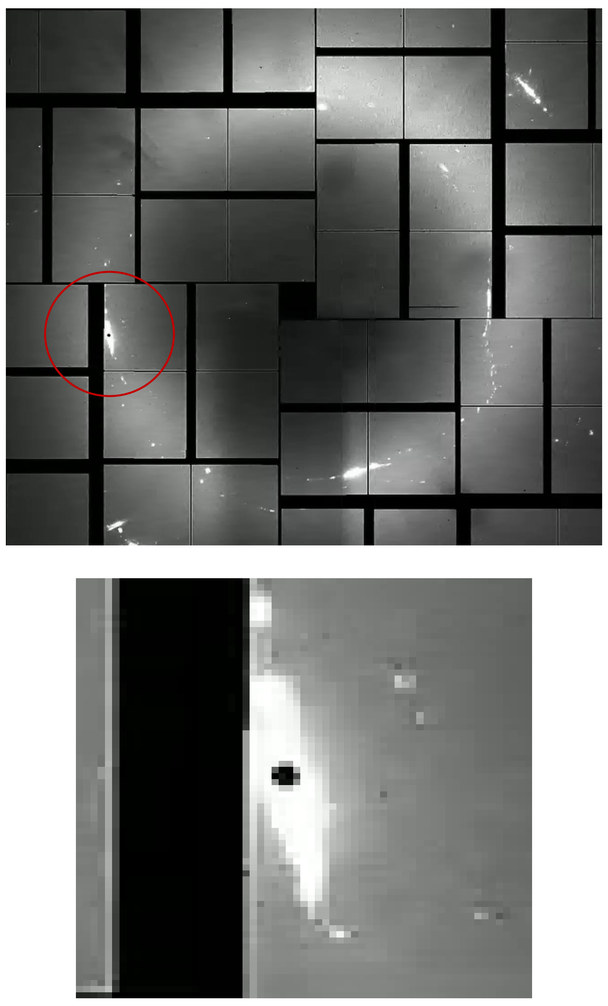
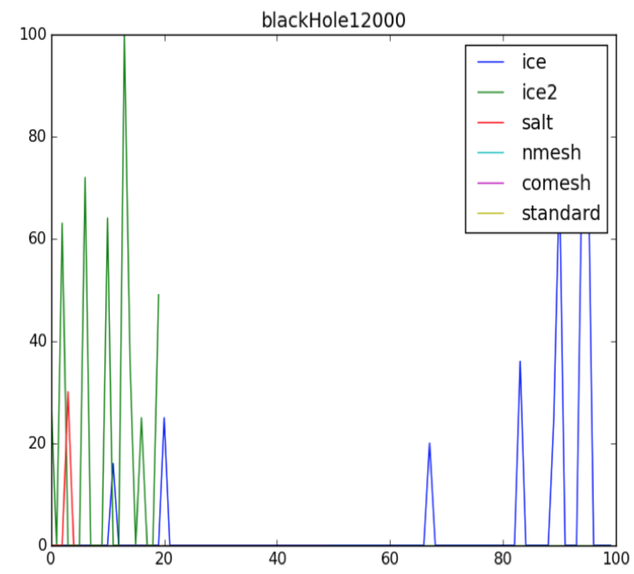


Figure 3: Ice rings on DsCsPad with a 'Black Hole'

Current Research

I'm currently working on a c++ script that searches for pixels above threshold and then uses a flood fill algorithm to find all other above threshold pixels connected to it. It then runs an edge finder algorithm to label the edges of the above threshold region and checks to see if there are edges within the region, which would indicate the finding of a 'Black Hole'

Give Stephen Hawking his Nobel Prize already.



Conclusions… so far

Progress:

•Black hole algorithm looks most promising so far
•Radial integration algorithm also looks promising, but geometry will be tricky and error-prone
•Soon: deploy black-hole algorithm in AMI plug-in (optimize performance after we get some experience) to get experience

Weaknesses:

•Only works when the DAQ is running
•If one shot can kill the detector we are quite powerless
•If software is not fast enough we will miss events
•AMI plugin doesn't see all events. Could move algorithm to DAQ readout node in future and get all events.

In future (LCLS-II) we should try to develop hardware (firmware?) protection.

Acknowledgments

Use of the Linac Coherent Light Source (LCLS), SLAC National Accelerator Laboratory, is supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences under Contract No. DE-AC02-76SF00515.

Date: 09/06/2016