Beam Background Merging

Sub title

University of New Hampshire

Maurik Holtrop

Merging data with MC

- This presentation is entirely based on Kyle's work, presented at the software meeting: Feb 6, 2018

- We have discussed merging beam background data with MC signal for a while as a method of speeding up the MC generation.

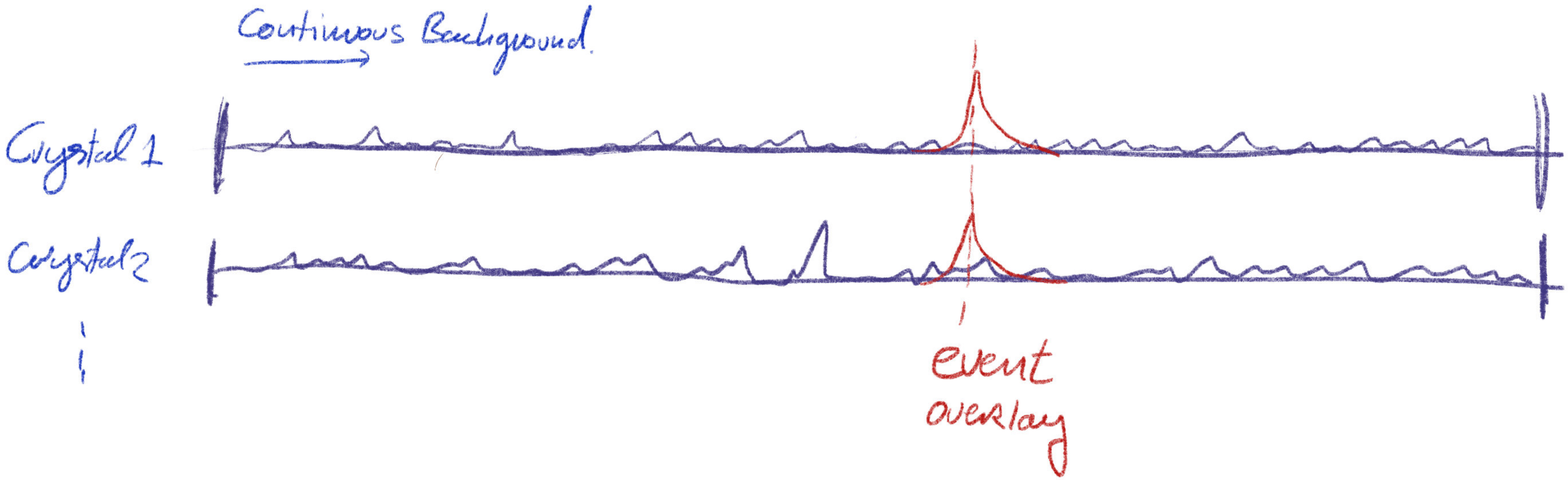
- Currently we simulate the beam background. This is time consuming, and it is easy to miss some physical process or not simulate it well enough.

What we want

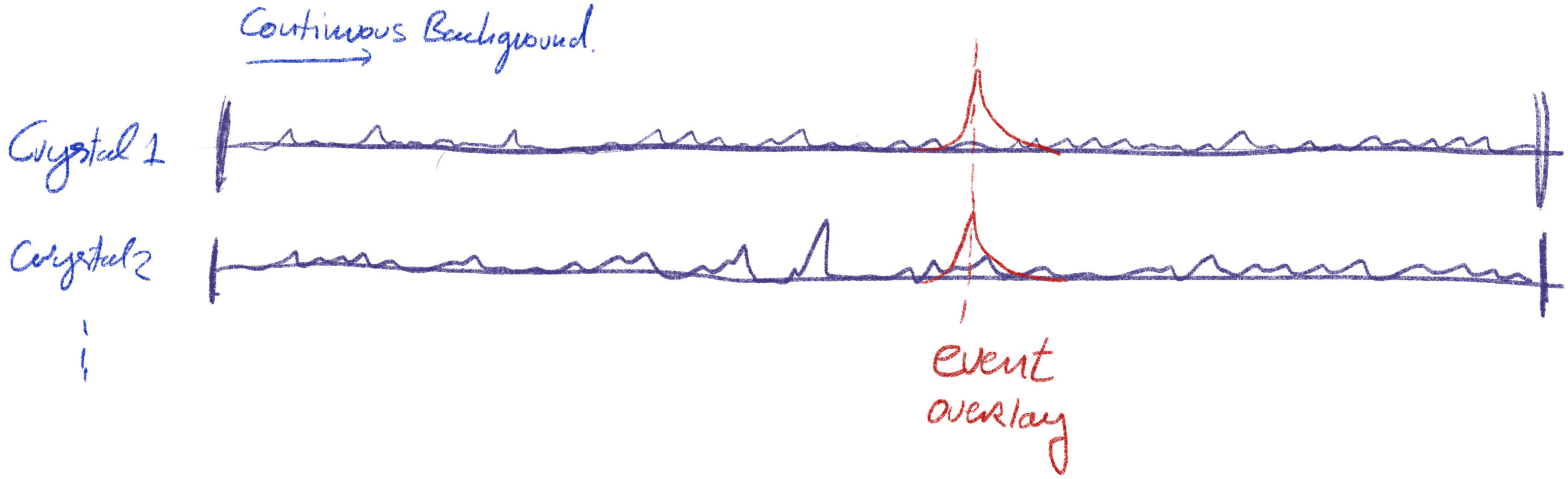
- A Monte Carlo event contains a "signal" (trident, or simulated A') and beam background (all the other signals in the detector)
- We generate out signal using the MadGraph event generator, then simulate the detector response using SLIC (Geant4).
- Instead of simulating a beam background with EGS4/5+GEANT4+SLIC, we want to take data from the detector.
- In theory, this seems really simple, use data without trigger or any bias and add the MC signal….
- The difficulty is in the details.

Ideally

- For simplicity, consider the ECAL only.
- In the ideal situation, we would have continuous readout from the detector, so we can take a time period, ΔT, of several trigger windows, and we just overlay the pulses from MC. Properly summing the signals.
- We then search this data for a trigger with the trigger simulation, which allow for “trigger stealing” and other effects to be simulated.

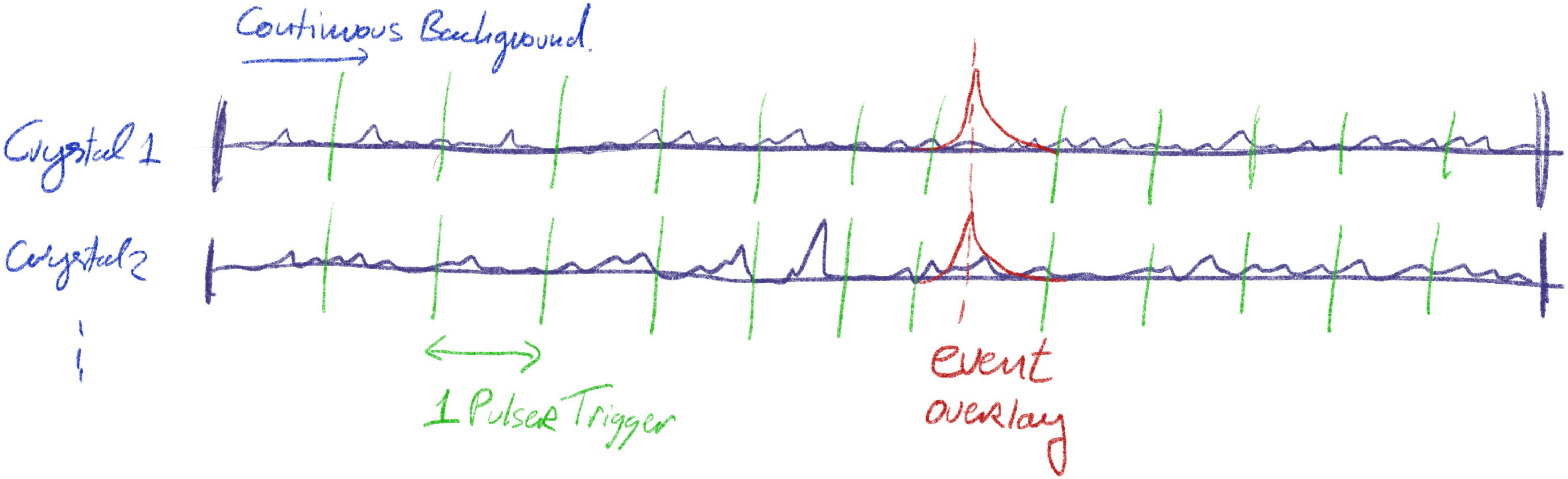


Ideally



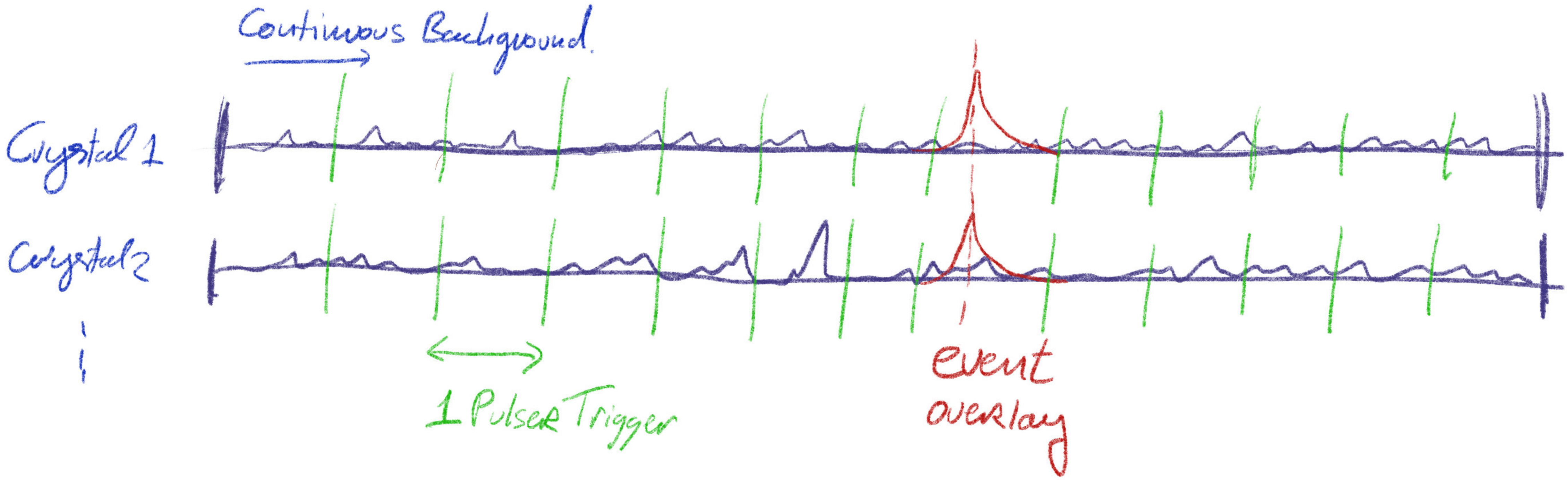
- Problem: We do not have streaming readout.
- Our data comes in 96 ECAL FADC samples: 384 ns chunks that are discontinuous in time.

Method-1



- Can we just take the pulser data, which represents unbiassed triggers, and stitch the events back to back to get longer samples?

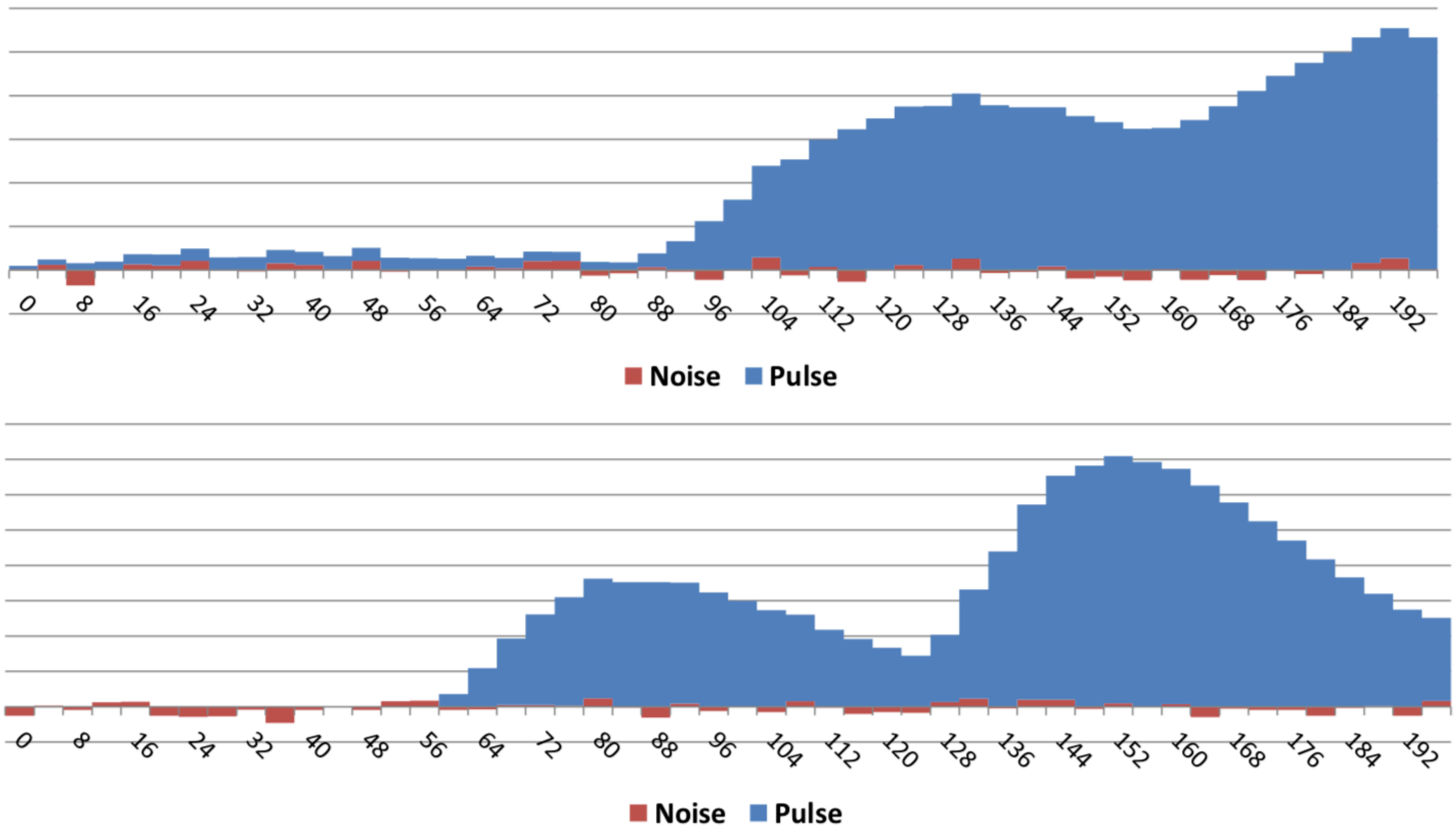
Method-1



- Can we just take the pulser data, which represents unbiassed triggers, and stitch the events back to back to get longer samples?
- The problem is with he boundaries where you do the "stitching", you would get discontinuities.
  - Maybe possible to do this anyway. Just check the edges.
  - Worth a try!?
  - Is the trigger simulation capable to run over long stretches?

Method-1

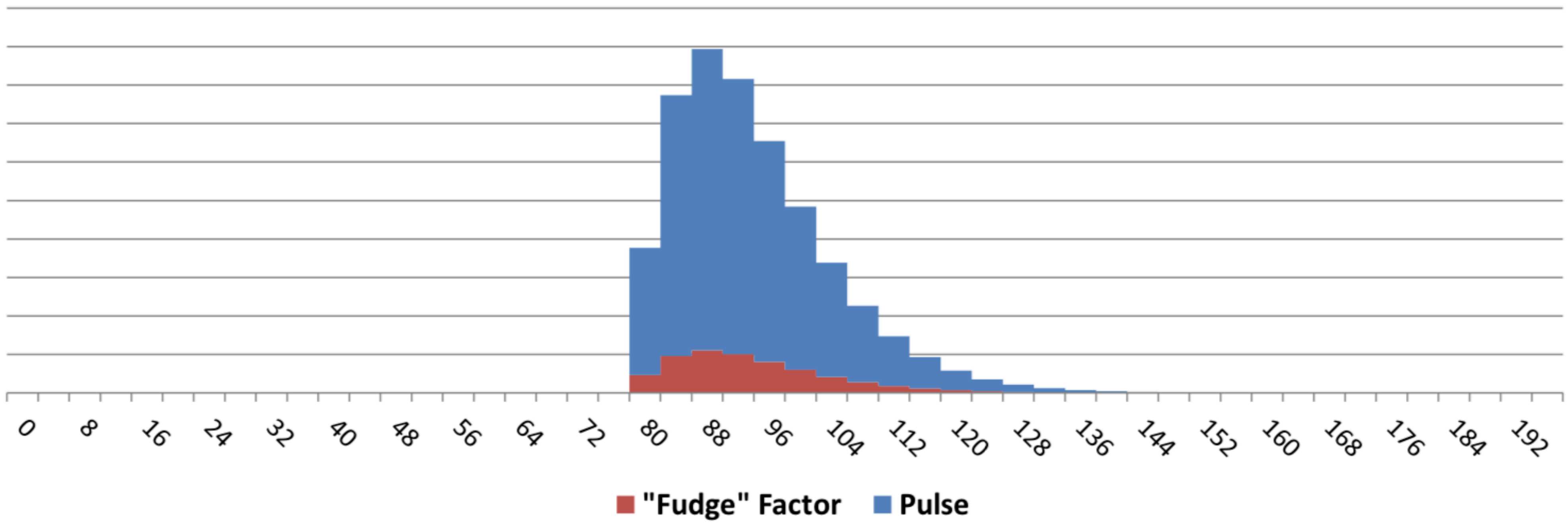
- Boundary issue for pulser events:
  - Consider two consecutive events.



- We need some more data analysis to see how bad the problem really is.
- It seems possible to extrapolate the pulse in cases where this happens.

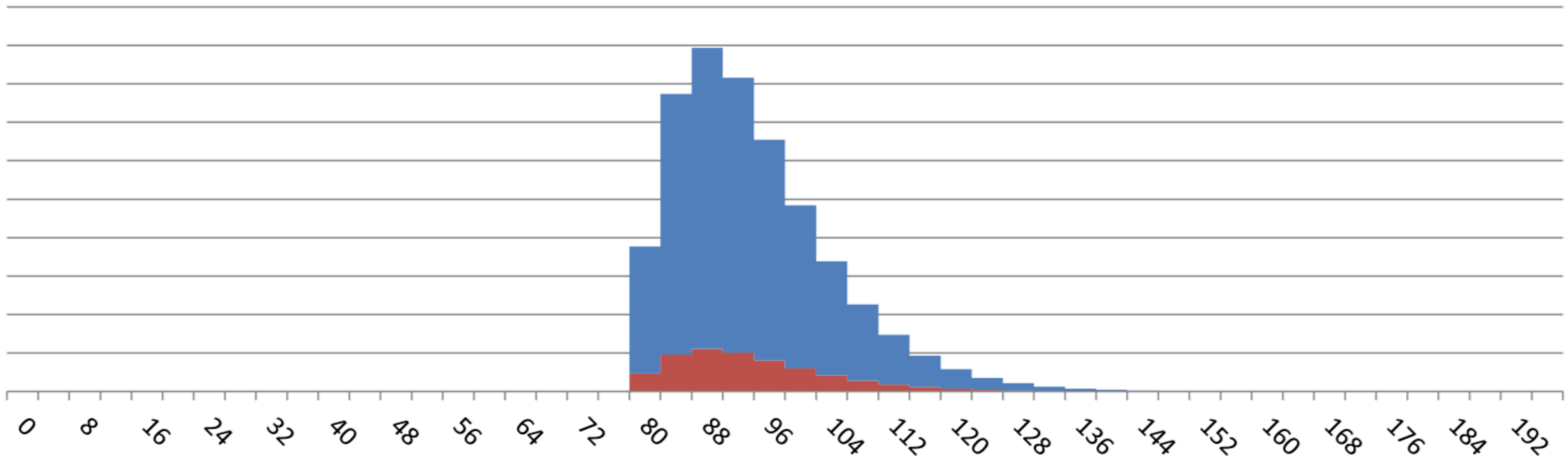
Method-2

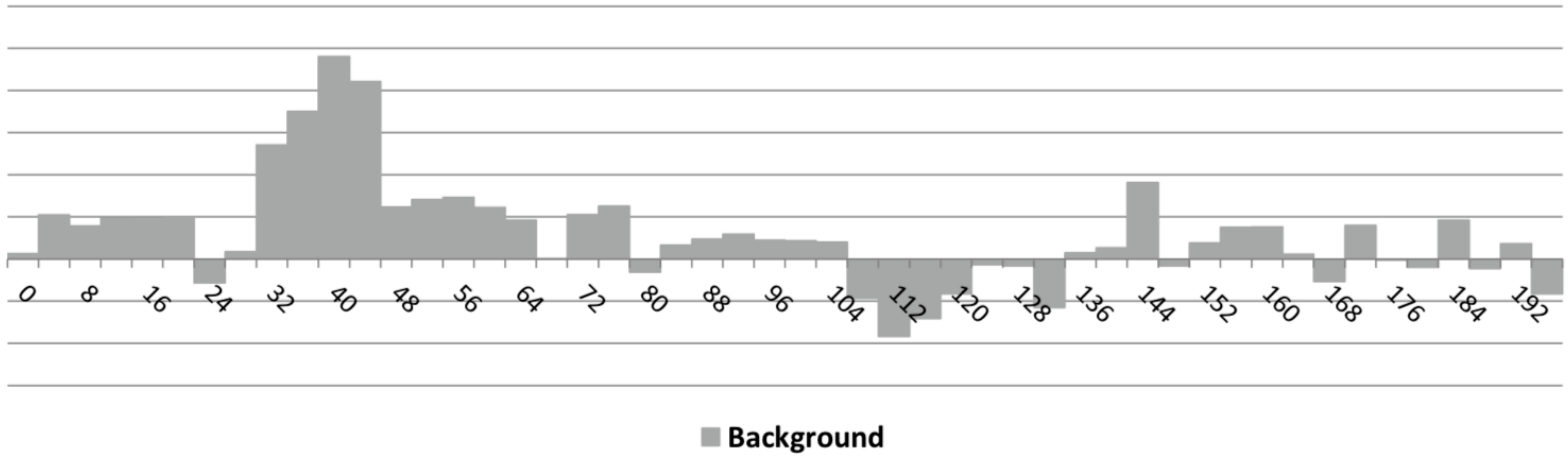
- If we give up on the trigger stealing effect, perhaps taking that into account in a different way, we can only consider the 96 ECal samples of one event.
- Simulate the MC event and process with GEANT4 (SLIC).
- One crystal:



Method-2

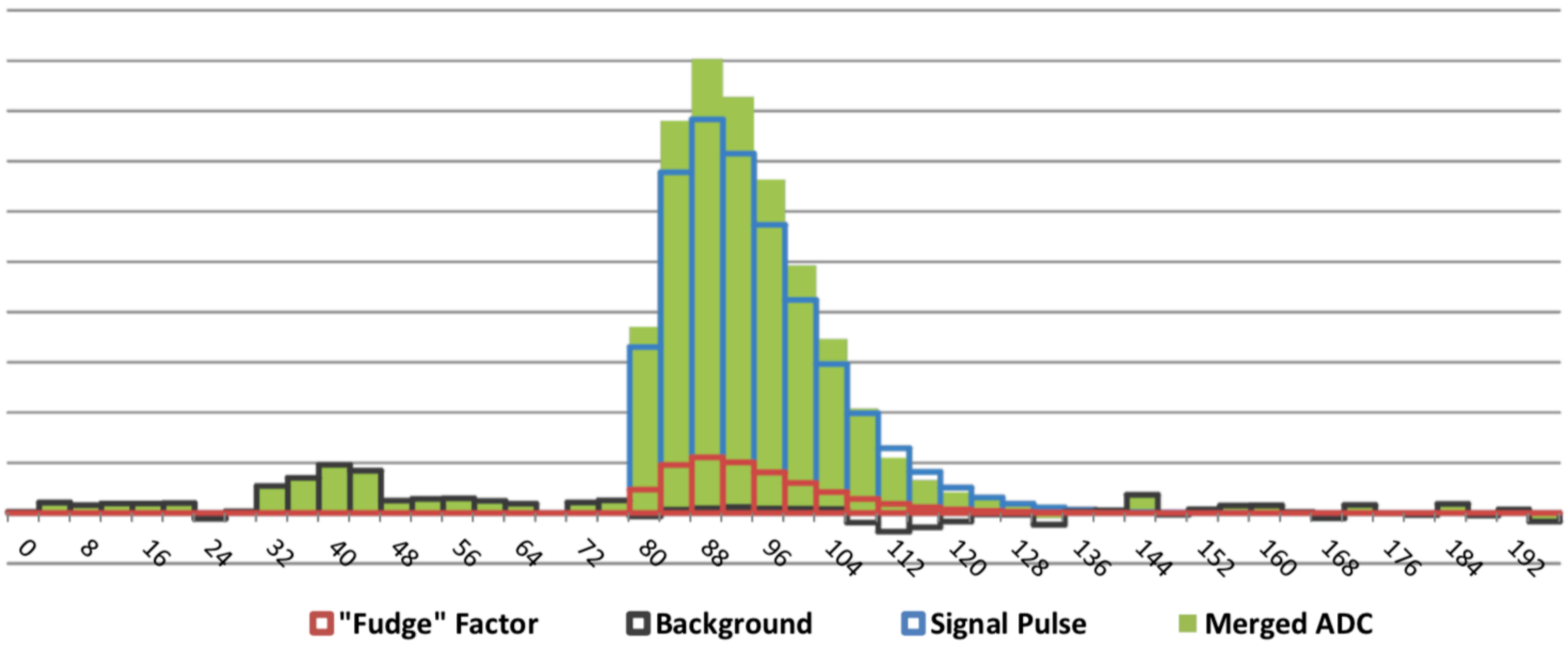
- Simulate the MC event and process with GEANT4 (SLIC).
- Take a random pulser event and look at the same crystal channel.





Method-2

- Simulate the MC event and process with GEANT4 (SLIC).
- Take a random pulser event and look at the same crystal channel.
- Overlay the pulser data on the signal.
  - No problems with edge effects.
  - Noise is entirely from the pulser data.
  - Need to be careful with pedestal subtraction etc.



Method-2

- This method does not fully take into account any trigger stealing effect of hits that may occur before this one window.
  - Probably not a large effect.
  - Could be other ways to account for it.
  - Not sure it this is fully accounted for in current simulation either.
- Method-2 (probably) does not require a complete rework of the trigger readout code.
  - But it will need a revisit of this code.
- Either method needs to make sure the background events are chosen correctly:
  - Match the luminosity and running conditions of what you want to simulate.
  - Make sure no beam trip, SVT good, etc.
  - Can randomize the pulser sample, if needed.
- Noise!
  - We need to re-write how noise is handled in the simulation. Currently we only add "noise" to the pulse size, this not quite correct.

Conclusion

- It is high time we get beam background merging implemented.
  - We cannot afford the CPU time not to.
  - Hopefully the result will be more accurate.
- We should just get started, and see how well it works, then incrementally improve the method.