Wire Cell Event Reconstruction Software for LArTPC Detectors

Brett Viren
for the BNL Wire Cell Group

Physics Department

BROOKHAVEN NATIONAL LABORATORY

BNL CSC Seminar 2015 Sep 29

Outline

Who, What, How, Why

who The **Wire Cell** development is centered in the **Electronic Detector Group** of the **BNL Physics Department**.

what Experimental study of the **properties of neutrinos**,

how using **Liquid Argon Time Projection Chambers** (LArTPC) and **accelerator-neutrino beams**

why in order to reach these **Physics objectives:**

- Discover and measure **CP symmetry violation** in neutrino interactions.
- Determine **neutrino mass ordering**.
- Precisely measure **neutrino oscillation parameters**.
- Potentially observe **neutrino bursts from supernova**.
- Search for **proton decay** and **sterile neutrinos**.

Why Liquid Argon TPC

These goals require neutrino detectors to be:

- **big** active mass of 10 – 100 kt's (mitigate tiny neutrino cross sections).
- **efficient** to capture neutrinos and separate signal from background.
- **accurate** and precise to measure energy, neutrino flavor and event type.
- **quiet** low background environment shielded against cosmic-μ and natural radioactivity, $\Rightarrow$ deep underground.

Two main, excellent technologies:

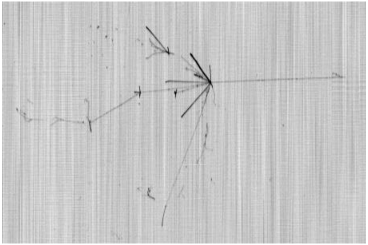
- **Water Cherenkov** big++, **efficient+**, accurate++, quiet++
  simple, mature technology, modest resolving power.
- **LArTPC** **big+**, eficient++, accurate++, quiet++,
  high-resolution, R&D still improving, excellent future potential.

Many challenges still to overcome, but **LArTPC**'s promises make it the **chosen technology for neutrino experiments in the US**.

LArTPC Experiments - ICARUS

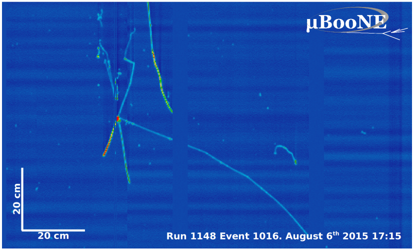
The origin of LArTPC technology for Neutrinos: C. Rubbia, 1977 led to **ICARUS, the first, large-scale LArTPC**.

- $2\times$ 300 t modules.
- Took data at Gran Sasso tunnel, Italy from CERN neutrino beam.
- Moving to Fermilab as part of the **Short-Baseline Neutrino** Program.



LArTPC Experiments - MicroBooNE

Just started taking data at Fermilab!

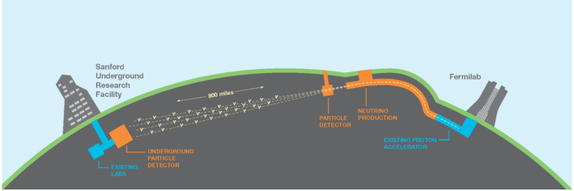


- 85 t fiducial mass.
- 8256 channels
- 3 mm wire pitch.
- Investigate *low energy excess puzzle*, look for sterile-ν, measure ν-Ar cross sections.

MicroBooNE is the initial test bed for Wire Cell reconstruction.

LArTPC Experiments - DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT

"International **mega-science** project"



Three stages of LArTPC detectors:

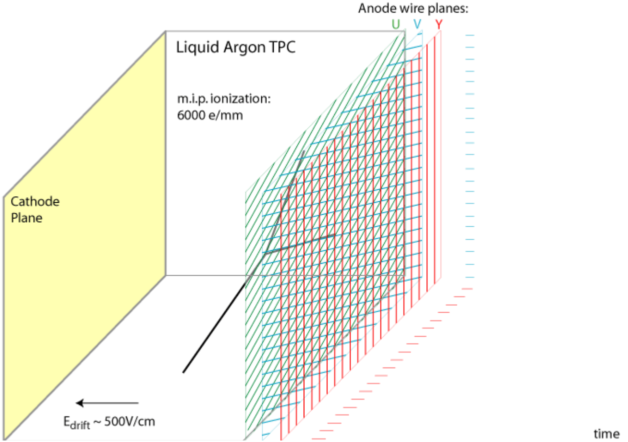
1. "**35ton**" prototype (at FNAL) commissioning now, running early 2016.
2. Full-scale "**protoDUNE**" (at CERN) 2017 with π, K, p beam tests.
3. Full "**DUNE**" far detector underground in South Dakota ~2025.
    - **40 kt fiducial mass** in 4 modules.
    - 5 mm wire pitch, **1.5M channels**.

DUNE design "features" driven by **detector size** and **underground location** ⇒ some extra reconstruction challenges (two sided anode planes + wrapped readout wires = extra ambiguities).

Next: primer on how LArTPC works.

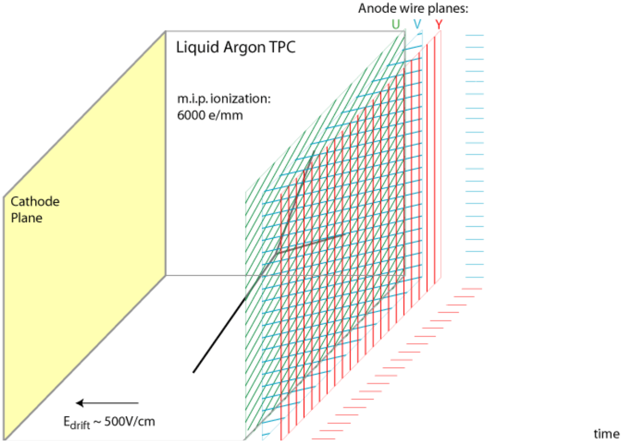
Illustration by Bo Yu (BNL)

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



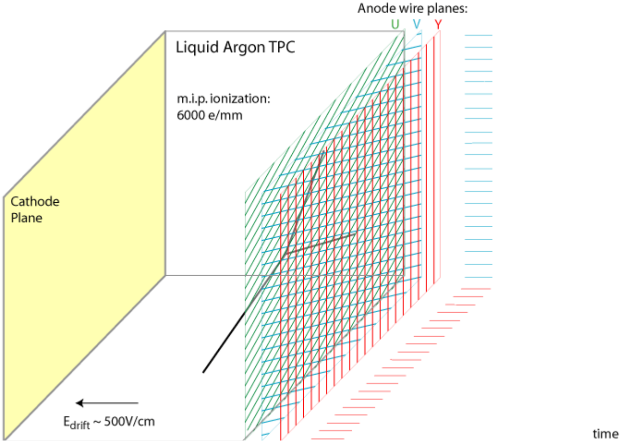
Charged particles traverse the detector at relativistic speeds.
This leaves behind **ionized argon** and **electrons** in their tracks.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



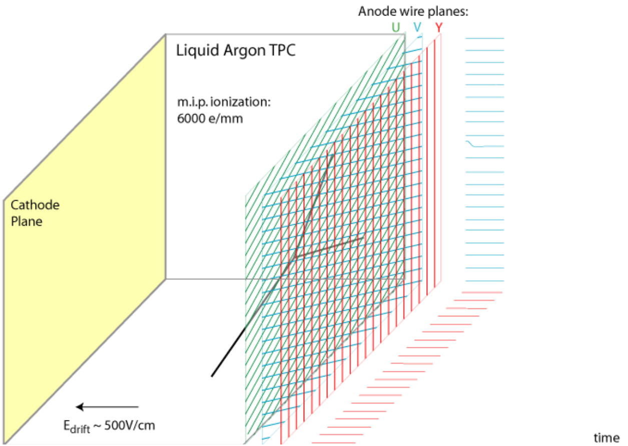
Ion and electron pairs drift off in opposite directions in the electric field applied between **anode wires** and the cathode plane.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



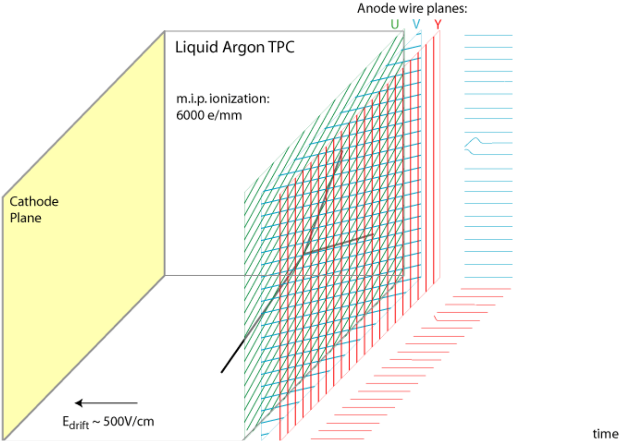
Here, we watch the electrons drift. They drift with a speed of 1.6 mm/μs (for nominal E-field of 500 V/cm).

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



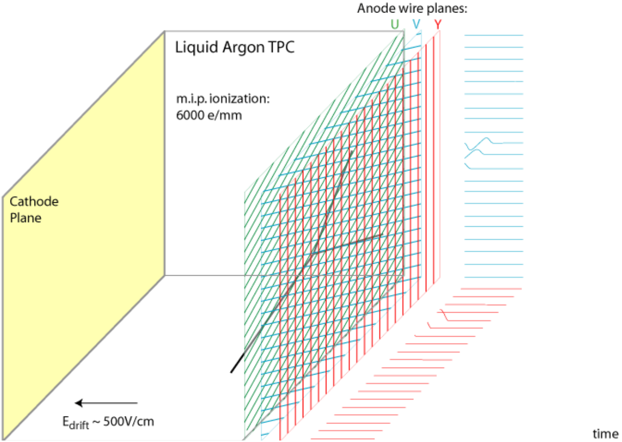
As the electrons approach the wires they induce signals. Wires in each of **three planes provide independent measure** of the **drifting charge**.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



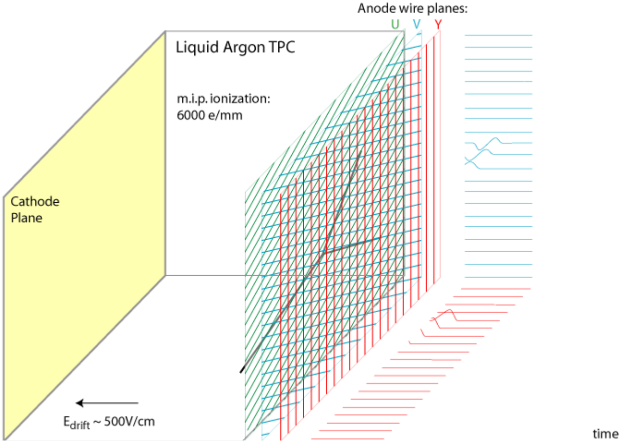
There are **two** planes of **induction wires** (waveform from "V" plane in blue). Wires see a **bipolar signal** as the charge first drifts toward and then away.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



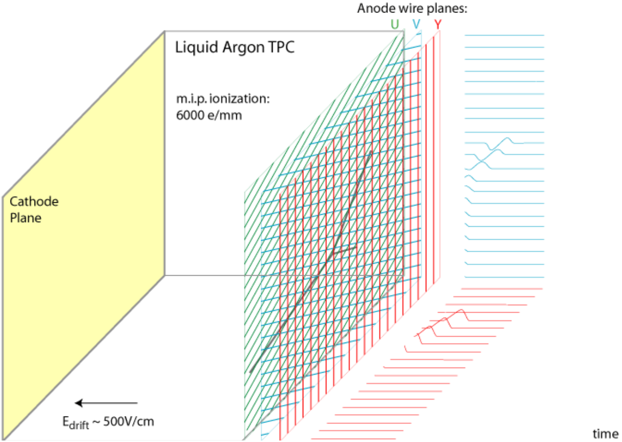
The final, **third** plane **collects the drifting charge** and thus sees a **unipolar signal** (in red waveform).

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



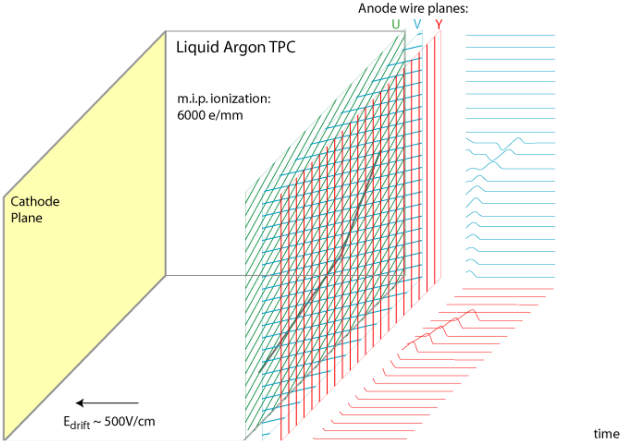
The exact **signal response** is complex and depends on details of the local electric field and readout electronics.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



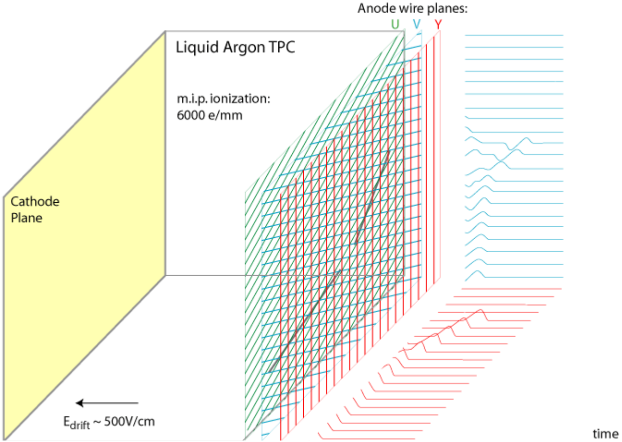
Charge waveform from each wire is digitized as a function of time. Typical digitization "tick" is 0.5 µs, wire spacing is 3 to 5 mm.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



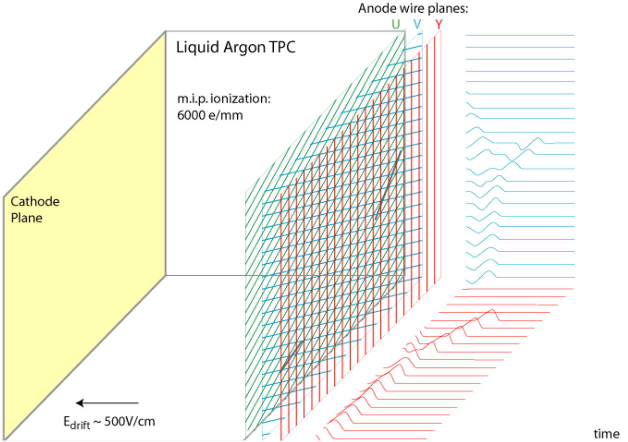
Digitized waveforms are **deconvolved** with a **response function** and a **noise filter** to infer the amount of drifted charge near the wires at any given time.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



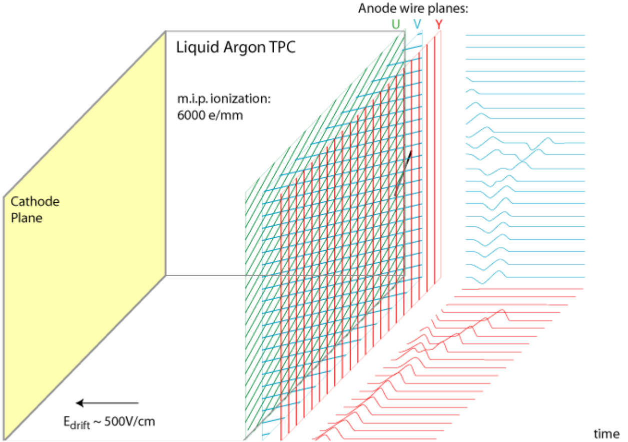
As a batch of electrons drift, it **diffuses** in both the **longitudinal** and **transverse** directions. LAr impurities also **attenuate** the charge.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



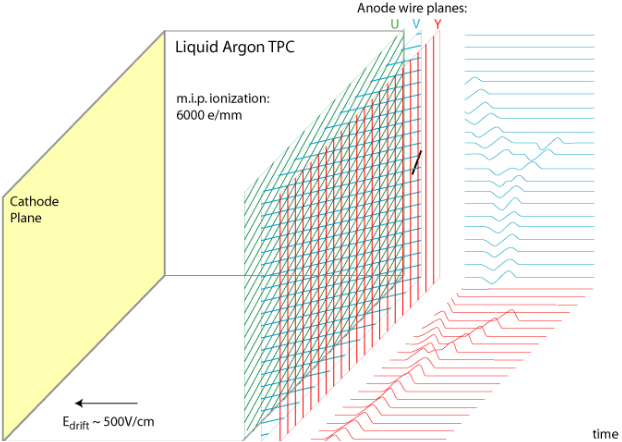
At the maximum drift distance allowed by the detector, diffusion can be as much as 1 μs longitudinally and 2.5 mm transverse.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



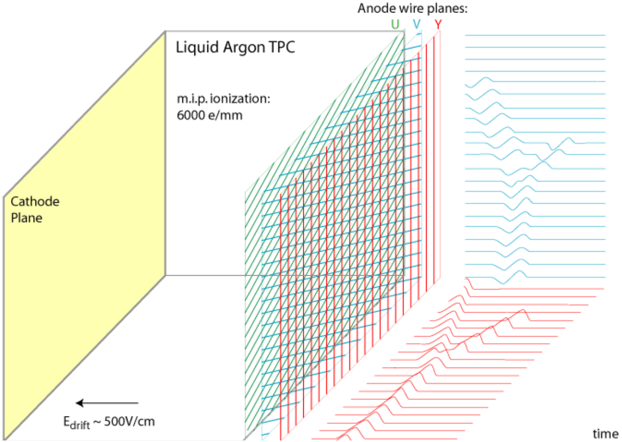
Still drifting...

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



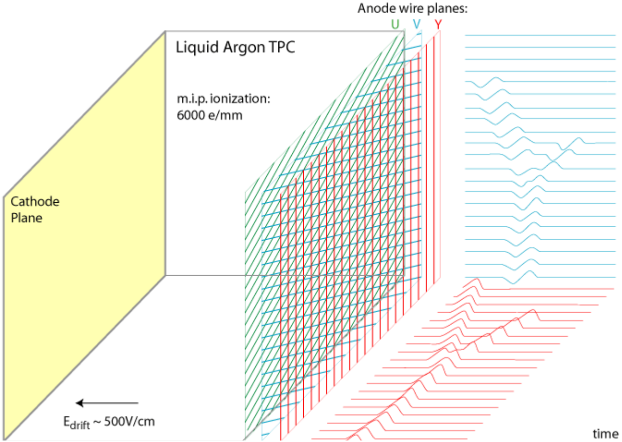
Still drifting... It's a slow detector!
1.6 mm/μs drift speed × ∼meter drifts ⇒ few ms to readout.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



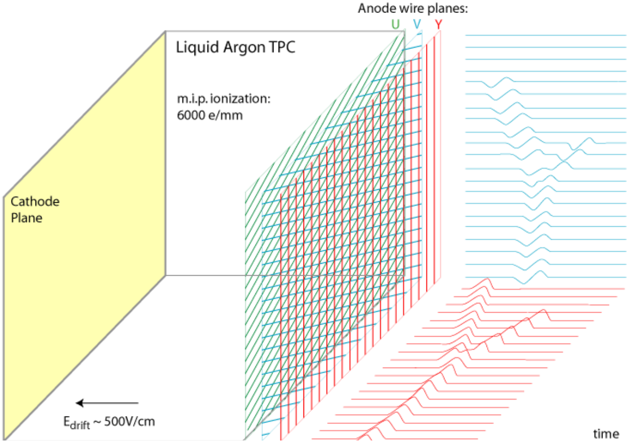
Still drifting...

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



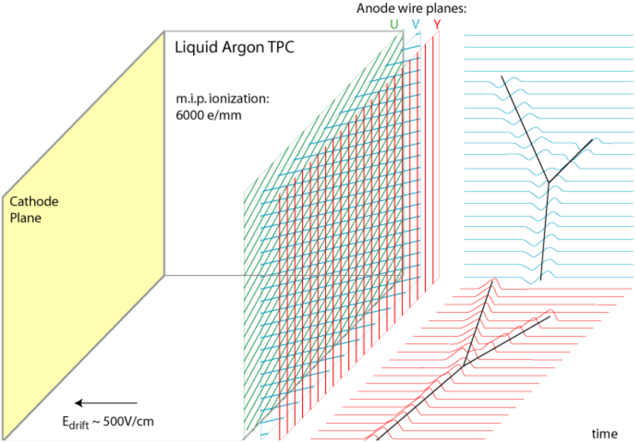
Finally, each wire plane reads out one 2D view (time vs. wire-pitch direction) of the original pattern of ionizing particles. **The drifting charge has three independent measures.**

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber



Combining three 2D views can give a tomographic reconstruction of the location and amount of energy deposition of the original particle trajectories.

LArTPC = **L**iquid **Ar**gon **T**ime **P**roportional **C**hamber

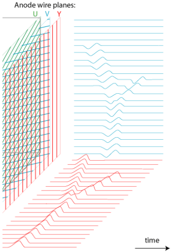


Traditional methods work in 2D first, then combine views by assuming/asserting a particular event topology.
Wire Cell goes directly to 3D imaging with no such assumptions.

Some Data Numerology

LArTPC can produce **prodigious quantities** of **high-resolution** data from **huge detector volumes**:

- 10^4 – 10^6 channels
- 2MHz @ 12 bit FADC digitization
- each "event" spans several milliseconds



0.5 μs waveform digitization.

Two general DAQ readout strategies:

Full Stream reads out entire waveform (**MicroBooNE**)

- **30GB/s in 120 MB "events"**.
- DUNE would produce 5 TB/s in 25 GB "events"!

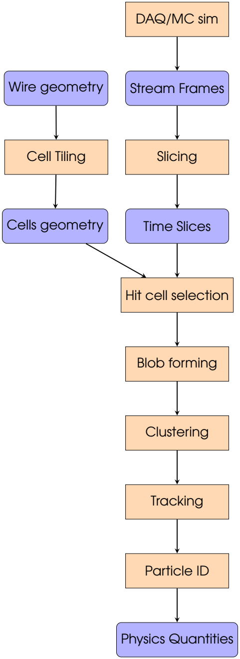
Zero Supression drops waveform chunks below a threshold (**DUNE**)

- Threshold based on noise (E_{equiv} <0.5 MeV/wire)
- 2.5 MB/event $\rightarrow$ **100's TB/year**
- caveat: must veto ^{39}Ar decay in DAQ or accept **50 PB/year**

Wire Cell Reconstruction Method

Four main parts:

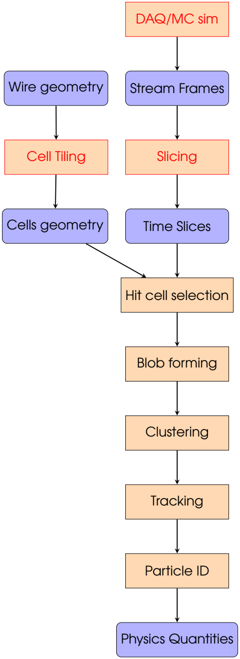
1. Data Preparation
   - Construct wire geometry and **associated cells**.
   - Read framed data stream, form **time slices**
2. Imaging of Activity
   - The heart of the Wire Cell technique
   - Identify the likely "hit" **cells** in the time slice.
3. Pattern Recognition
   - Application of Wire Cell imaging
   - **Cluster** activity in space and across time slices.
   - **Categorize**: track, shower, etc.
4. Physics Quantities
   - Determine particle ID and kinematics of tracks/showers.



Wire Cell Reconstruction Method

Four main parts:

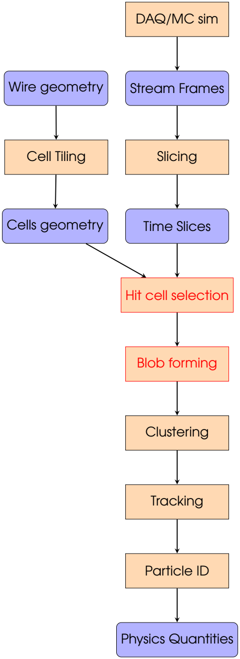
1. Data Preparation
   - Construct wire geometry and **associated cells**.
   - Read framed data stream, form **time slices**
2. Imaging of Activity
   - The heart of the Wire Cell technique
   - Identify the likely "hit" **cells** in the time slice.
3. Pattern Recognition
   - Application of Wire Cell imaging
   - **Cluster** activity in space and across time slices.
   - **Categorize**: track, shower, etc.
4. Physics Quantities
   - Determine particle ID and kinematics of tracks/showers.



Wire Cell Reconstruction Method

Four main parts:

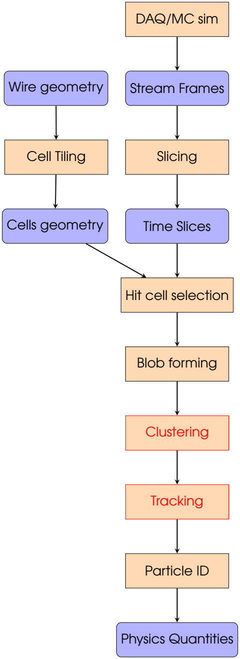
1. Data Preparation
   - Construct wire geometry and **associated cells**.
   - Read framed data stream, form **time slices**
2. Imaging of Activity
   - The heart of the Wire Cell technique
   - Identify the likely "hit" **cells** in the time slice.
3. Pattern Recognition
   - Application of Wire Cell imaging
   - **Cluster** activity in space and across time slices.
   - **Categorize**: track, shower, etc.
4. Physics Quantities
   - Determine particle ID and kinematics of tracks/showers.



Wire Cell Reconstruction Method

Four main parts:

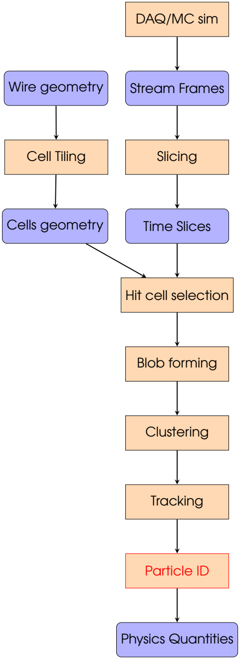
1. Data Preparation
   - Construct wire geometry and **associated cells**.
   - Read framed data stream, form **time slices**
2. Imaging of Activity
   - The heart of the Wire Cell technique
   - Identify the likely "hit" **cells** in the time slice.
3. Pattern Recognition
   - Application of Wire Cell imaging
   - **Cluster** activity in space and across time slices.
   - **Categorize**: track, shower, etc.
4. Physics Quantities
   - Determine particle ID and kinematics of tracks/showers.



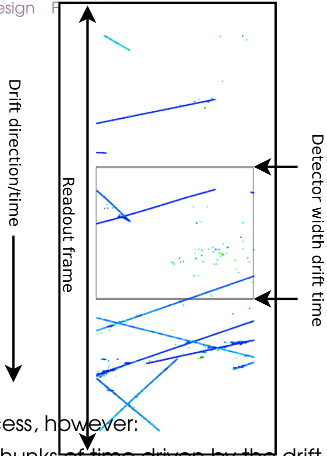
Wire Cell Reconstruction Method

Four main parts:

1. Data Preparation
   - Construct wire geometry and **associated cells**.
   - Read framed data stream, form **time slices**
2. Imaging of Activity
   - The heart of the Wire Cell technique
   - Identify the likely "hit" **cells** in the time slice.
3. Pattern Recognition
   - Application of Wire Cell imaging
   - **Cluster** activity in space and across time slices.
   - **Categorize**: track, shower, etc.
4. Physics Quantities
   - Determine particle ID and kinematics of tracks/showers.



Framing

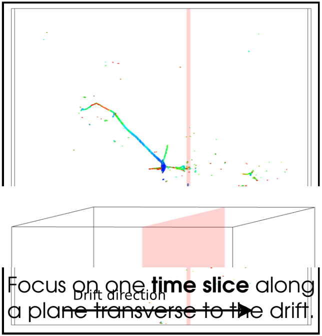


Wire Cell is a streaming process, however:

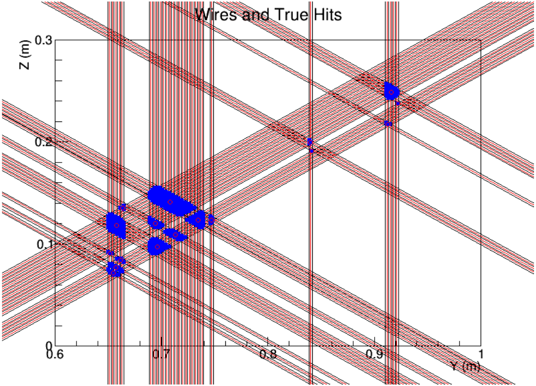
- DAQ writes out large chunks of time driven by the drift time across the detector,
- Seamlessly capture any early/late activity encroaching around the trigger time.
- Triggerless running, to capture any low threshold early/late activity.

To Wire Cell, frames are “just an artifact” of the DAQ, wire cell wants time slices →

Time Slicing



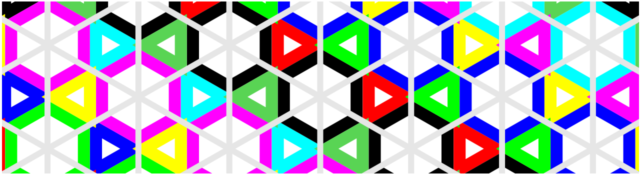
Focus on one **time slice** along a plane transverse to the drift.



- Slice duration is chosen to **match electronics shaping time**: 4 FADC "ticks" = 2 μs.
- Shows "true" (simulated) hits red circles.
- Select wires hit in the slice.
  → hit = FADC charge-in-slice above a threshold.
- Potential "cells" holding charge.

Tiling

Zoom in on the wires and their associated (constructed) cells.



MicroBooNE geometry, grey wires, colored cells.

Cell construction:

- Define small, 2D regions near *approximate triple crossings* of one wire from each of the three planes.
- A "cell" is the association of this region with the three wires.
- Cells completely tile the plane (time slice): no gaps, no overlaps.

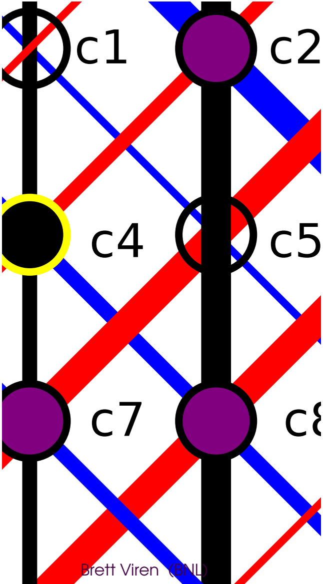
Awkward consequences of current construction:

- Two cell types: *wire-centered* and *gap-centered* cells.
- In general (eg, DUNE), cell shapes and sizes not uniform nor regular, depend on wire plane pitches, angles and offsets.

The heart of the Wire Cell concept: if all three triple-crossing **wires** are "hit", the associated **cell** likely contains drifted charge.

Cell Ambiguity - Example Hit Pattern

Zoom in on $5 \times 5 \times 3$ wires:



Spatial multiplexing $\Rightarrow$ **ambiguity**:

Good	wire v3 measures no charge, $\therefore$ all its cells must not be hit.
Bad	hits **c2**, **c7** and **c8** induce "ghost" at **c4**.
Ambiguous	multiple cells measured by same wire. How much charge is in **c6**???

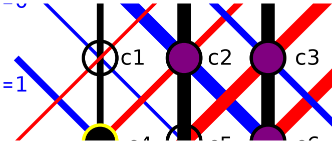
In some cases ambiguity can not be resolved at the cell level!

Solution Attempt 1 - cell-level

The charge expected on the **wires** ($\vec{w}$) can be calculated given (the unknown) charge in the **cells** ($\vec{c}$):

$$\vec{w} = \mathbf{G_{wc}}\vec{c}$$

$\mathbf{G_{wc}}$ is the **wire-cell adjacency matrix**, purely geometrical and perfectly known, function of detector design.

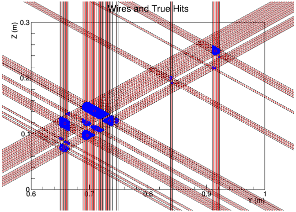


$$\Leftrightarrow \begin{pmatrix} 0.0 \\ 1.0 \\ 2.0 \\ 2.0 \\ 0.0 \\ 1.0 \\ 1.0 \\ 0.0 \\ 2.0 \\ 1.0 \\ 1.0 \\ 2.0 \\ 2.0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.0 \\ 1.0 \\ 1.0 \\ 0.0 \\ 0.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 0.0 \end{pmatrix}$$

Wish to solve inverse: $\vec{c} = \mathbf{G_{wc}^{-1}}\vec{w}$. However, $N_{cells} \approx N_{wires}^2$
$\Rightarrow$ as N_{cells} grows, more unknowns ($\vec{c}$) than knowns ($\vec{w}$)!

Solution Attempt 2 - blob-level

Goal: **reduce matrix size** and **remove ambiguity** by making "blobs" of cells.



1. Select set of all cells with all three wires "hit".
2. Partition set into spatially contiguous subsets: "**blobs**".

Equation to solve is like the previous:

$$\vec{w}_b = \mathbf{G_{wb}} \vec{b}$$

$\vec{c} \rightarrow \vec{b}$ vector of charge in each blob.

$\mathbf{G_{wc}} \rightarrow \mathbf{G_{wb}}$ the wire-blob adjacency matrix for the slice.

$\vec{w} \rightarrow \vec{w}_b$ vector of charge on all wires associated with a blob.

But now **more favorable numerology**: $N_{blob} \lesssim N_{w_b}$

Another wrinkle: charge uncertainty

Measure of the drifting charge by a wire has uncertainty:

- Environmental, electronic and thermal noise.
  - $\rightarrow$ Can be correlated across wires/channels/chips/boards/etc.
- Statistical uncertainty due to digitization.
- Systematic uncertainties from detector response deconvolution.

Compare measured wire charge ($\vec{w}_{meas}$) with expected ($\vec{w}_{exp}$) and form a χ^2 with a covariance uncertainty matrix V.

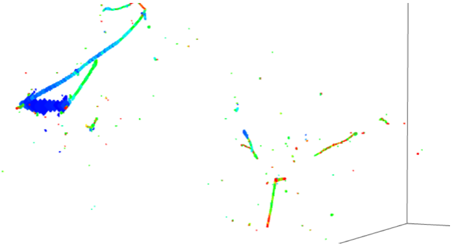
$$\chi^2 = (\vec{w}_{meas} - \vec{w}_{exp})^{\intercal}\mathrm{V}^{-1}(\vec{w}_{meas} - \vec{w}_{exp})$$

"It can be shown" that minimizing this χ^2 is equivalent to inverting $\mathbf{G_{wc}}$ (or $\mathbf{G_{wb}}$) matrix equation.

$\rightarrow$ **This is a very CPU intensive, but critical step!**
(and, btw, we are still in the raw-data processing stage!)

The Payoff: imaged 3 GeV ν_e interaction

True energy depositions.



Wire Cell Imaging.

- **Excellent imaging** of major track features as well as isolated activity.
  → a static 2D view doesn't do it justice! See it for yourself.
- **Residual ambiguity** seen as wide blue patches.
  → **Inherent in LArTPC technology**, Wire Cell just makes it evident.
    - due to activity running parallel to the wire plane in one time slice.

  → Battling this will likely require (even more) **CPU-intensive algorithms** such as **iterative reconstruction**.

Pattern Recognition

Our current **post-imaging** approach:

1. **cluster** together blobs contiguous in space and time (slice).
2. **track** a line through a cluster.
3. **categorize** success/failure of track to account for the cluster's charge distribution.

Some categories:

track	cluster is well characterized by the track.
shower	cluster consistent with an EM/hadronic shower.
short	cluster appears to be a "short track" (eg, δ-ray).
undefined	no well-suited categorization.

- This is an **active area of development** for Wire Cell.
- **Collaborate with machine learning experts?**.

Physics-level Reconstruction

This development is just beginning, still largely conceptual.
The usual obvious goals:

- associate clusters with a particle trajectory
- determine particle type
- determine momentum, range, dE/dx and other kinematics.

→ feed to Physics analysis.

Wire Cell Software Ecosystem

Wire Cell breaks up into three main parts:

visualization the "Bee" web application (Chao Zhang)

prototype reconstruction algorithms, initial proof of principle (Xin Qian)

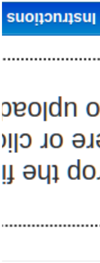
toolkit "production" toolkit for LArTPC reconstruction (bv)

Bee: an interactive 3D visualization system

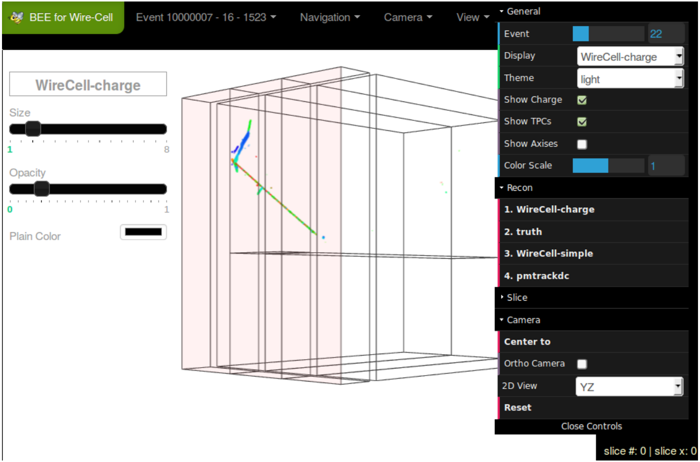
Bee features:

- Displays results of different reconstruction algorithms.
- Shows "true" particle trajectories from simulation.
- Very good for developers to debug and compare different methods and for users to gain Physics/LArTPC intuition.
- Implemented in **JavaScript/WebGL** + **Django**.
- Works on popular desktop and mobile browsers.
- Requires hardware WebGL support.
- JSON data file format, easy to implement schema.
- Supports drag-and-drop user file uploads.
- New features almost daily (**Chao Zhang**!)

Bee: Select and upload event sets



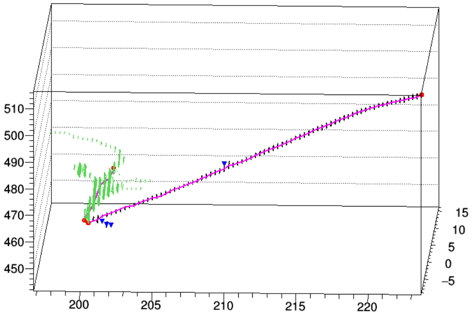
Interactive 3D visualization



Try it yourself! `http://www.phy.bnl.gov/wire-cell/bee/`

Wire Cell Working Prototype

- **Very successful proof of principle!**
- Currently **leads the state of the art** in LArTPC reconstruction techniques.
- Amazingly fast development (**Xin Qian**!)
  - → Work started only ∼5 months ago!



Colors indicate identified tracks and showers.

Compromises to Remedy

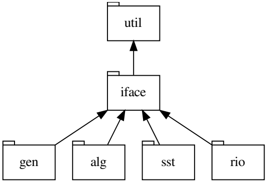
Some compromises were accepted to enable the **rapid prototyping** and now need "back-filling". Need to remedy:

- No support for long-term, multi-developer contributions.
- Many monolithic, single-threaded applications.
- Hard-coded "magic" numbers $\rightarrow$ configuration.
- Rigid code structure and hard-coded workflows.
- External software integrated only through file exchange.
- Need comprehensive CPU/memory optimization campaign.
- Some high-level tests, need coverage/granularity.
- Intimate dependency on the large ROOT[1] toolkit.

Decision: rewrite rather than refactor $\longrightarrow$

[1] https://root.cern.ch/

Wire Cell Toolkit - "Prototype 2.0"



High level overview:

- Includes **packaging and build** system.
- Comprehensive **API** via abstract base classes.
- Mindful of **dependencies**, external and internal.
- Built-in wire and cell **geometry descriptions** or load from file.
- Includes simple **LArTPC detector simulation**.
- Rewrite implementations of core **prototype algorithms**.
- Abstracted **execution model**, step towards parallelism.
- Support for native and foreign **file I/O**.
- Follow "**toolkit**" software paradigm.

Good progress along all fronts but still much work to do.

Source Repositories and Build

- Code in GitHub WireCell organization.
- Code aggregation with `git submodule`.
- A simple, customized waf-based build.[2]

```
$ git clone git@github.com:WireCell/wire-cell.git
$ cd wire-cell/
$ git submodule init
$ git submodule update
$ ./wcb --prefix=/path/to/install configure build install
```

Builds, tests and installs:

- toolkit shared libraries + header files,
- no main applications yet but many, unit+integration tests

Also: Doxygen code reference and MkDocs user documentation.

[2] `wcb` = `waf` + extra Waf tools for Boost, ROOT, Eigen and Wire Cell packaging

Wire Cell Interfaces

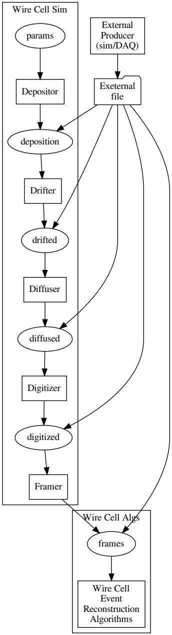
The `wire-cell-iface` package:

- API composed of abstract base classes covering:
  - **Data model** (wires, cells, frames, slices, tracks, ...)
  - **Active components** (blob maker, matrix solver, clustering, ...)
- Pervasive use of C++ `shared_ptr<>` for (mostly) **worry-free memory management**.
- Dynamic instance lookup via `NamedFactory` pattern allows for a **plugin architecture**.
- Initial support for **user configuration** system based on Boost property trees.
- Initial support exists for an **abstract execution model**.

  → more on this coming up.

Wire Cell Simulation

- Provided by the `wire-cell-gen` package.
- Simple implementation of the 4 "D"s of LArTPC: **deposit, drift, diffuse, digitize**.
- Granular, use as little or as much as needed.
  - → Parts can be replaced by **external simulation**, or bypassed entirely with **real DAQ data**.
  - → **External integration** via file I/O or API calls.
- Provides reference implementation to guide integrator/developers of external frameworks.
- Useful for quickly generating data to feed unit tests.



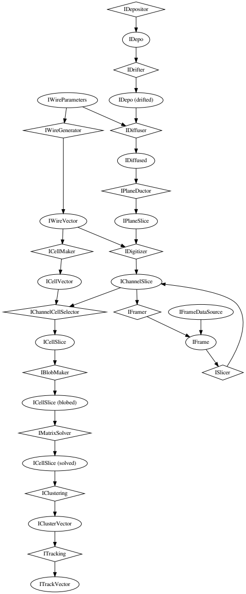
Wire Cell Execution Model

The toolkit supports **data flow programming** paradigm

- Design directly influenced by CSC Seminar on VisTrails, March 2013!
- Data flows through a graph made from:

  vertices: computational units / algorithms
  edges: data queues of a given type

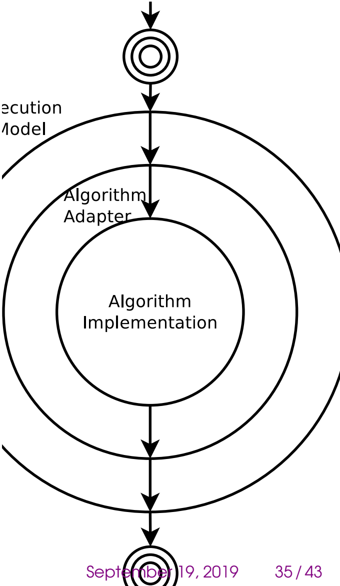
- Streamed processing minimizes RAM usage.
- Graph-level "programming" to define workflows.
- Thread-safe queues $\Rightarrow$ parallel processing.
- Graph execution machinery swapable: uniproc, multiproc, or distributed (MPI) parallel.
- Encourages isolated, targeted development and testing of each compute vertex.
- Feedback loops to implement iterative workflows.
- Instrument graph to collect performance data.



One possible high-level flow.

Abstract execution model

Want ability to change execution model while leaving “real” algorithm code untouched!



Each graph vertex made from **three layers**.

From outside to in:

1. Execution model layer implementing the data flow control
   - sync’ed load-and-flush, async, multi-processing, message-passing, ...
   - some implementations identified
2. Adapter wraps actual algorithm and presents uniform interface to execution model layer.
3. Algorithm implementation and interface left unrestricted except “no shared globals” (thread safety).

Future Plans

...in which I frequently beg for help :)

Pattern Recognition

Pattern recognition stage is turning out to be "hard".

- Many challenges are **inherent to the LArTPC technology**.
  - → Wire Cell has just allowed us to reach them.
- Current implementation is largely **heuristic based**
  - many "cross-algorithm assumptions"
  - **large solution space** with **many corner cases**.
  - human Physicists find good but **per-event/ad-hoc solutions**.
  - → how to teach them to the computer?
- → Our attempt: Bee 2.0 (next slide)

We have a few ideas but we feel we may be entering to the area of "real" computer science problems like **machine learning** and maybe **machine vision**.

And we are not experts.

Help Wanted!

Bee 2.0: Human-directed Automated Reconstruction

Interim Solution: inject human pattern recognition.

1. Bee will let humans **interactively inject commands**, eg:
   1. "Join cluster X to cluster Y"
   2. "Remove blob AAA from cluster Z"
   3. "Now, rerun automatic reconstruction"
2. Package human intentions into command objects.
3. Send to back-end, interpreting and execution inside a Wire Cell service.
4. Send results back to user, possibly iterate.
5. Record everything for later replay and to mine for potential Wire Cell improvements.

Think: "**Physics-PhotoShop**" + "**Google Analytics**".

We have a rough conceptual architectural design but it will be a **rather big** system. ⇒ **Help wanted!**

Wire Cell Production Processing

A **1-2 punch** for today's computing:

$\rightarrow$ Need **high thoughput** to keep up with detector data rates.

$\rightarrow$ Need **high performance** due to CPU-heavy algorithms.

Data volumes:

MicroBooNE	$\sim$630 TB/year (compressed)
protoDUNE	$\mathcal{O}$(1PB) total, (ZS, uncomp.)
DUNE 40kt	$\mathcal{O}$(PB)/year (ZS, uncomp.)

- Okay, it's not quite ATLAS-levels
  - Run 1 produced $\sim$3PB/year raw.
- But: **Wire Cell needs all raw wire waveforms** as input
  - No "Level-X" triggers to save us.
  - Waveform thresholding, but otherwise everything.
- Current speed is **30 minutes – 1day / event!**
  - **Major caveats**: we expect some optimization still to be had, we will parallelize the code, etc, etc,
  - But: ... ***GULP*!**

(with apologies to Frank Wuerthwein)

Taking Wire Cell Parallel

Our plans for **parallel processing** mostly involve implementing and testing different **parallel execution models**:

1. Boost.Pipeline nice and simple threaded data flow package. Easy to start with but not yet officially part of BOOST (longevity concerns?).
2. Intel TBB a common and well supported library with a fantastic looking "flow graph designer" tool. Some effort needed to learn/adopt.
3. MPI/ZeroMQ or other ways to run Wire Cell as multiple-node distributed process (maybe on **HPC**). Big learning curve for us.
4. Finally we will investigate potential for **GPU/Phi** hardware acceleration of bottlenecks. Even bigger learning curve!

This level of parallelism is **new ground for us**[3].

Help wanted!

[3] This is a subject of a pending BNL/Computing LDRD proposal.

Summary

- The **Wire Cell working prototype LArTPC reconstruction** method and software has been developed.
  - already producing some of the **world's best results**,
  - technical and performance improvements still needed.
- The **Bee interactive 3D event visualization** application developed, critical for understanding LArTPC and developing Wire Cell. Plans to evolve to "human-directed automated reconstruction" system.
- The **Wire Cell Toolkit** for LArTPC reconstruction partly complete and will be the basis for long term development including investigations into parallel processing.
- To be successful we must wade into **new waters** (for us) of parallel processing, hardware acceleration, computing science and mathematics.

$\rightarrow$ **Expert input, help and collaboration are most welcome!**

Wire Cell on the web

- home page:
  http://www.phy.bnl.gov/wire-cell/
- Bee entry page:
  http://www.phy.bnl.gov/wire-cell/bee/
- prototype user manual:
  http://bnlif.github.io/wire-cell-docs/
- prototype repositories:
  https://github.com/BNLIF
- toolkit user manual:
  http://wirecell.github.io/wire-cell-docs/
- toolkit code reference:
  http://www.phy.bnl.gov/wire-cell/doxy/html/
- toolkit repositories:
  https://github.com/WireCell