Wire-Cell Toolkit Wires

Brett Viren

December 16, 2022

Topics

- overview

Wire data provenance

Caveat, LArSoft parts may not be exact as described.

1. Some Perl script generates wires in GDML format.
2. Special LArSoft code loads GDML, dumps “text file”
3. Convert to WCT wire JSON format.
4. Save official files to `wire-cell-data` repo.
5. WCT C++: `WireSchema::load("wct-wires.json.bz2")`

Problem: precision of wire data is not great

pitch	mean	rms	min	max	p0
pdsp					
U (mm)	4.6691	0.000006	4.6684	4.6693	4.6691
V (mm)	4.6691	0.000005	4.6684	4.6693	4.6691
W (mm)	4.7920	0.000000	4.7920	4.7920	4.7920
uboone					
U (mm)	3.0028	0.000033	2.9938	3.0045	2.9997
V (mm)	3.0028	0.000033	2.9938	3.0045	2.9997
W (mm)	3.0000	0.000000	3.0000	3.0000	3.0000

- The “p0” is the pitch between first two wires (importance later).
- Unknown if precision is lost in wring GDML or the wires “text file”.
- The trip through JSON should preserve IEEE 64bit/double precision format.

Numbers from: `wirecell-util wires-info wct-wires.json.bz2`

MB end-wires pathology

- The last two wires at the end of U and V are very short
  - channels 2398,2399 and 4798,4799
- Happens to be rather co-planar but 0.13° non-parallel.
  - Results in near zero pitch (ray of closest approach).

```
TINY PITCH: aid=0 fid=0 pid=0 wid=2399 chn=2399 -- >  wid=2398 chn=2398
prev: [(6.33552e-13 -1155.1 10358.5) --> (6.34915e-13 -1148.67 10369.6)]
next: [(6.34287e-13 -1155.1 10364.5) --> (6.34915e-13 -1152.13 10369.6)]
TINY PITCH: aid=0 fid=0 pid=1 wid=2399 chn=4799 -- >  wid=2398 chn=4798
prev: [(-3 1168.07 10369.6) --> (-3 1174.5 10358.5)]
next: [(-3 1171.53 10369.6) --> (-3 1174.5 10364.5)]
```

Solving precision problems

Not sure what best to do.

Some ideas:

- Regularize by moving the input wires to their ideal locations based on either user-configured or calculated averages.
- Implement original wire generator algorithms in either WCT Python to still generate WCT JSON wires file or generate wires directly in in WCT C++ component.
- Address the symptoms of this imprecision (some are listed next).

All pose some concerns.

Ray grid reminder to understand the impact on tiling

A **ray** is an ordered pair of "tail" to "head" endpoints, $(\vec{t}, \vec{h})$.

- A **wire segment** is a ray
  - $\mathcal{W}_i \equiv (\vec{t}_i, \vec{h}_i)$.
  - $\mathcal{W}_i$ and $\mathcal{W}_{i+1}$ are separated by some pitch $\vec{P}$.
- A **wire region** has a ray representing its lower bound
  - $\mathcal{R}_i \equiv (\vec{t}_i - \frac{1}{2}\vec{P}, \vec{h}_i - \frac{1}{2}\vec{P})$

A **ray-grid layer** is defined by a **ray pair** $(\mathcal{R}_0^l, \mathcal{R}_1^l)$.

- $l \in \{0, 1\}$, "active region" layer types bound overall vertical and horizontal.
- $l \in \{2, 3, 4\}$, "wire region" layer types corresponding to wires in planes.

Problem: “active region” coincides with wire endpoints bounding box

A blob **corner** is a **crossing** of two **rays** each from their own layer.

- The “active region” rays coincide with anode face wire-endpoint bounding box.
- This leads to comparing wire-centers to wire-region bounds.
- Corners just outside the active region are unintentionally excluded from their blob.

Solution: pad the “active region” bounds by some amount.
Most simply, pad each side by 1/2 the maximum pitch over the wire planes.
More precisely, would take into account wire angles and pattern offsets.

Extrapolation errors become progressively worse across plane

wire num/tot	25%	50%	75%	100%
pdsp				
U (mm)	0.04	0.07	0.11	0.14
V (mm)	0.13	0.26	0.40	0.53
W (mm)	2.2e-11	4.4e-11	6.6e-11	8.8e-11
uboone				
U (mm)	0.004	0.013	0.009	0.010
V (mm	0.004	0.013	0.009	0.010
W (mm)	0	0	0	0

Extrapolate $\mathcal{W}_0$ center to actual $\mathcal{W}_i$ center (as from JSON file) and compare.

- Use pitch vector and wire direction vector from $(\mathcal{R}_0, \mathcal{R}_1)$.
- $\vec{p}_f = \vec{c}_0 + n_f \vec{P},\ \delta = \hat{W} \cdot (c_f - p_f),\ p'_f = p_f + \delta \hat{W},\ err = |p'_f - c_f|$
- $\vec{p}$ predicted, $\vec{c}$ actual wire center, f wire number, $\vec{P}$ pitch vector, $\hat{W}$ wire direction

Problem: layer definition maximizes extrapolation error

- A ray-grid layer for wires is defined by $\mathcal{W}_0$ and pitch to $\mathcal{W}_1$.
  - These are the **shortest wires** in a plane (ignoring horiz/vert wire planes)
- Uncertainty "lever arms" are not in our favor:
  - Errors in endpoint locations propagate worse to center/angle for short wires.
  - Extrapolation errors grow linear as one goes away from $\mathcal{R}_0$.
  - $\mathcal{R}_0$ at one extreme side of detector so maximizes the extrapolation.

Solutions:

1. Find **average** wire **direction** and **pitch** vectors over entire plane.
2. "Back-define" an "average" zero wire based on the center (longest) wire:
   - $\overline{\mathcal{W}}_0 \equiv \mathcal{W}_{(\frac{1}{2}N_{wires})} - \frac{1}{2}N_{wires}\vec{P}$
   - Effectively, this is the "regularize" solution from above.

Problem: pathological pattern precision

- Detectors like MB have 3 ray-grid rays meeting at the same point.
- Even with perfect precision, blob-inclusion tests must be based on a closed-interval (not half-open).
- Finite precision leads to just-in / just-out inclusion test failures (both false positive and negative).

Solution: "nudge" a crossing slightly toward the center of the blob prior to testing for inclusion.

WCT wire convention requirements

- Wire direction $\hat{w}$, unit vector pointing from wire tail to wire head, uniform across plane.
- Pitch direction $\hat{p}$, unit vector perpendicular to W, coplanar with and uniform across all wires in a plane.
- Pitch magnitude, p, perpendicular separation between any two neighboring wires in a plane, uniform.
- Cross product rule: $\hat{x} \times \hat{w} = \hat{p}$
- Sign of $\hat{p}$: if $\hat{p} || \hat{z}$ then $\hat{p} \cdot \hat{y} > 0$ else $\hat{p} \cdot \hat{z} > 0$.

Problem: wire files are largely invalid

With good excuse, requirements are ill documented and WCT does not validate!

- Sign of $\hat{p}$ is almost universally violated (MB is okay)
  - Many "violators" get it piece-wise correct and piece-wise inverted.
- Finite precision causes small uniformity and orthogonality violations.

WCT mitigation strategy

Implement 4 levels of wire corrections, all on by default:

load	just load wires as-is from file
order	fix wire endpoint and wire-in-plane order
direction	make all wires parallel to the “mean wire”
pitch	make coplanar and all pitches equal to “mean pitch”.

These are applied during loading of “wires file”.

- Assure consistency across all code.

Results for MicroBooNE U-plane

V-plane similar to U-plane and W-plane requires only ordering fixes.

MicroBooNE pitches and Y-axis ∠

	mean	rms	min	max
P before	2.9988	0.06125	2.6e-10	3.00751
P after	3	0	3	3
∠ before	149.959	1.98976	52.542	150.001
∠ after	150	0	150	150

Relative size of corrections for U plane

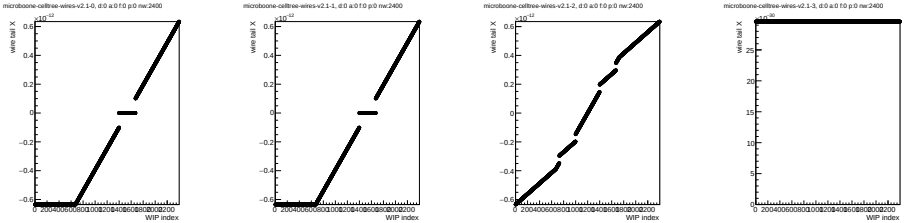
	mean	cumulative
pitch	6.1 μm	14.7 mm
tail	6.1 μm	14.8 mm
head	8.5 μm	20.4 mm

MicroBooNE plots

Next plots are in columns according to which progressive correction.

| raw | order | direction | pitch |

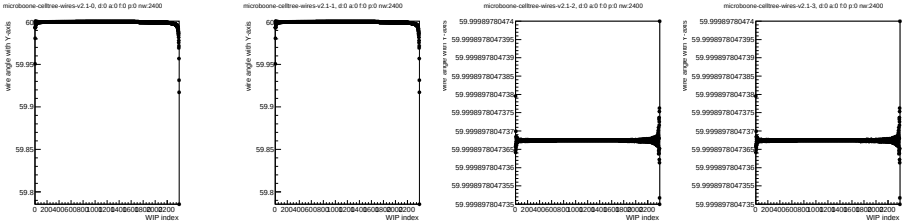
MicroBooNE wire tail ends



Tail X coordinate as function of wire-in-plane.

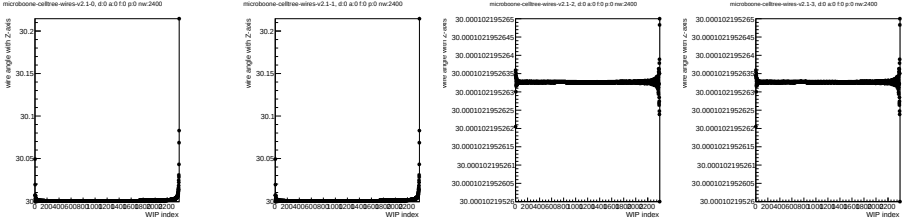
- Ray wire endpoints are very slightly not coplanar to start with (10^{-12} mm).
- As part of pitch corrections they are brought exactly into their plane.

MicroBooNE wire angles with Y-axis



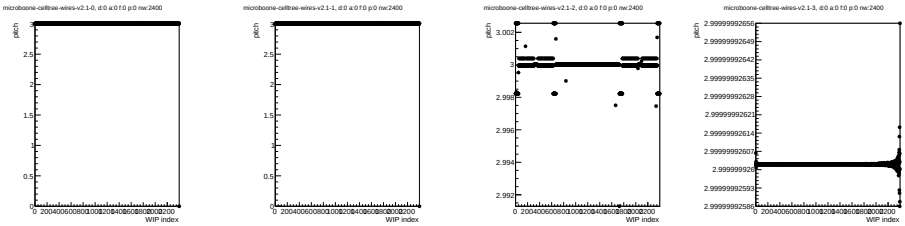
- Short wires at either end are particularly bad.
- $0.2°$ worse case deviation in raw, corrected to $\approx 10^{-6}$

MicroBooNE wire angles with Z-axis



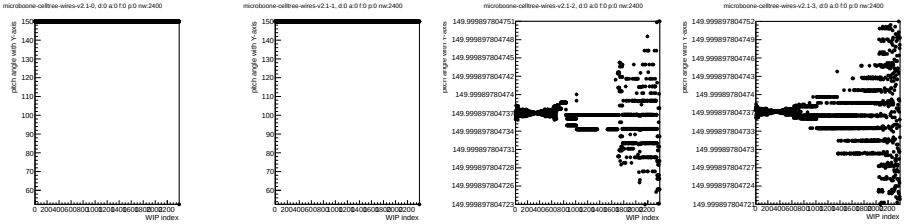
- Similar as Y-axis angle

MicroBooNE pitch magnitude



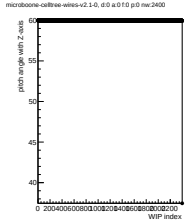
- Note, very wrong pitch in “raw” wires at plane end
  - Due to short, $0.13°$ non-parallelism
- Corrected uniformity to $\approx 10^{-10}$
  - $\approx 10^{-19}$ within nominal 3 mm.

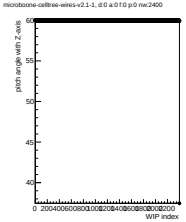
MicroBooNE pitch angle with Y-axis

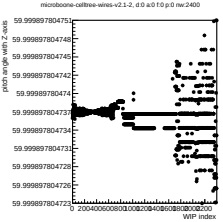


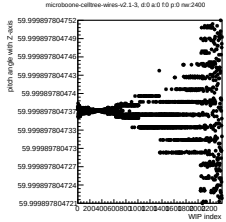
- The “very wrong pitch” case seen here as well
- Correcting direction alone clears up
- Full correction gives $\approx 10^{-12}$ precision and $\approx 10^{-6}$ accuracy.

MicroBooNE pitch angle with Z-axis









- Similar to Y-axis