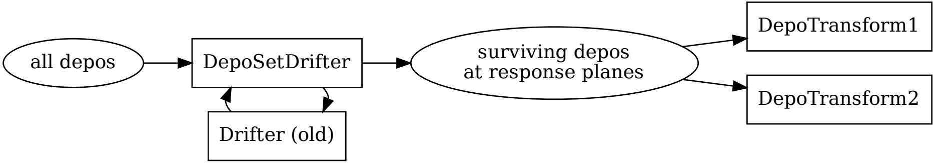
“Bent” Cathodes

Brett Viren and Wenqiang Gu

September 10, 2024

Context

Cartoon of Drifter-related data-flow graph.

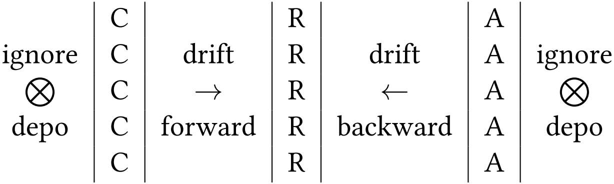


- All depos go to all `DepoTransform`'s which then select based on active area.

Selecting and drifting depos

Old `Drifter`'s behavior vs X-position of depo w.r.t **C**athode, **R**esponse, **A**node planes:

- The "cathode" and "anode" **planes** only for selecting depos and have no physical meaning.



	C		R		A	
ignore	C	drift	R	drift	A	ignore
⊗	C	→	R	←	A	⊗
depo	C	forward	R	backward	A	depo
	C		R		A	

Behavior is mediated by one `Xregion` instance for each contiguous drift region.

- Eg, entire DUNE HD has 4 Xregions: C[-]A[-]C[-]A[-]C.

Support for a “bent” cathode

Icarus, “we wanna model the X position of the cathode as a function of the Y and Z positions”.

$$X_{cathode} = f(Y, Z).$$

Constraints (from me):

- Do not make Drifter more complicated.
- Keep backwards compatibility in configuration for current usage.
- Keep single `Drifter` for whole detector.

GitHub action: Issue 329 and PR 336.

- Props to Wenqiang for excellent back-and-forth design discussion!
    - Though extended, it allowed the implementation to be fairly trivial.

New `CoordRegion.h`

Extend `Xregion` geometry functions and move them from `gen/` to `util/`.

- `CoordBounds` **abstract base class** models a drift (X-axis) boundary.

  `.distance(pt)` pt → boundary (signed).

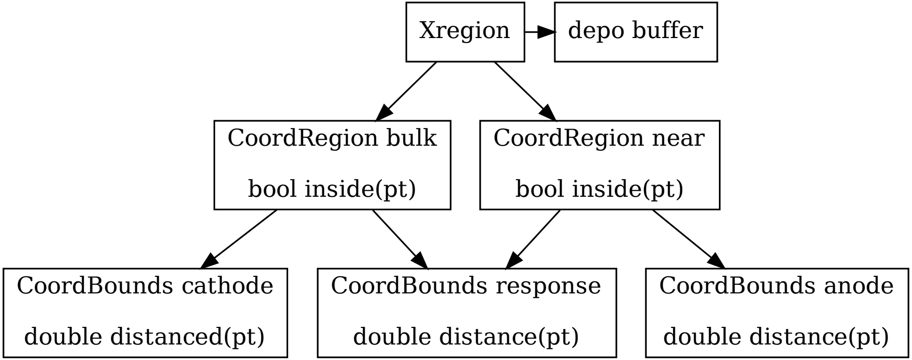
  `.location()` the nominal X-position (to locate "response" plane).

  `.operator<(Point,CoordBounds)` partial ordering using `.distance(pt)`.
- Arbitrary shaped boundary but "aligned" in some way with X-axis / drift direction.
- `CoordRegion` holds a `{lo,hi}` pair of bounds and provides:

  `.between(pt) { return lo < pt.x <hi; }` is point inside me?

  `.inside(pt)` same, but allows for `hi < lo` (makes Xregion simpler).
- `CoordBounds* make_coorbounds(Configuration)`
  - Creates and configures concrete type of `CoordBounds` based on `cfg` object.
- `CoordBounds{Scalar,Ray,Sampled}`
  - 3 specific models: **normal** plane (original), **tilted** plane, arbitrary **surface**.

New `Xregion` structure



Xregion

depo buffer

CoordRegion bulk

bool inside(pt)

CoordRegion near

bool inside(pt)

CoordBounds cathode

double distanced(pt)

CoordBounds response

double distance(pt)

CoordBounds anode

double distance(pt)

Configuration examples

```
local ra = { response: 10*wc.cm, anode: 0 };
// ...
{ type: "Drifter", data: {
    xregions: [
        // "scalar" aka "normal plane" (original)
        ra { cathode: 2*wc.m },
        // "ray" aka "tilted plane"
        ra { cathode: {
            tail: [ /* point on plane */ ],
            head: [ /* from tail to point on normal */ ],
        }},
        // "sampled" arbitrary surface
        ra { cathode: {
            x: [ /* array of x values */ ],
            y: [ /* array of y values */ ],
            z: [ /* array of z values */ ],
        }}],
    // ...
    }
```

`CoordBoundsSampled` - how to get bent

Perform a 2D linear interpolation on sampled points:

- User provides set of `{x,y,z}` points **sampling the surface**.
- Points are built into a **3D k-d tree**.
- $k_{NN}(depo, k = 3) \rightarrow 3$ nearest points to a depo.
- Analytical function for the plane containing the three points.
- Calculate X-distance from depo to plane, returned by `distance()`.

Tests and validation

- Test `CoordRegion.h`, includes a speed test targeting the k-d tree.

```
./build/util/wcdoctest-util -tc='coord *'
  709.000 ns      make coord region for 100 samples
  535.766 us      check 1000 depos against 100 samples
  517.705 ms      check 1000000 depos against 100 samples
  22.053 us       make coord region for 1000 samples
  785.587 us      check 1000 depos against 1000 samples
  632.623 ms      check 1000000 depos against 1000 samples
  68.464 us       make coord region for 10000 samples
  2.216 ms        check 1000 depos against 10000 samples
  814.564 ms      check 1000000 depos against 10000 samples
```

- Test `Drifter`, with focus on depos near boundaries.

```
./build/gen/wcdoctest-gen -tc='drifter *'
```

- Wenqiang: full sim test finds less than ±1 ADC change with `CoordBoundsScalar`'s.
  - Plots in PR 336.

Do we need anything more fancy than `CoordBoundsSampled` ?

The k-d tree performs 1M queries of 10k samples in less than 1s (slowish i7 CPU).

- Is this fast enough for cases where a non-planar model is needed?
- Are there simpler geometries that yield a faster `distance()` implementation?
  - And/or can be configured more conveniently than via sample points?

Some possible models of intermediate complexity:

- An **analytic function** for the `distance(point)` can be written.
- A **piece-wise homogeneous** model.
  - Eg: the physical cathodes are planar but not co-planar.
  - A new `CoordBounds` would map Y/Z rectangle to another, "real" `CoordBounds`.
  - Reflect this recursion in `make_coorbounds()` and configuration schema.

FIN

Depo exclusivity

In `Drifter` the first `Xregion` in which a depo is “inside” gets the depo.

- Only an issue if two `CoordBounds` physically overlap.
- Could extend this behavior to allow a depo to be “in” multiple `Xregion`.

This would not be physical, but maybe the experiment has physically inconsistent models for either side of a cathode?