Blob Depo Fill

Brett Viren

July 23, 2024

Contents

1 Overview **1**

2 Algorithm **1**

3 Selection **2**

4 Alignment **2**

This replaces charge in blobs with that from overlapping depos.

1 Overview

The `BlobDepoFill` WCT flow graph node is in the "join" DFP category. It accepts an `ICluster` on input port 0 and an `IDepoSet` on input port 1 and produces an `ICluster` on output port 0.

The output retains all information from the input `ICluster` except that the charge of each blob is replaced with the charge from the portions of the input depo distribution which falls in the blob volume. The section 2 below describes how the depo charge is assigned to blobs. See section 3 for details on which depos are considered and section 4 for details on how blobs and depos are correlated.

See also presentation blob-depo-fill.pdf

2 Algorithm

The goal is to integrate the depo charge distribution over the volume of each blob. This ends up being rather complex for a few reasons.

- depo-to-blob overlap represents an N-to-M mapping with no direct way to form.
- both depos and blobs are numerous and the mapping is sparse.
- depos have truncated 3D Gaussian extent both longitudinally and transverse and this extent varies substantially across the set of depos due to their differing drift times.
- blobs have irregular and varied shape defined in terms boundaries which themselves are defined by wire rays on each of the three planes.

The algorithm joins the volumes of depos with the volumes of blobs in a series of reductions of the full possible space.

1. Narrow to a single `IAnodeFace`
2. Narrow to a single `ISlice`
3. Produce an "`s-d-w` graph" (described below)
4. Narrow to a single `IDepo`
5. Narrow to a single `IWire`
6. Iterate over (the narrowed) blob set finding those with this wire
7. Find bounds on this wire from other two blob layers.

The `s-d-w` directed (acyclic) graph associates slices to depos (`s-d` edges) and depos to wires (`d-w` edges). Both edge types represent the orthogonal integration of the depo Gaussian over a section in one dimension.

The `s-d` edge property holds the integral of the the Gaussian along the longitudinal dimension bounded by the slice's time span.

Likewise, the `d-w` edge property holds the integral of a section of the Gaussian in the transverse (pitch) direction of the wires of the so called "primary wire plane". Which wire plane to use as "primary" is configurable and the collection plane (index `2`) is the default. The `d-w` edge holds the result of this integral.

As the algorithm descends the narrowing listed above from slice to depo to wire it also descends the `s-d-w` graph. Reaching the wires, the full `s-d-w` path is known and each blob in the current face and slice context is checked to determine if it contains the current wire. When it does the final integral of the Gaussian is performed. This requires calculating the bounds on the current wire provided by blob boundary rays from the other planes. The three integrals from each dimension then scale the depo charge and the result is added to a sum associated with the current blob.

3 Selection

The `Drifter` produces depos at a fixed location along the drift direction which is near to anode plane faces. However, the set of depos may extend transversely beyond the sensitive bounds of any one anode plane. A `DepoBlobFill` will accept only those depos within the sensitive area of the `IAnodeFace` instances associated with the input blbos. This input selection is identical to that used by `DepoTransform`.

As a consequence, a `BlobDepoFill` may be used either in a flow graph context where it receives an `ICluster` with blobs from a single `IAnodePlane` yet depos which may span many or it may be used in a context where the `ICluster` spans multiple anode planes (eg, after some form of cross-anode "stitching").

4 Alignment

In order to correlate portions of the 3D Gaussian extent of a depo with a blob, it is likely necessary to apply a time offset to match any time shifts applied between `Drifter` output and the production of the blobs which are input to `BlobDepoFill`. At the least, `DepoTransform` may apply a time shift via its `start_time` configuration parameter. The `BlobDepoFill` accepts a `time_offset` parameter which will be added to depo times just prior to matching them with blob time slices.

To check for a time offset one may create paraview/vtk files for both the drifted depos and the reconstructed blobs:

```
$ wirecell-img paraview-depos \
  --speed '1.56*mm/us' --t0 '314*us' \
  depos-drifted.npz depos-drifted.vtp

$ wirecell-img paraview-blobs \
  --speed '1.56*mm/us' \
  clusters-img-5.zip clusters-img-5.vtu

$ paraview
```

Open the `.vtp` and `.vtu` and check alignment along the X-axis.

Note the `--t0` **adds** the offset to the depo time. Likewise, the a `time_offset` value given to `BlobDepoFill` will be **added** to the depo times prior to locating them within blobs.