Blob Depo Fill

Brett Viren

July 23, 2024

Contents

1 Overview **1**

2 `BlobSampling` **1**

3 `BlobSampler` **2**
3.1 Configuration parameters . . . 2
3.2 Strategy . . . 2
3.2.1 `center` . . . 2
3.2.2 `corner` . . . 3
3.2.3 `edge` . . . 3
3.2.4 `grid` . . . 3
3.2.5 `bound` . . . 3
3.2.6 `stepped` . . . 3
3.3 Time binning . . . 3
3.4 Attribute matching . . . 4

4 `BlobSampler2D` **5**
4.1 2D strategy . . . 5
4.2 2D attributes . . . 5

1 Overview

To transition from the blobs produced by WCT tiling to the point clouds that WCT clustering operate on, the blobs must be sampled.

2 `BlobSampling`

The sampling is orchestrated by the `BlobSampling` data flow graph component. It consumes an `IBlobSet` and produces an `ITensorSet` representing a point cloud (PC). A "point" in the PC is an abstract element to which attributes are associated. The coordinates of the "physical point" that is sampled is represented by a specific subset of these attributes (eg, 3D Cartesian `x,y,z` or 2D view `pitch,wire` coordinates).

The `BlobSampling` itself merely provides object/graph I/O and delegates the sampling itself to an `IBlobSampler` tool component.

`sampler` name of the `IBlobSampler` that actually performs the sampling.

`datapath` root of the tensor data path (see tensor data model).

The `BlobSampler` and `BlobSampler2D` are two implementations of `IBlobSampler` and each are described next.

3 BlobSampler

The `BlobSampler` samples blobs over their 3D volume by applying one or more **sampling strategies**. In addition associating the physical point Cartesian coordinates with the PC point it may also associate a selection of a large set of additional attributes depending on configuration.

3.1 Configuration parameters

The following configuration parameters are accepted.

`strategy` sampling strategy. May be null, string, object or array or strategies. See 3.2.

`time_offset` a time added to blob times prior to any construction of time related point attributes.

`drift_speed` used in order to translate blob time to a spatial drift position.

`prefix` string to prepend to names of tensors in the resulting dataset.

`tbins` int, see 3.3

`tmin` float, see 3.3

`tmax` float, see 3.3

`extra` a list of strings, see 3.4

3.2 Strategy

The `strategy` parameter describes the strategies to apply for sampling the blob in the transverse plane. It accepts the description in flexible form which may be of the following data types:

`null` use default strategy ("center")

string provide the name for a sampling strategy, use its default configuration

object provide an object with the strategy string provided by a `.name` attribute and the remaining attributes providing non-default strategy configuration.

array an array of these types

When an **object** is provided, in addition to providing strategy-specific options, its attributes may override top-level configuration parameters for the strategy. For example, one may override `tbins` for a particular strategy.

Each strategy is named and described below.

3.2.1 `center`

Sample a single point at the geometric mean of the blob's corners.

3.2.2 `corner`

Sample point at each of the blob's corners.

3.2.3 `edge`

Sample point at center of each of the blob's edges.

3.2.4 `grid`

Sample points along the sub-grid formed by stepping the ray-grid crossing points of two given planes and requiring points to be also inside the third plane.

`step` the sub-grid step size (default is 1, in units of ray grid spacing)

`planes` an array of two plane numbers (default is `[0,1]`)

3.2.5 `bound`

Points are sampled by taking steps around the boundary of the blob. For each edge, the stepping starts but does not include the corner. Add the `corner` strategy to include those.

`step` how far to step (in units of distance)

3.2.6 `stepped`

Similar to `grid` but select the two plains and form the sub-grid with a per-plane step based on the "aspect" ratio of the blob. The sub-grid is formed as a subset of the crossing points of the view with the least and the most wires. the step is the maximum of a minimum step size and a step size that will take a maximum number of steps total.

`min_step_size` minimum number of wires a step can cross (default=3)

`max_step_fraction` the maximum fraction of total wires a step can span (default=1/12).

3.3 Time binning

The "time binning" defines how a blob is sampled in the time dimension or equivalently along the drift direction. The binning is specified with two bounds `tmin` and `tmax` and a number of bins `tbins`. The bounds are half-open and in units of one time slice. For each transverse sampling, a point in time on a lower-edge of a time bin will be taken.

As example, the default time binning sets `tbins=1`, `tmin=0` and `tmax=1`. With this, all points that are sampled in the transverse plane will be given a common time (and X drift location) at the start of the slice. On the other hand, with `tbins=2` then transverse samplings will be duplicated with one set at the start of the slice and one half-way in the middle. Alternatively, with `tbins=1` and `tmin=0.5` the transverse samplings will be placed longitudinally in the middle of their time slice.

3.4 Attribute matching

Every point has a set of associated attributes. This set is includes mandatory coordinate attributes and optional attributes specified by the `extra` configuration parameter. The coordinate attributes are:

`x_coord` (double) point X-position in the global coordinate system

`y_coord` (double) point Y-position in the global coordinate system

`z_coord` (double) point Z-position in the global coordinate system

`t_coord` (double) point time

Note, `x_coord` and `t_coord` are simply derived from the `tick` via the `time_offset` and `drift_speed`.

The optional attributes are specified by the `extra` configuration parameter. This provides one or more strings interpreted as regular expressions. They are matched against the names of available attributes from the following list:

`sample_strategy` (int) the ID of the sampling strategy that produced the point.

`blob_ident` (int) the blob ident number.

`blob_index` (int) the index of the blob in the input vector.

`{u,v,w}blob_begin` (int) wire-in-plane index that begins the bound of the blob in the plane.

`{u,v,w}blob_end` (int) wire-in-plane index that ends the bounds of the blob in the plane.

`slice_ident` (int) the slice ident

`slice_start` (double) the start time of the slice.

`slice_span` (double) the time span of the slice.

`tick` (int) the time sample

`{u,v,w}wire_index` (int) index of nearest wire in a plane (`IWirePlane::wires()` order)

`{u,v,w}pitch_coord` (double) location of point along the pitch direction.

`{u,v,w}wire_coord` (double) location of point along the wire direction.

`{u,v,w}channel_ident` (int) the ident number of nearest channel (`IChannel::ident()`).

`{u,v,w}channel_attach` (int) the attachment number of nearest channel (`IChannel::index()`).

`{u,v,w}charge_val` (double) the charge value of nearest channel.

`{u,v,w}charge_unc` (double) the charge uncertainty of nearest channel.

4 BlobSampler2D

The `BlobSampler2D` works similarly to `BlobSampler` but is somewhat more simple. It accepts the following configuration parameters.

`strategy` sampling strategy. May be null, string, object or array or strategies. See 4.1.

`time_offset` a time added to blob times prior to any construction of time related point attributes.

`drift_speed` used in order to translate blob time to a spatial drift position.

`prefix` string to prepend to names of tensors in the resulting dataset.

`extra` see 4.2.

4.1 2D strategy

The 2D tomographic views are sampled. This is a sampling of "pixels" in the channel vs tick readout "image" for the view. A "pixel" is considered "in" a blob based on its lower channel and tick bin edges. The tick must be within the half-open time slice defining the blob. The channel is in the blob in a given view if any wire feeding the channel is in the half-open blob bounds in that view.

The current strategies are defined:

`pixel` all pixels in a blob are sampled.

4.2 2D attributes

The 2D coordinate attributes are:

`tick` (int) the tick

`channel_attach` (int) the wire-attachment-number of the channel

`plane_index` (int) the index of the plane assuming `[u,v,w]` order

Together these make up the "pixel coordinates" of a 2D sampled point.

The `extra` configuration parameter gives strings interpreted as regular expressions to match against a set of optional attributes to include in the resulting point cloud.

`sample_strategy` (int) the ID of the sampling strategy that produced the point.

`blob_ident` (int) the blob ident number.

`blob_index` (int) the index of the blob in the input vector.

`blob_begin` (int) wire-in-plane index that begins the bound of the blob in the plane.

`blob_end` (int) wire-in-plane index that ends the bounds of the blob in the plane.

`t_coord` (double) point time found from tick and t_{offset}

`slice_ident` (int) the slice ident

`slice_start` (double) the start time of the slice.

`slice_span` (double) the time span of the slice.

`channel_ident` (int) the channel ident number

`charge_val` (double) the sampled charge (activity) value of channel and tick "pixel".

`charge_unc` (double) the uncertainty in `charge_val`.

Any attributes produced with the same name as used by `BlobSampler` have identical meaning.