WCT version of PR3DCluster

Providing a **portion** of what is used by WCP

Brett Viren

August 28, 2023

Overview

*New **blob** and **cluster** classes for porting WCP pattern recognition to WCT.*

And with various questions/unknowns/todos.

Currently in the `apply-pointcloud` branch.

https://github.com/WireCell/wire-cell-toolkit/blob/apply-pointcloud/util/inc/WireCellUtil/PointCluster.h

Scope (for now)

For now, focus on **context of single APA-face**.

- Eventually extend to **two-face** and **multi-APA** contexts (not part of WCP/MicroBooNE model).

Also, just **data representation**, **primitive operations** and **collections**.

- Expect to construct and serialize blob/cluster with other code.

PointCloud::Blob

A **pseudo box** in 4, non-orthogonal dimensions: **u,v,w** and **t**.

- **Discrete dimensions**: counts of **wires** or **ticks** (8 `int` values)
- **Half-open intervals** bound blob in each dimension (C++ iterator convention).

Primary purposes:

- Blob-level **geometry operations**: "blob connectivity", "cluster overlap", etc.
- A **constituent** of a **cluster** that can be **associated** with other data.
- A **value** object to support types/code like:

hashable	`std::map<Blob,T>`
copyable	`Blob a = b;`
comparable	`if (a == b) {...}` (compares intervals)
ordered	`if (a < b) {...}` (by "volume")

Blob methods and functions

`blob.sizes()` vector of sizes of intervals in each dimension (u,v,w,t).

`blob.empty()` true if any interval is empty

`blob.volume()` a pseudo-volume: $dt \times min(du, dv, dw) \times max(du, dv, dw)$

`envelope(blobs)` a blob that surrounds a set of blobs

PointCloud::Cluster

Cluster is **set of blobs** plus **blob-derived data**

- blob-derived: PCs, k-d trees, PCAs, PC-graph (maybe)
- blob-level PCs are **registered** with a **name**.
- blobs associate to their named blob-level PCs.
- cluster-level PC is **concatenation** of blob-level PC of same **name**.
- blobs have unique ID in cluster (`IndexedMultiset`)
  - cluster provides **context** and **identity** to blobs.

Questions,

- Unsure if best to put **point-cloud graphs** inside/outside cluster.
- Must we support **removal** of blob(s) from cluster?

Cluster methods and functions

`insert(blob)` add a `Blob` to the cluster, returns unique blob index number.

`associate(name, index, pc)` add a named PC to a blob with index.

`envelope()` return pseudo `Blob` representing envelope of blobs.

`nearby(other, dist)` return set of this cluster's blobs "near" the `other` cluster according to `dist` (per-dimension intervals).

`kdtree(name, coordinates)` return a k-d tree query object on (whole cluster) PC of given name and list of coordinate attributes.

`pca(name, coordinates)` return PCA using listed coordinate attributes in named PC.

`overlap(one,two,dist)` return blobs from clusters `one` and `two` that are mutually within `dist` of each other.

- return as set of blobs? Or two clusters? or some kind of mapping?

Global point cloud

A cluster is primarily a collection of blobs and so we may use a **global cluster** of **all blobs** to represent a **global point cloud**.

- A `Blob` is light weight (8 ints) so copies are cheap'ish.
  - Okay to have same `Blob` value in more than one cluster.
- But, **full** PCs are "heavier" so some memory care is needed.
  - I have some things to try to address this.

Identity and its context

Blob has `std::hash<Blob>`

- This is a global ID by itself (assuming no hash collisions).
- Enables **index** in the **context of a cluster** (`IndexedMultiset<Blob>`).

Cluster has no identity

- We can hold `shared_ptr<Cluster>` and use pointer as an "identity".
- We can add `std::hash<Cluster>` *eg* as hash of blob hashes.
  - Then can use `IndexedMultiset<Cluster>`.

Is this enough?

If not, in what context do we assign unique Blob/Cluster IDs?

The name “cluster”

We have already many things named “cluster”

- `ICluster` and its “cluster graph”, `PR3DCluster` and now this new one.

Maybe we should avoid “cluster” here?

- `PointCloud::Blobset` ?

Or, though it violates C++ best practices, tack on extra qualifiers

- `PointCloud::PCCluster` ?