Thoughts on Clusters

Leon Rochester
June 10, 2004

• Clustering algorithm
• Content of clusters
• Representation of dead clusters
• Structure of TDS object
• How the cluster info is stored in the recon objects
How clusters are formed

Basically, the hits are arranged in ascending order in each plane, and then contiguous hits are grouped together. The interesting part is how we handle hot/dead strips.

Before making clusters, we merge the actual hit strips with the dead strips, with the latter marked. Then we form the clusters. When a gap is detected (this would signal the end of the clusters), we determine if there are any actual hits in the cluster just formed. If there is even one “good” hit, the cluster is saved, otherwise it is discarded.
Clustering, continued

So, any dead/hot strip that is contiguous to a good strip is added to the cluster. This is true whether the bad strips are within the cluster or just touching on the outside.

Three details:

• Clusters are not allowed to cross ladder boundaries. (The gap between ladders is 2 mm.)
• Any cluster with 10 or more strips is marked bad and removed. (Is this a good idea?? Probably not!)
• Dead and hot strips are treated the same way.
Clustering, III

Once a cluster is formed, we don’t know what the bad-strip content was. This should be easy to fix, perhaps simply by adding a member containing the number of bad strips!
Bad strips at recon time

Currently, during patrec, if we project a track into a plane and find no hit, we have no good way to ask if the track passes near a dead strip. If we could, then in some sense, a missing hit would be almost as good as a real one! (But I think that we should only look for bad clusters if we don’t find a good one.)

Two fixes:

• Keep the bad clusters in the list of clusters, but marked.

• Make a separate set of bad clusters, which can be accessed separately.
Bad strips, II

If all the clusters are in the same list, we need some mechanism to return a list of real clusters and bad clusters separately.

Is this easy? Can an iterator be written that goes over the clusters an only returns when a good (or bad) cluster is found? Should be possible, but don’t know how at the moment.

Is having all the clusters in the same list useful for anything?

Is there a downside to keeping two separate lists? Clarity? Performance?
Currently, there is a vector of pointers to clusters, and then a 2-D array (layer, view) of the same pointers.

- ClusterCol doesn’t look like any of the other Cols, which are all just typedefs.

But we do want the clusters accessible by plane, so maybe we want

- an objectVector of objectVectors?
- One objectVector with a separate class that organizes it?
Cluster info in the recon classes

The current recon classes store the ID (index) of the cluster. This seems to go against the spirit of OO, in that we are assuming something about the storage of the hits, namely that they're in a vector.

We should probably modify the classes that store the cluster ID to use a smart ref or smart pointer instead. Again I don't know how to do this, but I think Tracy does.