ActRoot manual

Miguel Lozano González

Contents

1 Introduction 1

2 Algorithms 1

2.1 Filtering algorithms 1

2.1.1 Common tasks 2

1 Introduction

ActRoot aims at being a simple program to convert, process, and obtain the physical data for an ACTAR TPC experiment. The purpose of this manual is to provide a quick insight into the different algorithm settings for the *Clustering* part.

2 Algorithms

Two different types of algorithms to deal with voxels exist in the *Algorithm* folder of *ActRoot*:

- Inheriting from `VCluster`, they perform the basic clustering operation: joining voxels to form sets of voxels, i.e., clusters. Two have been implemented:
  - **RANSAC**: randomly samples two points from the cloud to form a line. Points close to it by a given distance threshold constitute the cluster. The process is repeated N times and only the *best-ranked* clusters are kept.
  - **Climb**: clusters points based on the continuity principle: neighbor points *should* belong to the same cluster. It was developed by J. Lois-Fuentes during his PhD [1], thus for more information see his thesis.

  Both can be configured in the `ransac.conf` or `climb.conf` files, respectively.
- From `VFilter`, this set of classes performs the treatment of the clusters, meaning this:
  - Cleaning noise
  - Merging broken clusters
  - Splitting clusters
  - Finding the **reaction point**
  - Improving the quality of the line fits

  Two are available now: `MultiStep` and `MultiRegion`. The former was employed for J. Lois experiment E796, while the latter is an *easier* version.

  Both leverage common functions that execute particular tasks. It is to this that we are going to explain in detail its workflow.

2.1 Filtering algorithms

Following the `VFilter` class, the shared structure for all the filtering classes is:

1. A pointer to the `TPCData` has to be set by `void VFilter::SetTPVData(TPCData* data)`
2. The algorithm is executed by `void VFilter::Run()`. This virtual function (i.e., mandatory to be implemented in any derived class) **sets the order of the inner functions**.

3. Other methods are available depending on the derived class.

2.1.1 Common tasks

There are a few steps that are usually needed by both classes. Hence, they are implemented in a *global* fashion under `ActAlgoFuncs.h`, `.cxx`.

These are:

- **Merging similar clusters**: a cluster might be broken into smaller parts as a result of charge collection inefficacies or damaged pads. Therefore, it is essential to remerge them. This function iterates over the cluster vector and, by pairs, joins clusters if some conditions are met:
  1. The distance between the central points of the two clusters is bellow `MergeDistThresh`.
  2. Both are quite parallel: their scalar product of fits is above `MergeMinParallel` $\in [0, 1]$.
  3. If 1 **and** 2 are met, the combined fit is done.
  4. If, and only if, the $\chi^2_{new} \leq$ MergeChi2Factor $\cdot \chi^2_{old}$, the larger cluster is merged into the smaller one and deleted. The scaling factor is usually ~ 1.5.
- **Compute a reaction point (RP) in 3D**: In general two lines in 3D do not perfectly intersect. On the contrary, the goal is to find the minimum approach distance and the two points on each line at which this occurs.

  This function schematically is defined as:

```
std::tuple<Point on A, Point on B, double min dist>
ComputeRPIn3D(Point A, Vector A, Point B, Vector B);
```

  Specifying the points and the direction vectors of each line, one is given the two points and the distance between them. The RP is afterward calculated as the mean.

References

[1] Juan Lois Fuentes. “Complete spectroscopy of 16C and 20O with solid and active targets using transfer reactions”. Available at http://hdl.handle.net/10347/30947. PhD thesis. Universidade de Santiago de Compostela, July 2023.