Cluster Matching Algorithms Documentation

David Caratelli, Ariana Hackenburg

June 19, 2015

Contents

1 Introduction

In order to fully reconstruct 3D objects from 2D signals in (wire, time) coordinates information from multiple wire-planes in a TPC must be combined. 2D clusters from multiple planes can be associated together to create a 3D shower. Any particle that deposits energy in the TPC will leave a signature on all 3 planes. Correctly identifying the deposited energy on different planes associated to the same particle is necessary to reconstruct 3D objects. We call this process **matching**.
Matching is a stage of shower reconstruction that follows cluster merging on individual planes and serves as input to the final shower reconstruction. Once clusters from two or more planes are associated together a 3D shower can be reconstructed.
A number of matching algorithms have been developed to perform this task. Each matching algorithm, given 2 or more clusters on different planes, returns a score that indicates the compatibility (or matching score) between those clusters. Several algorithms can be used in conjunction. A manager then handles how to use the information provided by the algorithm. In this document we will describe the matching algorithms currently available.

2 Matching Algorithms

2.1 CFAlgoStartPointMatch

This algorithm is useful only when three clusters on three different planes are given as input. The algorithm calculates a match score only by looking at the start point of the clusters on the three different planes. For each pair of planes the (Y,Z) intersection point of the wires on which the start points of the clusters are found is calculated. This way, three different (Y,Z) intersection points are calculated. A score inversely proportional to the area of the triangle bound by these three points is returned. A better agreement between the start wires of the clusters leads to closer (Y,Z) intersection points, which means a smaller area. If for any pair of clusters no intersection point is found, a negative score is returned. Fig. 1 illustrates how this algorithm works.

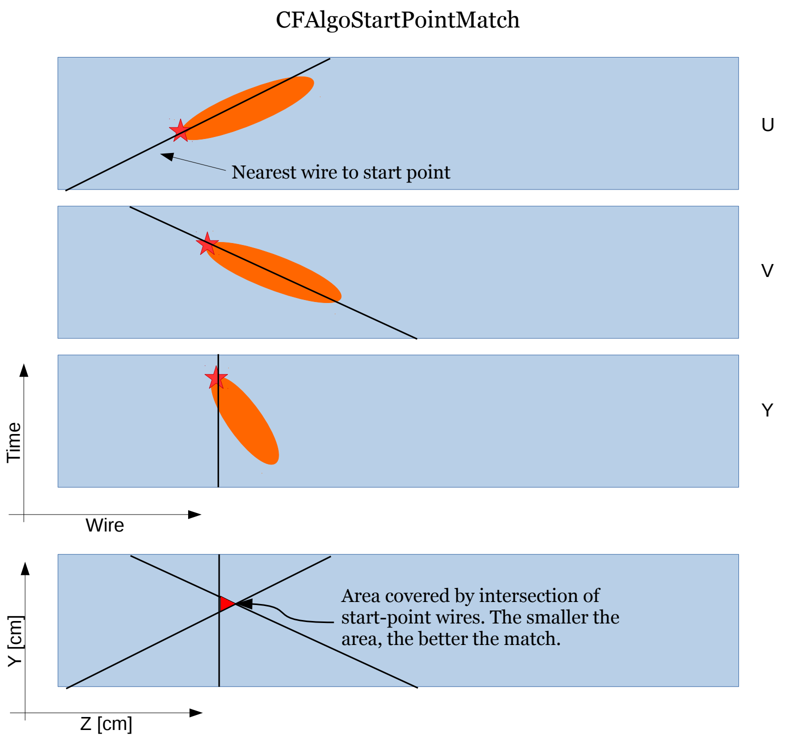
CFAlgoStartPointMatch
U
Nearest wire to start point
V
Y
Time
Wire
Y [cm]
Z [cm]
Area covered by intersection of start-point wires. The smaller the area, the better the match.

Figure 1: *Illustration highliting how CFAlgoStartPointMatch uses the start wire of three clusters to calculate a measure of the compatibility between the three clusters*

2.2 CFAlgoWireOverlap

This algorithm finds a rough measure of the area ovelapped by multiple clusters. For each cluster a band of wires determines a polygon on the (Y,Z) plane of the TPC. The intersection between different wire-bands on different planes is the wire-overlap that is used to determine the match score for this algorithm. Correctly matched clusters should have a large overlap. Fig. **??** illustrates how this algorithm works.

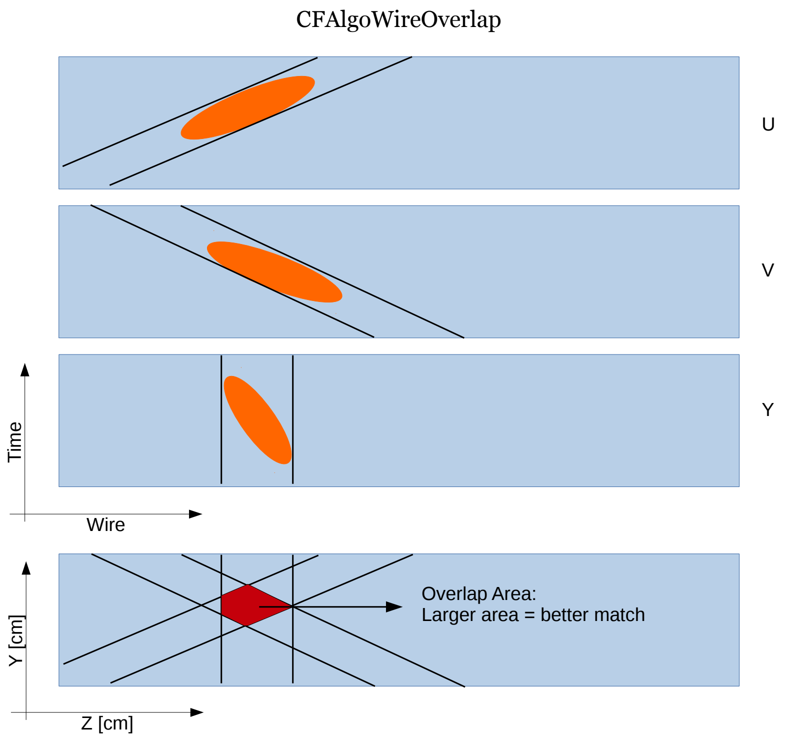

Figure 2: *Illustration highliting how CFAlgoWireOverlap uses overlapping wire-bands of clusters on different planes to calcualte a match score.*

2.3 CFAlgoVolumeOverlap

This algorithm finds a rough measure of the TPC volume overlapped by multiple clusters on different planes. A score proportional to the overlapping volume is returned as the match score. The overlapping volume is calculated as follows:
- An overlapping time window between all clusters is found. This is then converted in cm (the overlap in X)
- Each cluster's start and end wire are used to determine a band of wires defining the cluster's projection in (Y,Z) space. The intersection polygon common to all wire-bands determines the intersection area in (Y,Z) space.
- Finally, the intersection area in (Y,Z) is multiplied by the overlap region in X to get an overlap volume. This value is returned as the match score. Fig. 3 illustrates how this algorithm works.

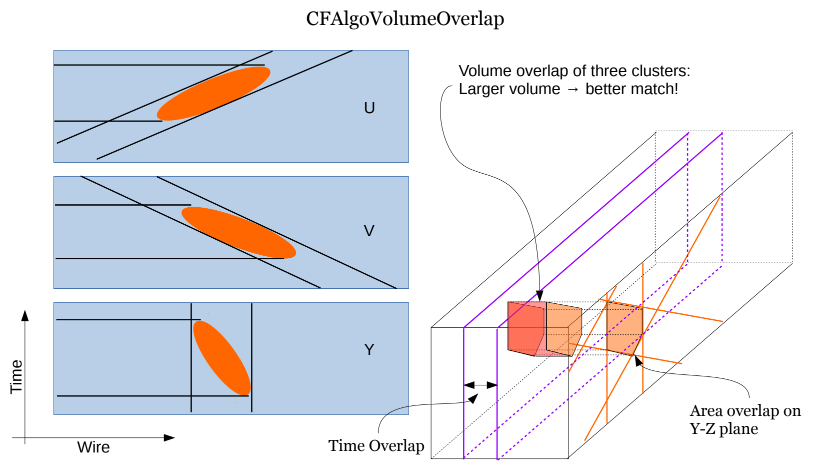

Figure 3: *Illustration highliting how CFAlgoVolumeOverlap uses the time and wire intersections across planes to calculate an overlapping volume common to all clusters.*

2.4 CFAlgoChargeDistrib

This algorithm looks at the charge distribution in time for multiple clusters. Clusters that truly belong to the same particle should see similar charge distributions in time. This algorithm computes a convolution (of sorts) of the time-distribution of the charge in clusters on different planes. The larger this convolution, the larger the score returned by the algorithm.
For each cluster, the distribution in time of the charge collected in the hits belonging to the clusters is scaled to a unit vector so that the charge from the hit with the smallest time is at 0 and that from the hit with the largest time is at 1. The charge distributions are then compared. The intersection of the two distributions is calculated. This intersection is a measure of the convolution of the charge distributions on the various clusters and it is used as the match score for this algorithm. Fig. 4 illustrates how this convolution is calculated.

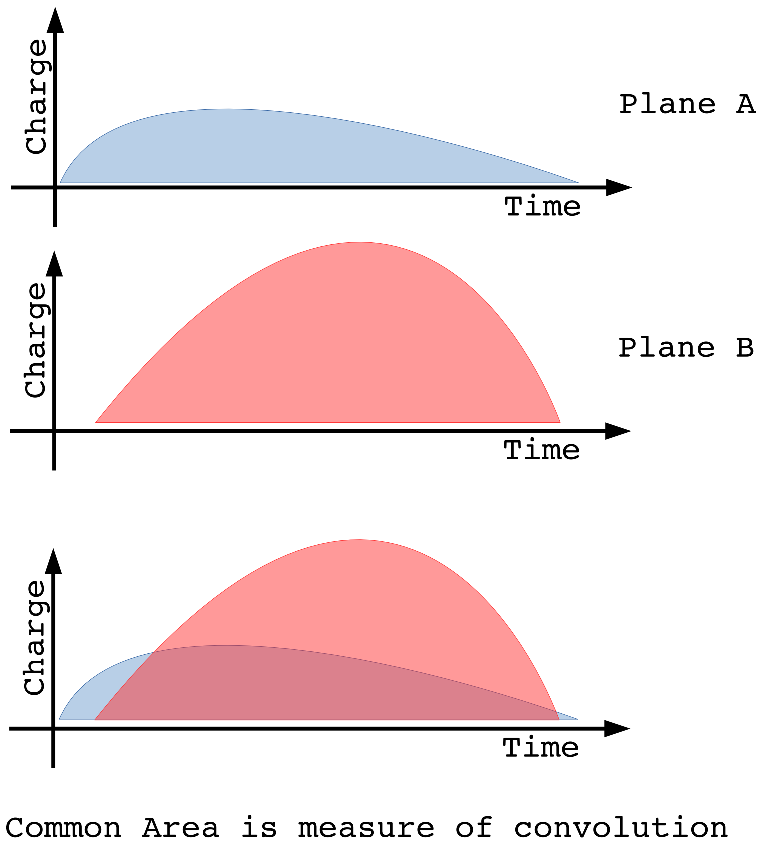

Figure 4: *Illustration highliting how CFAlgoChargeDistrib calculates a match score by comparing the time-distribution of charge on each cluster.*

2.5 CFAlgoQRatio

This algorithm compares the ratio of charge in the clusters considered. At first, the cluster with the largest charge is found. The charge for this cluster is denoted as Q_{max}. A charge ratio is found by calculating:

$$\sum_{i=0}^{\mathrm{N_{clusters}}} \frac{Q_i}{Q_{\mathrm{max}}} \tag{1}$$

If this ratio is larger than a certain cut value, the ratio is returned as the score. Otherwise, the return is -1.

2.6 CFAlgoStartPointCompat

This algorithm works only when clusters on all 3 planes are provided. For every pair of clusters, the intersection in (Y,Z) coordinates of the start-point wires on those clusters is found. This (Y,Z) point is then projected back to the plane of the 3rd cluster and the wire on the 3rd plane closest to this point is found. This is the reconstructed start wire on the 3rd plane. If this reconstructed start wire is in-between the start and end wires of the cluster on the third plane, the closer of the two to the reconstructed start wire is chosen. The inverse of the distance between these two is taken as the matching score. For a group of 3 clusters this calculation can be repeated for 3 different combinations. The highest score across all three combinations is finally returned by the algorithm. Fig. **??** illustrates how CFAlgoStartPointCompat works..

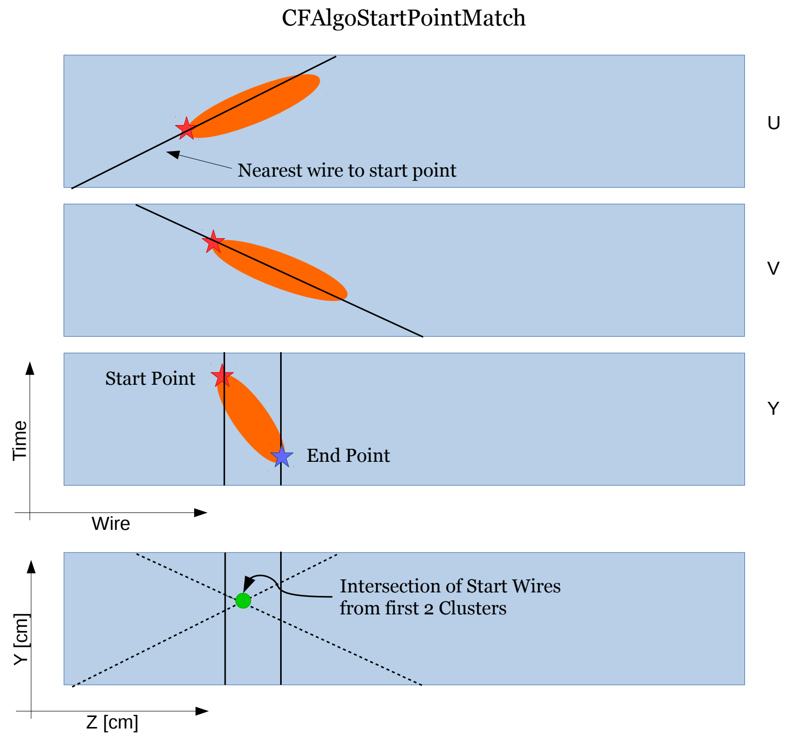
CFAlgoStartPointMatch
U
Nearest wire to start point
V
Start Point
End Point
Y
Time
Wire
Intersection of Start Wires from first 2 Clusters
Y [cm]
Z [cm]

Figure 5: *Illustration highliting how CFAlgoStartPointCompat calculates a match score by checking the compatibility of the start point reconstructed using two planes, with the cluster information on the 3rd plane..*

2.7 CFAlgoTimeOverlap

This algorithm calculates score based on the cluster overlap in time. It works for as few as 2 planes.

A vector of clusters (1 from each plane) is passed to the Algo; a score is returned for every pair and in the end the match with the best score wins the matching prize. For this algo, for every combination of clusters, the cluster with the largest timespan is used as the demoninator in the score ratio. In Figure 6, the correct cluster matches are [A, 1, a] and [B, 2, b], but all permutations are passed to the algo in order to determine this. One (incorrect) cluster combination passed to the Algo is [A, 2, a]. Notice here that it is not enough to take just time different into account, because the time differences between the 2 cluster pairs are too similar–we may end up with mismatched pairs. To handle this we add a requirement that the start points be within 10 cm of each other. We also handle the case of incorrectly reconstructed start and end points by making all the starting points the ones that occur earliest in time. Using this scheme, cluster combination [A, 2, a] will not get a better score than the correct match [A, 1, a].

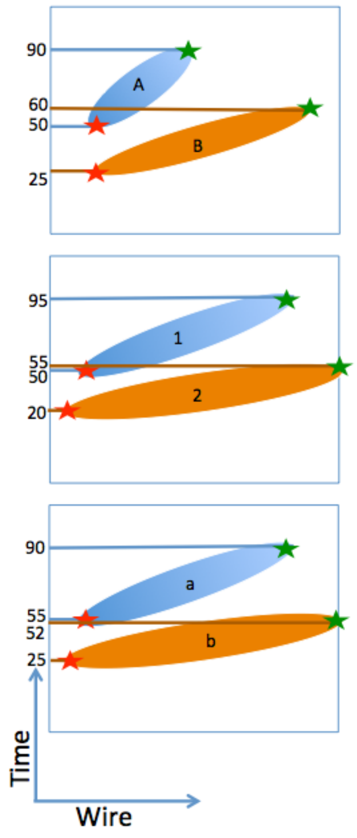

Figure 6: Illustration highlighting how CFAlgoTimeOverlap calculates score for cluster pairs.

2.8 CFAlgo3DAngle

This algorithm only works for 3 or more planes (or rather, for 3 or greater cluster pairs).

CFAlgo3DAngle calculates score based on similarity in theta and phi angles (Figure 7). Need 2 clusters in different planes to calculate one angle (theta or phi), so for each 3-cluster combination, we end up with 2 thetas, and 2 phis. We then compare these angles similarly to how we compared time. A caveat: very often one of the 3 planes will look more like a blob of charge than an object with direction (Figure 8); in this case, start and end points can be misassigned, and the angle comparison loses its effectiveness. To take this into account, we add in a charge weight that acts in conjunction with the angle comparison part of the algo. Examples of successful matches shown in Figure 9.

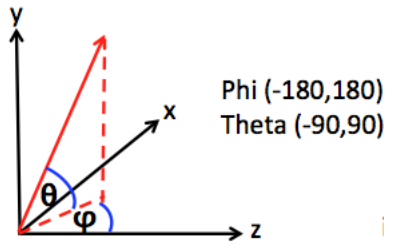

Figure 7: Angle definitions used in 3DAngle Algo

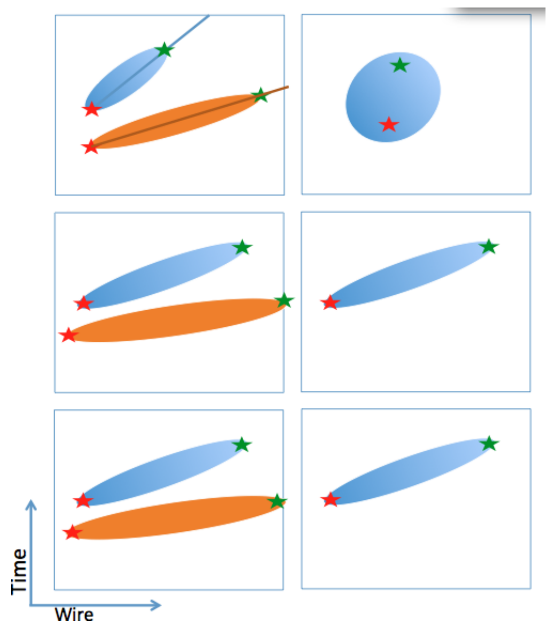

Figure 8: Illustration showing case where 3rd plane can be blobby and non-directional.

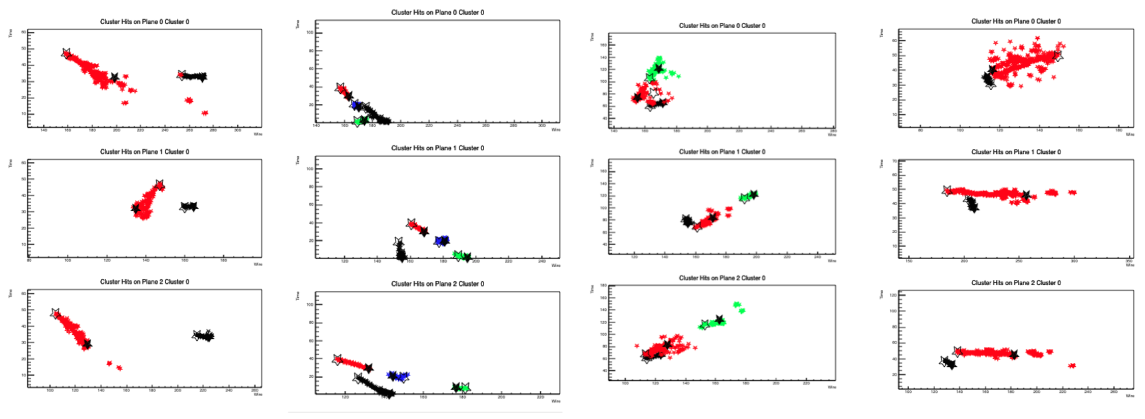

Figure 9: (Left two panels) Successful match examples using TimeOverlap Algo. (Right two panels) Successful match example using 3DAngle Algo.