The EMShower Reconstruction Technique

Michael Wallbank*[1]

[1]University of Sheffield, Sheffield, UK

Contents

*email: m.wallbank@sheffield.ac.uk

1 Introduction

This document describes the EMShower algorithm, written within the shared LAr software framework, LArSoft, and designed for shower reconstruction in LArTPCs [1]. It is written as an extension of the BlurredCluster 2D reconstruction algorithm [2][3] and uses these input clusters to form 3D shower objects.

The EMShower algorithm was developed for shower reconstruction in DUNE (specifically, initially, the 35-ton prototype) but is flexible and can be used by any LArTPC experiments (and has been in, e.g., LArIAT). It is intended to be very high-level and depends heavily on previous reconstruction, specifically 2D shower-like cluster reconstruction (provided by BlurredCluster) and 3D tracking reconstruction (provided by, e.g., ProjectionMatchingAlgorithm (PMA) [4]).

The art producer module which controls the running of the various algorithms lives in `larreco/ShowerReco/EMShower_module.cc` and is described in section 2. In the first instance, the algorithm simply matches the previously found, well formed, 2D clusters between the views to form 3D objects with associated hits in all views. It is written very generically such that there is no assumption whatsoever on the number of views present in the detector (to allow the same algorithm to be used in, for example, DUNE with three views and ArgoNEUT or LArIAT with two views). Once these hits are grouped to form 3D objects, various further algorithms are applied in order to analyse these hits and determine properties of the shower, such as start point, direction, energy and initial dE/dx. These algorithms are contained in `larreco/RecoAlg/EMShowerAlg.(cxx/h)` and discussed in detail in section 3.

Further algorithms written for use with the shower reconstruction are briefly mentioned in section 4 and a discussion of the current performance of the reconstruction is contained in section 5. A summary is provided in section 6.

2 Reconstruction Module

An art producer controls the running of the reconstruction and is configured as using within a .fcl steering file, using `@local_emshower`. This module reads in the user-defined parameters necessary to run and calls the individual algorithms in the relevant order.

2.1 User-Defined Parameters

The following are the parameters which describe the overall configuration of the reconstruction within art, along with their default values:

- `std::string HitsModuleLabel` ("linecluster") – the hit collection used by the BlurredCluster algorithm from which 2D clusters are formed. Needed to find the hit associations.
- `std::string ClusterModuleLabel` ("blurredcluster") – the 2D shower-like clusters previously found. Will almost always be BlurredCluster.
- `std::string TrackModuleLabel` ("pmtrack") – the 3D track reconstruction already performed. Necessary for the 3D shower matching.
- `art config EMShowerAlg` (`@local::standard_emshoweralg`) – standard configuration for the implementation of the EMShower reconstruction algorithm.

2.2 Standard EMShowerAlg Configuration

The `@local::standard_emshoweralg` configuration depends on other art configurations assumed to already be set up:

- `art config CalorimetryAlg` (`@local::standard_calorimetryalg`).
- `art config ShowerEnergyAlg` (`@local::standard_showerenergyalg`).
- `art config ProjectionMatchingAlg` (`@local::standard_projectionmatchingalg`).

3 EMShower Algorithm

The implementation of the EMShower algorithm is the subject of this present section; there are many separate algorithms contributing to the reconstruction which are each discussed. An overview of the methods used is provided in section 3.1 before the details of the various algorithms are presented. The user-defined parameters, and their default values, are listed in section 3.3.

3.1 Overview

The shower reconstruction proceeds in two general steps:

- Pattern recognition provides the geometrical shower shapes, formed by finding the associated hits in each of the views;
- Analysis of these hits across the views is carried out to determine the properties of these shower objects.

These output showers are then used to construct `recob::Shower` objects, which are placed into the event by the producer module. The methods used in each of the two general steps are outlines in section 3.2 below.

3.2 Algorithm Details

This section contains the details of the various algorithms used in the shower reconstruction. fhicl parameters read in at run time which can be altered by the user are highlighted in `typewriter` and listed for reference in section 3.3.

3.2.1 shower::EMShowerAlg::FindShowers

This function carries out the process described in the first bullet point in section 3.1; it takes as input the 2D shower-clusters (previously found by BlurredCluster) and 3D tracks and returns a vector of showers, where each shower is itself a vector of the 2D clusters of which it is comprised. The matching is done by simply using the hit associations between the 2D shower-clusters and the 3D tracks; by matching hits from two separate clusters to the same 3D track, the two clusters can be pulled together to form a 3D shower-like object. This is demonstrated in Fig. 3.1. The minimum length [in cm] required of a track in order for it to be used to perform this matching is set by the parameter `MinTrackLength`.

Once the algorithm returns the showers patterns to the producer module, the associations between showers and hits, clusters and tracks are trivial to find. With this information, finding the properties of the showers can proceed.

3.2.2 shower::EMShowerAlg::CheckShowerPlanes

This function is run after finding the initial showers in order to determine if there is a single bad view which is ruining the reconstruction. Since the reconstruction has been developed with π^0s in mind, an obvious example is when the showering decay photons overlap in a single plane but not the others. The nature of the 3D matching as described above would mean this event would be reconstructed in a single shower.

CheckShowerPlanes looks at the showers after they have been made to try to determine if this has occurred. It looks at the fraction of hits coming from the different clusters in each of the views and discards a particular view if this is suspiciously low in one view compared to the others. The clusters are then set to be ignored, and the **FindShowers** method run again.

3.2.3 shower::EMShowerAlg::FindInitialTrack

The next few algorithms are run to determine properties of the shower, specifically start point, direction, initial dE/dx and total energy. In determining the first three of these, it is imperative to reconstruct the initial part of the shower, which tends to be track-like (see for example the

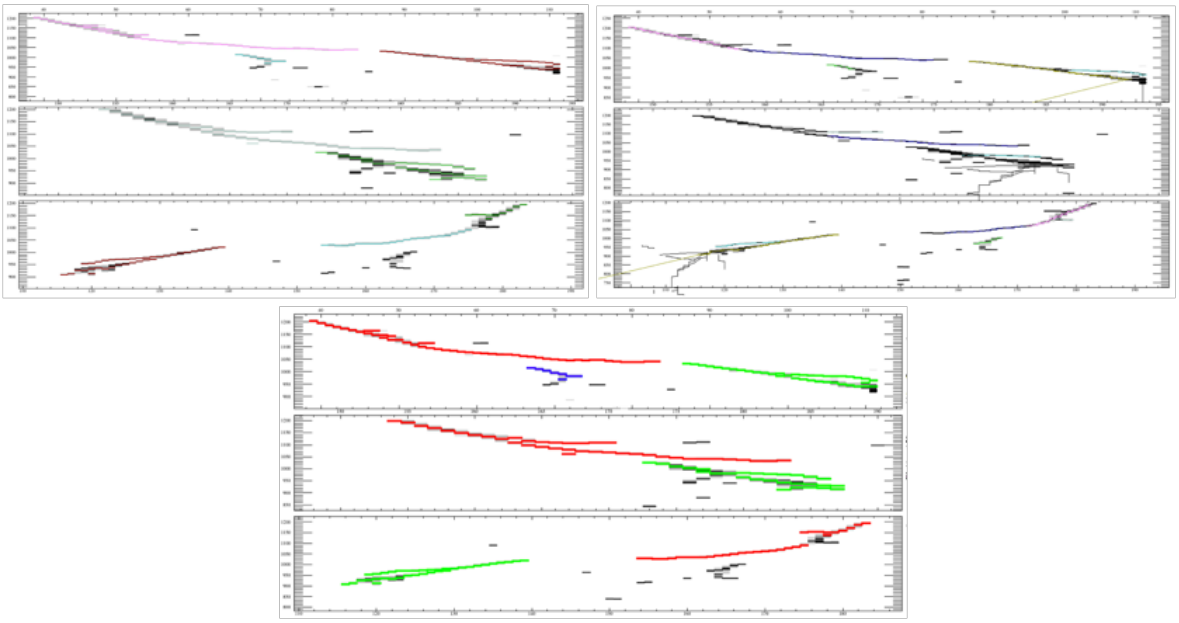


Figure 3.1: Demonstration of the EMShower shower-forming algorithm. The event displays (as usual, wire vs tick for each of the three planes separately) show a 35-ton π^0 particle event, with the two decay photons visible as separate showers. The top shows the event reconstructed separately by BlurredCluster (left) and PMTrack (right) and the bottom shows the final reconstruction with the EMShowers visible. In each case, the hits are highlighted with different colours to demonstrate separate reconstructed objects. It can be seen the BlurredClusters are all different colours across the views, demonstrating the 2D nature of this output, whereas the PMTracks have associated hits between all the views as this is a 3D reconstruction method. The final EMShowers (bottom) are formed by associating the hits from the BlurredClusters and PMTracks, thereby matching clusters between the views and making 3D shower-like objects. This has the effect of combining the well-formed nature of the BlurredClusters (evidenced by the above event display showing complete clusters whilst also maintaining a nice separation between the two decays photon showers) with the accurate tracking nature of PMTrack.

showers in Fig. 3.1). The EMShowerAlg algorithm actually finds this track in all views and constructs a `recob::Track` object within the code for straight forward use. The **FindInitialTrack** method takes as input the all the hits, across all views, associated with a shower and returns a `std::unique_ptr<recob::Track>` to represent the initial part of the shower.

There are three steps involved in the construction of this track, each discussed in detail below:

- order the hits correctly in each view so that their order follows the shower trajectory;
- from this correctly oriented shower, find the hits which form the initial track-like part of the shower in each view;
- use these hits to make a 3D track.

shower::EMShowerAlg::OrderShowerHits Firstly, the hits need to placed in an order and oriented such that they correctly describe the propagation of the showering particle. This in turn follows three general steps:

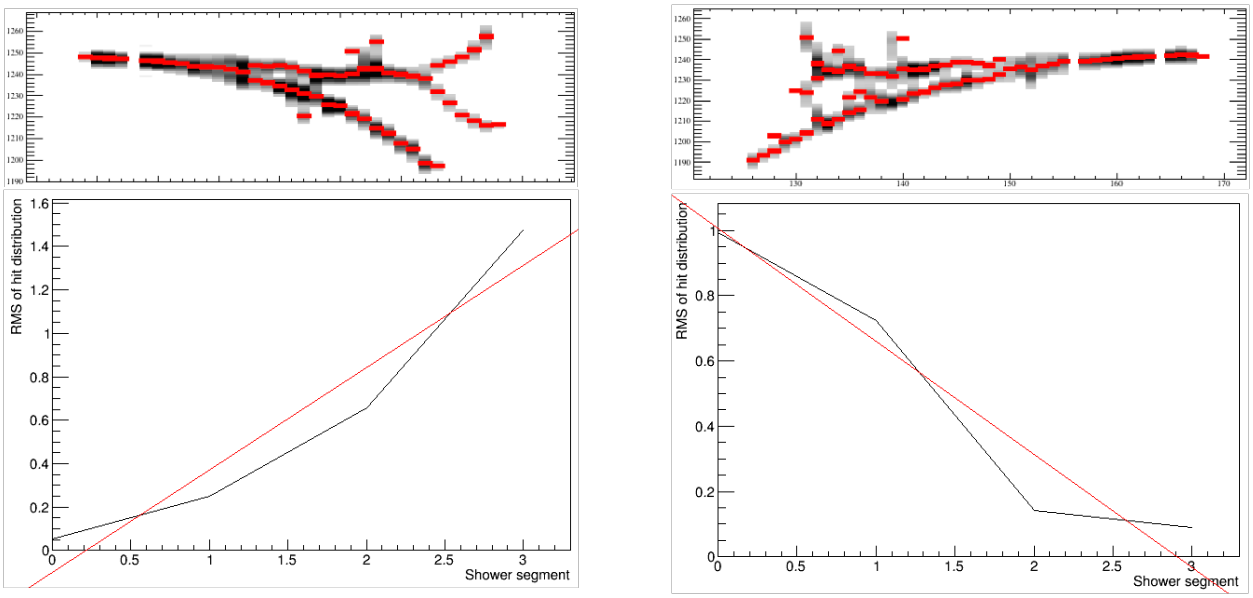
- first ordering the hits correctly, not necessarily with the correct orientation;
- check this ordering using hits from all planes;
- use the properties of the shower in each view to orient the hits correctly.

Each is detailed further below.

Order Hits The hits are ordered very generally by fitting an axis to each set of 2D shower hits in turn and using the projection of each of the hits back onto this line to order them correctly along the length of the shower. The axis used is simply the line of best fit which passes through the charge-weighted centre of the hits.

Check Order of Hits This stage is designed to determine if there is a particular view in which the shower isn't very well formed (for example, if the shower travels parallel or almost-parallel to the wires in one plane, leading to poor resolution of the shower features. This is found by comparing the RMS of the perpendicular distance of the hits from the central axis in each of the views and discarding any in which this is much smaller. This step has a big impact on the success of determining the 3D direction of the shower.

Orient the Hits Once there is sufficient confidence that the hits are correctly ordered in the remaining planes, they are then oriented correctly. This is done by evaluating the 'RMS gradient', the RMS of the distribution of the hits in discrete sections along the length of the shower. If a shower were incorrectly oriented, one would expect a negative gradient, and vice versa for a correctly oriented shower. This is demonstrated in Fig. 3.2.



(a) Positive gradient

(b) Negative gradient

Figure 3.2: Demonstration of the method used to orient the shower correctly in each view. The figures show the 'RMS gradient', illustrating how the RMS of the hit distributions change as each subsequent segment of shower is considered. The shower is broken into discrete sections and for each the RMS of the distances of each hit to the central axis is determined. How this changes represents the development of the shower. In Fig. (a), the shower hits are already oriented correctly and so the gradient is positive and nothing need change. However, in Fig. (b), a negative gradient, which implies a change in the order of hits is required, is evident. This method works remarkably well, even with showers that don't have an obvious development to the eye.

shower::EMShowerAlg::FindShowerStart After correctly orienting the hits along the axis of the track in each view separately, it is straight forward to determine the initial track-like parts. For each plane, the hits are considered in order until it becomes obvious that the shower is diverging (i.e. multiplicity of hits on a given wire). At this point, the track is considered complete and these starting hits are returned from the function.

shower::EMShowerAlg::MakeInitialTrack The **MakeInitialTrack** method accepts a map of initial track-hits hits in each plane, determined by **FindShowerStart**, and returns the `std::unique_ptr<recob::Track>` which is, in turn, returned from **FindInitialTrack**. It finds the two best views, should there be more than one, and passes the initial hits from these two views to the function **ConstructTrack**, which makes the `recob::Track` object.

shower::EMShowerAlg::ConstructTrack This method is incredibly generic and is designed to be used by any LArSoft algorithm that requires a track to be constructed from two sets of hits. It is therefore provided as a public method for this reason. It does in turn rely on

a PMA method, `pma::ProjectionMatchingAlgorithm::buildSegment()` but also provides further checks and reconstruction. By default, the method assumes the hits passed to it are ordered, so forces the start of the track to be at the point of intersection of the two initial hits.

3.2.4 shower::EMShowerAlg::MakeShower

The final main method called by `EMShower_module.cc` is **MakeShower**. This function accepts all the shower hits in each view and the initial track, along with associated hits, as previously determined by **FindInitialTrack**. It returns a `recob::Shower` object with all the properties set correctly, ready to be placed into the event record.

The start point and direction are taken directly from the `recob::Track` object which it is passed. The dE/dx for the initial part of the track is calculated by **FinddEdx**, given the initial track hits. The parameter `dEdxTrackLength` sets the maximum length of track [in cm] this is determined for. Finally, the total shower energy is determined by the method **shower::ShowerEnergyAlg::ShowerEnergy()**, which will be discussed in section 4. These two energy reconstructions, total energy and dE/dx, are saved for each plane separately. The `recob::Shower` data product however includes a `best_plane` parameter, which sets the view decided to be the most accurate for taking information from. This plane is set to be the one with the greatest number of hits associated with the shower on.

3.3 User-Defined Parameters

Here, all the parameters available to the user to tune the showering are listed for reference, along with their default values.

- `MinTrackLength` (3) – the minimum length [in cm] required of a track in order for it to be used to match BlurredClusters between views.
- `dEdxTrackLength` (3) – the maximum track length [in cm] to use in the calculation of the dE/dx of the initial part of the shower.

4 Further Algorithms

In the development of the EMShower algorithm, other algorithms have been added to LArSoft in the directory `larreco/RecoAlg/` for general use. These are briefly overviewed here.

4.1 shower::ShowerEnergyAlg

The `ShowerEnergyAlg.(cxx/h)` class contains methods used to determine the total energy of the showers. Each detector must be considered separately, for MC and for data. When configured correctly, **shower::ShowerEnergyAlg::ShowerEnergy()** will return the total shower

energy given a set of hits. It does this by utilising the linear relationship between the total deposited charge and the deposited energy to make the conversion. Such a relationship, using true MC for 35-ton particle electrons, is shown in Fig. 4.1.

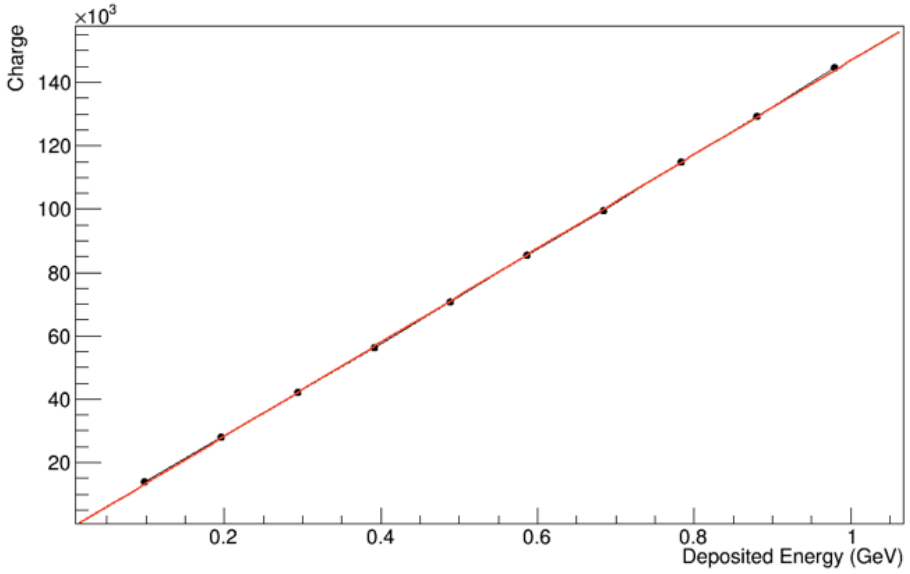


Figure 4.1: Demonstration of the striking linear relationship between true deposited energy and reconstructed charge in the TPC. The sample used is a 35-ton particle gun MC electron sample, generated at ten monochromatic energies.

This relationship must be determined separately for each detector. The necessary parameters are stored in `larreco/RecoAlg/showeralgorithms.fcl` for the algorithm to read in. For MC, it is possible to consider the true deposited energy (by looking at the `sim::IDE`s associated with each particle); for data this must be calibrated by using through-going MIPs (once the distance travelled and the charge deposited is known, the relationship follows).

5 Reconstruction Performance

This section contains the reconstruction performance of the EMShower algorithm. The following samples were used to provide validation of the reconstruction, allowing various properties of the showers and also a reconstructed π^0 mass peak to be analysed:

- DUNE 10kt MC electron, $0.1 - 5$ GeV, 10000 events
- DUNE 10kt MC photon, $0.1 - 2$ GeV, 10000 events
- DUNE 35t MC π^0, $0.4 - 1$ GeV, 10000 events

5.1 Reconstruction Efficiency

An interesting metric is to consider is how often a showering particle has an associated reconstructed object. This is demonstrated for both electrons and photons in Fig. 5.1.

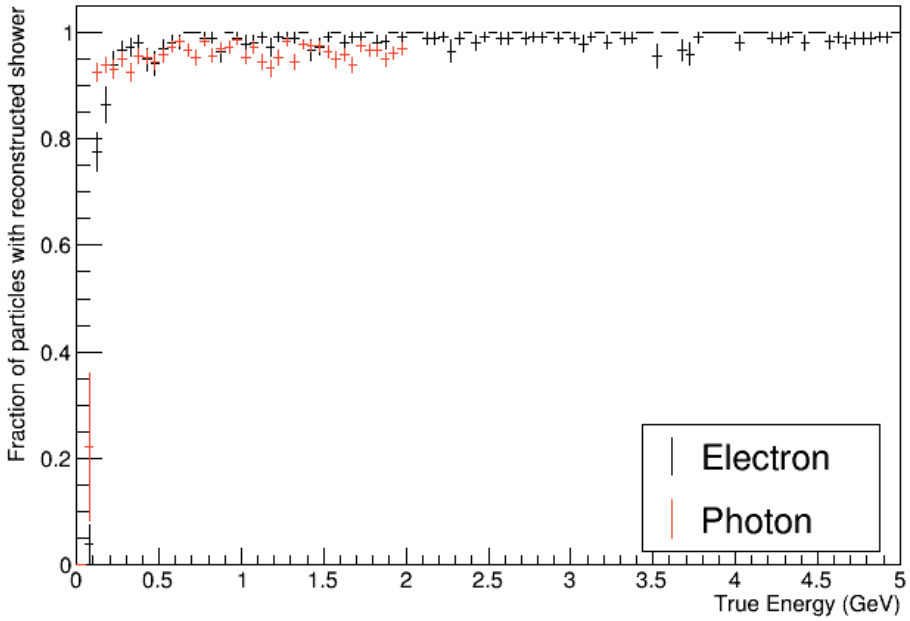
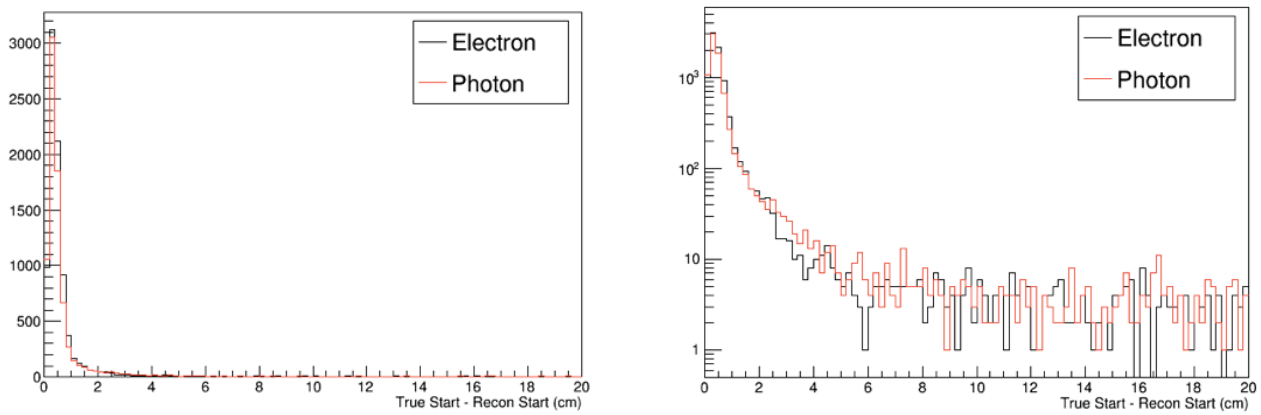


Figure 5.1: The fraction of showering particles which have an associated complete reconstructed shower object, as a function of true energy. A reconstruction shower is considered complete if it has all associated properties determined and associated hits.

5.2 Shower Properties

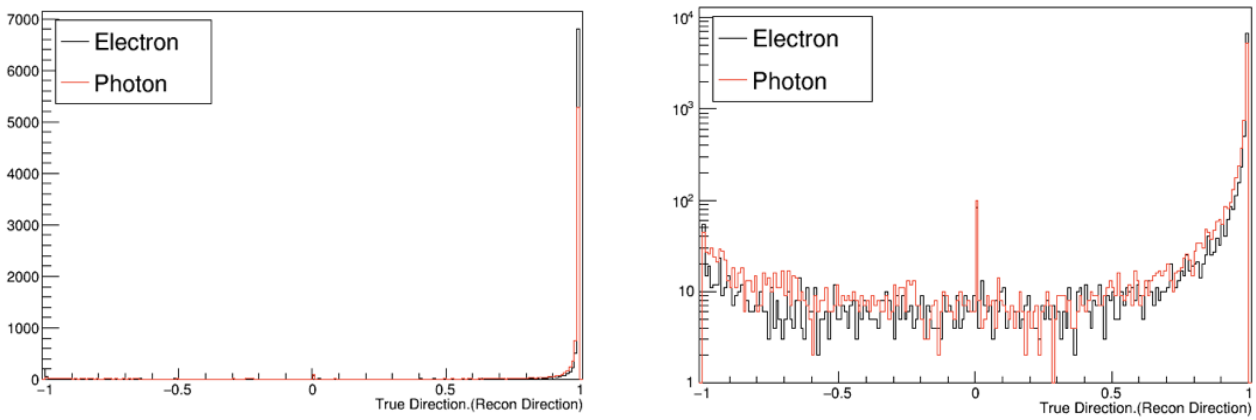
Various properties of the showers are presented here for both electrons and photons in the DUNE far detector. The start point (Fig. 5.2), direction (Fig. 5.3), dE/dx (Fig. 5.4) and total energy (Fig. 5.5) are all shown below. In general the properties look excellent and demonstrate well the good performance of the shower reconstruction provided by EMShower.



(a) Normal scale

(b) Log scale

Figure 5.2: Difference in true and reconstructed start positions for showering electrons and photons.



(a) Normal scale

(b) Log scale

Figure 5.3: Dot product of true and reconstruction directions for showering electrons and photons.

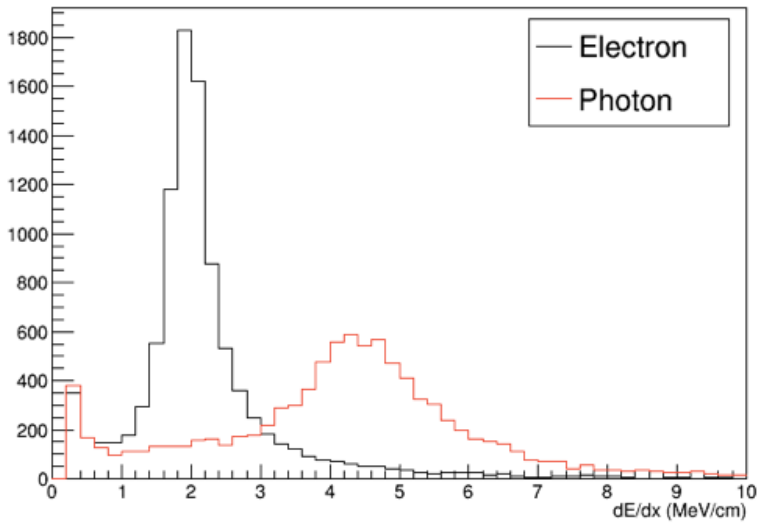


Figure 5.4: dE/dx of the initial part of the reconstructed shower for electrons and photons. A nice separation between the distributions for both particles can be seen.

5.3 π^0 Reconstruction

The original motivation for developing EMShower was to successfully reconstruct π^0s in LAr. It is therefore instructive to consider the reconstructed mass peak, shown in Fig. 5.6. This histogram is filled with only reconstructed quantities, but uses truth information to select the largest reconstructed shower for each true photon shower. It can be seen this looks very reasonable; the slightly low peak is due to a small amount of fragmentation leading to not all associated hits being reconstructed in the same shower (an issue also seen in Fig. 5.5) and the width of the peak is due to the shower direction reconstruction not being perfect.

However, in general, this looks good and validates the shower reconstruction provided by EMShower.

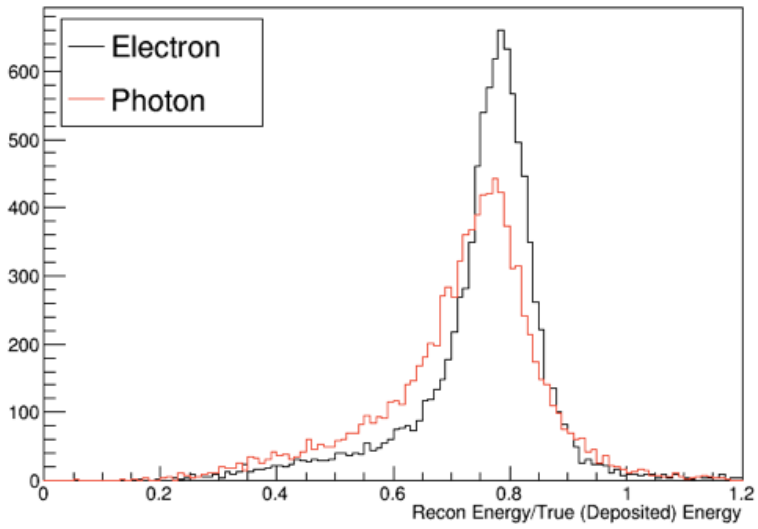


Figure 5.5: Fraction of true energy which is reconstructed in the shower object for showering electrons and photons.

6 Summary

The EMShower algorithm within LArSoft for shower reconstruction in LAr has been presented and discussed in this document. Specific algorithm details have been described and the configuration, within LArSoft, has been outlined.

As seen from the validation plots in section 5.2, the performance of the reconstruction is good and produces high-quality showers. Development is still ongoing so improvements will be made when necessary. This document will be kept up-to-date to reflect any such changes.

The method can easily be applied to further experiments which use LArSoft with minor adjustments; hopefully this document and the equivalent one for BlurredCluster [3] prove useful. The author is happy to help if necessary.

References

[1] http://larsoft.org/single-record/?pdb=113
[2] http://larsoft.org/single-record/?pdb=110
[3] http://docs.dunescience.org:8080/cgi-bin/ShowDocument?docid=54
[4] http://larsoft.org/single-record/?pdb=102

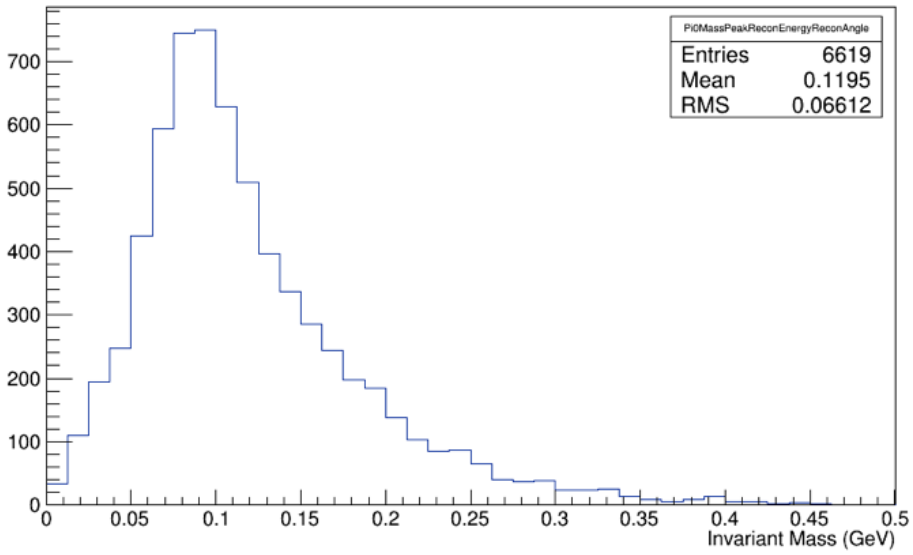


Figure 5.6: The reconstructed mass peak for a 35-ton π^0 sample using reconstructed showers provided by EMShower. See the text for a more detailed description.