Measuring charged current neutrino interactions in the Electromagnetic Calorimeters in near detector of T2K

Dominic Brailsford
TODO

A thesis submitted to Imperial College London
for the degree of Doctor of Philosophy

Abstract

To write

Declaration

This dissertation is the result of my own work, except where explicit reference is made to the work of others, and has not been submitted for another qualification to this or any other university. This dissertation does not exceed the word limit for the respective Degree Committee.

Dominic Brailsford

Acknowledgements

Something about my supervisor ...

Preface

This thesis describes my analysis of the ν_μ charged-current cross-section on lead using the T2K near detector electromagnetic calorimeters.

Contents

"These chickens jackin' my style. "

— Fergie

Chapter 1

Introduction

"I've got that boom boom pow"
— Fergie

Introduce neutrinos here [1–3].

1.1 Neutrino Oscillations

Neutrino oscillations are really kool

Chapter 2

The T2K Experiment

The Tokai-to-Kamioka (T2K) experiment [4] is a long baseline neutrino oscillation experiment located in two sites across Japan which was designed to study the parameters governing the PMNS matrix. The first site is the J-PARC facility in Tokai-mura on Japan's east cost which houses a 30 GeV proton accelerator complex that is used to generate a highly pure ν_μ beam. J-PARC also contains a suite of detectors designed to measure the neutrino beam's unoscillated characteristics. Super-Kamiokande (SK) is located 295 km (see Fig. 2.1) and measures the contents of the neutrino beam post-oscillation.

T2K was the first experiment to observe the $\nu_\mu \rightarrow \nu_e$ appearance channel [5] which excluded $\theta_{13} = 0$ at 7.3σ significance. By comparing this result with precise θ_{13} measurements from reactor experiments, δ_{CP} regions can be excluded at 90% confidence level (see Fig. 2.2). T2K's precision analysis of the ν_μ disappearance channel provide world leading measurements of θ_{23} and Δm^2_{23}. Independently of the oscillation analyses performed by the experiment, T2K's near detectors, ND280 and INGRID, are used to measure a range of neutrino cross-sections [6,7]. While this is not the primary aim of T2K, such measurements are still extremely important as T2K systematic uncertainties can be constrained with additional cross-section knowledge as well as helping to understand the general neutrino interaction picture.

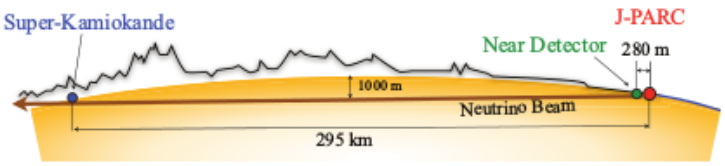

Figure 2.1: The T2K experiment.

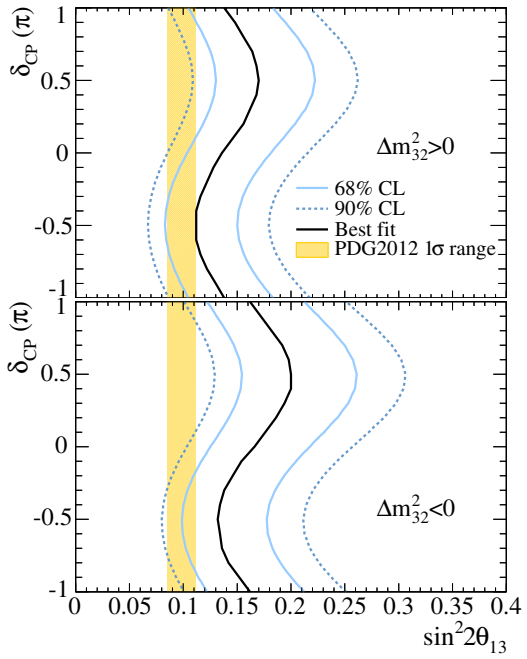

Figure 2.2: The 68% and 90% confidence level allowed regions for $\sin^2 2\theta_{13}$ as a function of δ_{CP} for normal hierachy (top) and inverted hierarchy (bottom) The solid line represents the best fit $\sin^2 2\theta_{13}$ for a given δ_{CP}. The shaded region shows the average θ_{13} provided by the reactor constraint [5].

2.1 T2K beam

The T2K neutrino beam is generated by J-PARC's accelerator complex which produces a 30 GeV proton beam which is fired at a fixed graphite target. The final states particles of interactions with the target are predominately charged pions which decay to produce the neutrino beam. Surrounding and behind the graphite target are a set of magnetic horns which focus the pions into a beam, resulting in a focused neutrino beam after the hadrons have decayed.

2.1.1 Accelerator complex

The J-PARC acclerator complex consists of three sections: the LINnear ACcelerator (LINAC), the Rapid-Cycling Synchotron (RCS) and the Main Ring synchotron (MR). Production of the proton beam starts at the LINAC where H^- anions are accelerated to 181 MeV which are subsequently converted to h^+ ions via charge-stripping foils at the RCS injection point. With a 25 Hz cycle, the ions are further accelerated by the RCS

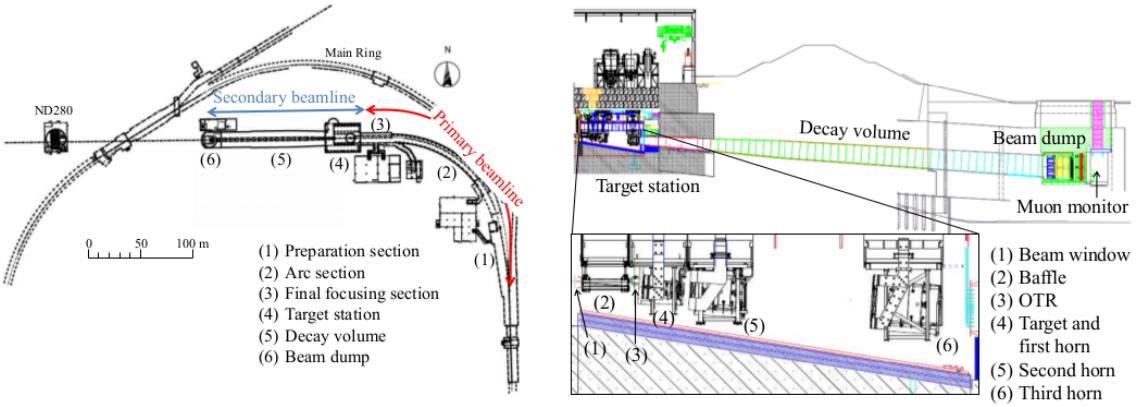

Figure 2.3: A schematic of the T2K neutrino beamline (left) and a side view of the secondary beamline (right).

to 3 GeV with two bunches per cycle. Roughly 5% of the proton bunches are fed into the MR where the final acceleration to 30 GeV occurs in bunches of eight. Extraction of the bunches occurs at two points for different experiments. For T2K, all eight bunches are extracted in a single turn by five kicker magnets and aimed down the neutrino beamline to the graphite target. The extraction of all eight bunches forms a single beam spill with a width of 5 μsec. The tight structure of the beam spills is vital for background discrimination in the downstream detectors.

2.1.2 Neutrino beamline

The neutrino beamline (NU) is split into a primary and secondary beamline, a schematic of which is shown in Fig. 2.3. The primary beamline consists of a preparation section, an arc section and a focussing section. The preparation section uses 11 normal conducting magnets to tune the proton beam for entry into the arc section where the proton beam is bent to its intended direction. As will be discussed in more detail in section 2.1.3, the axis of the beam is 2.5° away from SK. The final focussing sections then guides the proton beam into the secondary beamline and the graphite target.
Sound performance of the proton beam is vital for stability of the T2K neutrino beam. To ensure such performance, the primary beamline is equipped with a suite of monitors to measure the position, profile, loss and intensity of the proton beam. The beam position is measured by 21 electrostatic monitors (ESMs) which consists of four cylindrical electrodes surrounding the beam. The asymmetry of the beam is measured by the induced current in the electrodes is used to infer the position in a non-destructive

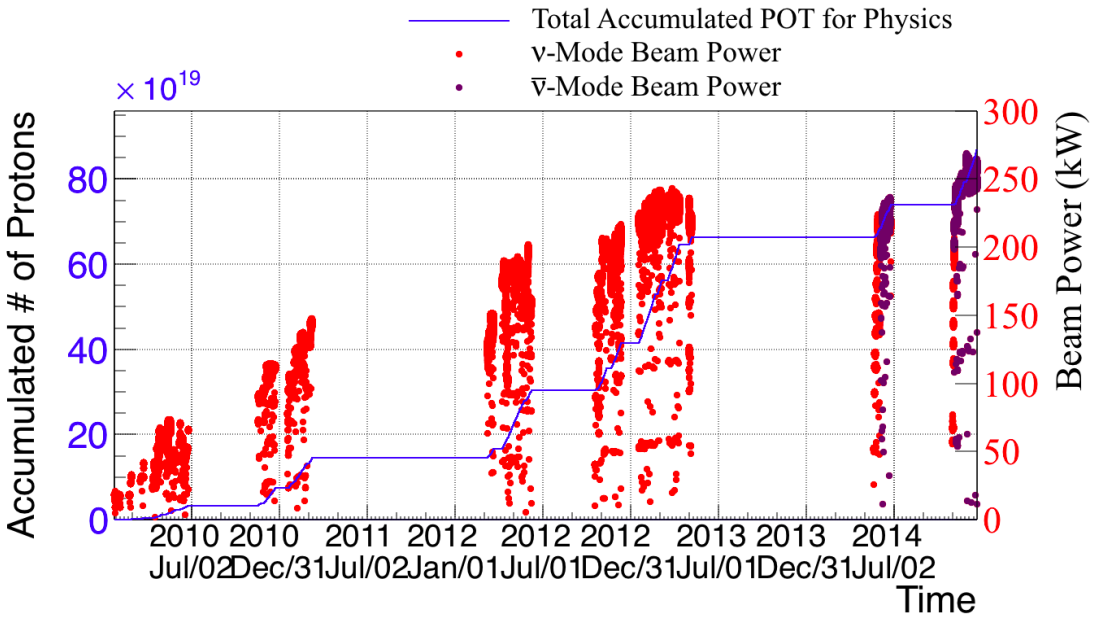

Figure 2.4: The POT recorded by CT5 as a function of time (blue line) and the recorded beam power in ν running mode (red dot) and $\bar{\nu}$ running mode (purple dot).

manner. Segmented secondary emission monitors (SSEMs) are used to measure the beam loss. Each SSEM has an anode foil sandwiched between two titanium foil strips. Protons interact with the strips causing an emission of electrons which electrically drift inducing a current in the strips. The charge distribution is used to reconstruct the profile. The beam loss monitors (BLMs) are Ar-CO_2 filled wire proportional counters and are used to for beam loss. The intensity of the beam is measured by five current transformers (CT) which consist of a 50-turn toroidal coil around a ferromagnetic coil. Passage of the beam induces a current in the coil which is used to infer the number of protons in the spill. The final CT (CT5) is positioned at the end of the focussing section of the primary beamline and is used to count the number of protons incident on the graphite target. The accumulated number of protons on target (POT) is used as a metric for the data collected by T2K. The total POT accumulated so far by T2K is shown in Fig. 2.4.

The secondary beamline consists of the graphite target, a set of magnetic focusing horns, a decay pipe and a beam dump. A schematic for the secondary beamline is shown in Fig. 2.3. The graphite target is a 2.6 cm diameter and 91.4 cm long rod which is surrounded by a 2 mm thick graphite tube and a 0.3 mm titanium case. The proton-graphite interactions produce charged pions and kaons which are focussed by three magnetic horns, one of which surrounds the target. The magnetic horns consist of two coaxial conductors which generate a magnetic field with a strength inversely

proportional to the distance from the beam axis. The current direction in the magnetic horns causes the induced field to focus or deflect particles depending on their charge sign. This simple control allows T2K to operatre in ν or $\bar{\nu}$ beam mode. The focused mesons then travel down a decay pipe filled with Helium to reduce pion absorption. It is here that the mesons decay to produce the neutrinos used as the T2K beam. To stop measurement contamination, other decay products must be stopped before reaching the downstream detectors. So, a 75 ton graphite beam dump is positioned at the end of the decay volume. The beam dump stops almost all non-wanted decay products, with only 5 GeV or above muons successfully propagating through. As the muons are generally simultaneously produced with the beam neutrinos, measurements of the muon can be used to monitor the direction of the neutrino beam. To do this, a muon monitor (MUMON) is installed at the downstream end of the beam dump.

2.1.3 Off-axis beam

The kinematics of the pion decays dictate the energy spectrum shape of the neutrinos. Specifically, the peak width of the neutrino energy narrows and shifts as an observer moves off-axis from the pions trajectory. As the pions are the neutrino parents in the T2K beam, the same effect can be seen by moving off-axis from the neutrino beam. This effect is illustrated in Fig.BLAH. By positioning T2K's baseline detectors at 2.5° off-axis, it is possible to align the neutrino beam's peak energy with the first oscillation maximum for the ν_μ disappearance channel. Separately, an off-axis configuration reduces the beam's unwanted high energy tail, improving sensitivity to ν_e appearance and ν_μ disappearance.

2.2 Near detector complex

Stuff about comples

2.2.1 Multi-Pixel Photon Counter

MPPCs

2.2.2 INGRID

Stuff about INGRID

2.2.3 ND280

ND280 time

2.2.3.1 The fine grain detectors

FGDS

2.2.3.2 The time projection chambers

TPCs

2.2.3.3 The π^0 detector

Here is ref for 2.2.3.2 pi0

2.2.3.4 The electromagnetic calorimeters

The beasts

2.2.3.5 The side muon range detector

Why bother?

2.2.4 The far detector

I mean SK

Chapter 3

ND280 software and existing ECal event reconstruction

The T2K experiment uses a bespoke software suite for simulation and analysis of ND280 data which is based on the ROOT framework [8]. The vast majority of the ND280 software suite utilies the oaEvent library which provides a unified framework for information manipulation and was specifically designed for this purpose. As ND280 consists of many subdetectors each providing a specific function, the ND280 software suite is designed to reflect this. Not only are there specific software modules for individual subdetectors, there are specific modules for each phase of the subdetector information processing e.g. trip-T calibration, TPC reconstruction etc.
As the software suite handles both production of simulated data and the processing of collected data, there are sections of the software chain which are specific to type of data being processed. While the Monte Carlo simulation and real data do see different areas of the software chain, the general philosophy is to maniuplate the Monte Carlo or the real data to a point where they can be treated as equals and them process them in the same manner. So, the description of the software will follow the same path: the Monte Carlo and real data specifics will be discussed first and then the unified treatment will follow.

3.1 Monte Carlo production software

As described above, parts of the software chain are unique to simulated data processing. Specifically, the simulation of the beam and the detector response need to be modelled

before the Monte Carlo can be treated on equal footing with the real data. This special processing is split into several steps, all of which are described below.

3.1.1 Neutrino flux simulation

The neutrino flux simulation uses Fluka2011 [9] and a software set called JNUBEAM to model the J-PARC neutrino beam. The process begins by using Fluka2011 to simulate the 30 GeV protons incident on the graphite target and their subsequent secondary interactions which produce the neutrino parent mesons. The kinematic information of the hadrons is the passed to the JNUBEAM simulation. JNUBEAM is based on GEANT3 [10] and models the J-PARC secondary beamline. The hadrons are tracked through the decay volume and are allowed to interact or decay to produce the simulated neutrino beam. Importantly, all information associated with the daughter neutrinos and their parents are saved at this point. By storing this information, the neutrino flux can be readily re-weighted to include new information associated with beam profile measurements or external data.
The main external tuning source is NA61/SHINE which is a hadron interaction experiment that uses a 31 GeV/c proton beam colliding with changable targets [11]. For use in the T2K flux simulation, NA61/SHINE has collected data using two graphite targets: one with a 4% nuclear interaction length thickness and a full T2K replica target. The flux simulation used for this analysis is tuned using full replica target data. Observed differences between the fluka2011 simulation and NA61/SHINE data are used to re-weight the simulated neutrino flux.
Additionally to the external data tunings measurements of the T2K beam profile are also used to re-weight the flux. By making such measurements on a run-by-run basis, the simulated flux can be re-weighted to better model variations of the neutrino beam in each data run.

3.1.2 Neutrino interaction simulation

After the neutrino flux has been modelled, simulation of the neutrino interactions with the T2K detectors follows. The NEUT [12] event generator is used to simulate interactions with ND280. The inputs to the interaction simulation are a neutrino vector file produced by the beam simulation and a ROOT based ND280 geometry. The used geometry includes the magnetic field return yoke and everything contained

within. Using the inputs, NEUT tracks the neutrino and calculates the probability of interaction for every material in crosses. To calculate the interation probability, the potential interaction nucleus must be modelled. For this, NEUT uses two models; the Moniz-Smith Relativistic Fermi Gas (RFG) [13] and the O. Benhar spectral function models [14]. The spectral functions are only implemented for carbon, oxygen and iron so the model used depends on the atomic number of the interaction candidate nucleus. It is important to note at this point that this thesis deals with neutrino interactions on lead, so it is the RFG model that is used for signal interactions.
The main interactions modes at T2K energies are quasi-elastic scattering (CCQE), single pion production (CC1π) and deep inelastic scattering (DIS) all of which have models in NEUT [15–17].
After the initial interactions, the final step is to simulate the final state interactions within the nucleus. Each particle involved with the interaction is pushed through the nucleus in discreet steps with the probability of a final state interaction being calculated at each step. If an interaction occurs, the final states of that interaction are also included in the subsequent steps. This interative procedure models the particle cascade until all the final states have reached the nucleus boundary. At this point, all final state particles are recorded along with all information that created those particles. This information is stored in a vector file and passed onto the ND280 detector MC package which handles the detector's response to these final state particles.

3.1.3 ND280 detector simulation

The simulation of the final state particles in ND280 is handled by nd280mc which is based on Geant4 [18] and ROOT. The neutrino interaction vector files are taken as input and used as seeds in the detector simulation. The neutrino vector inputs are not organised according to the J-PARC beam bunch structure so the detector simulation first groups the interactions into spills. The beam intensity being simulated is used to define how many interactions occur in a spill with Poisson fluctuations applied to that number. The timing of the beam bunch structure is then used to group the interactions into bunches.
nd280mc constructs a ROOT geometry of ND280 based on the design specifications of its subdetectors and then propagates the particles given to it by the neutrino generator through the geometry, simulating energy deposition, scattering and particle decay during propagation.

3.1.4 Detector response simulation

The next and final stage of the MC-only software chain is to model how the detector responds to the simulated particles propagating through it. The detector response software, named elecSim, takes the output of nd280mc and models how the active regions of the detector would respond given an energy deposition in that region of the detector. In the case of the calorimeters, elecSim handles the production of light produced by the constituent scintillator bars, how the light is propagated along the wavelength-shifting fibres and how the MPPCs would respond to the incident photons. For the TPCs, the drift of the ionisation electrons through the gas and the subsequent response of the MicroMEGAS which recieve them. In all cases, the readout electronics response is simulated to produce a data-like output format. This step concludes the section of the software chain which is specific to the MC.

3.2 Real data processing software

The real data specific section of the software chain is very short. Physics events deemed worth saving by any of the ND280 triggers are recorded by the detector and the saved for processing. The MIDAS file format is used for storing the saved ND280 events. All of the relevant information needed to process the event is stored in the MIDAS file, so the only unique step to the data processing is the conversion of the MIDAS file to the oaEvent format. After the step, the oaEvent data files are exposed to the same software as MC files outputted by elecSim (see section 3.1.4).

3.3 Main software chain

The aim of the rest of the software chain is process the readout from the detector, be it simulated or collected signal, and process it so essential information about the physics of the event can be extracted. This is separated into three steps: calibration, reconstruction and data reduction/summarising.

3.3.1 Detector calibration

The software package responsible for overseeing all aspects of the calibration stage is called oaCalib. This controlling package passes the digitised signal from ND280 to dedicated calibration packages for the various kinds of readout electronics. All of the information which can be extracted from an ECal event relies on the charge read by the MPPCs. Thus, for the Trip-T detectors, like the ECals, the main aim of the calibration is to remove all electronic effects so that an accurate estimation of the charge read by the MPPCs can be made.
Firstly, every MPPC is held at a bias in the absense of any signal to remove the non-linear dependence of the detector response on the lowest ADC channels. This zero signal output is called the pedestal and must be subtracted from the charge readout by the MPPC.
To estimate the number of photo-electrons produced in an MPPC, The readout charge is divided by the MPPC gain. The breakdown voltage of an MPPC linearly varies with temperature (approximately 50 mV/°C). This translates to a percent level variation in the MPPC gain per degree. This means small variations in temperature can cause a significant variation in the MPPC gain. So, the gains for each channel are calculated and stored individually.
The time of the signal read by the MPPC is charge dependent so it relies on the above corrections. However, the are extra steps needed to get an accurate estimate of the timestamp. The MPPC signal is given a timestamp once the charging capacitor exceeds a threshold value. The time it takes to reach this threshold depends on the final collected charge of the MPPC. This means that a lower charge signal would receive a later timestamp than a higher charge signal even if the signals occurred at the same time. This effect is known as the electronic timewalk and is corrected for.

The light emission rate of the WLS fibres follows an exponential decay function which means that the emission rate depends on the total number of photo-electrons. As described above, the Trip-T electronics only assign a timestamp once the collected charge passes a threshold. So, the fibre decay causes a separate timewalk effect, called the fibre timewalk effect which is also corrected for.

3.3.2 Event reconstruction

Once the information has been calibrated it is passed to the ND280 reconstruction software. The reconstruction algorithms are separated into two phases: the local reconstruction and the global reconstruction. The local reconstruction, which is run first, is separated into a set of algorithms specific to each subdetector. Each subdetector reconstruction attempts to form its own picture of the event which passed through it. After this, all of the local reconstruction information is passed to the global reconstruction which attempts to match and refit the subdetector results to maximise the amount of extractable information.

3.3.3 Data reduction and summarising

The final stage of the processing chain is to reduce and summarise all of the previous steps in the chain such that they are in a suitable format for the analyser. A lot of the information, particularly from the reconstruction stage, is stripped out and the physics related objects are extracted. Most importantly, all of the information is summarised in a pure ROOT format, meaning the analyser does not rely on the oaEvent library to analyse ND280 data.

3.4 ECal event reconstruction

As mentioned in section 3.3.2, part of the reconstruction process is to run algorithms specific to the ND280 subdetectors. The ECal is no exception and is equipped with an extensive suite of reconstruction algorithms designed to reconstruct events originating from the tracker region of ND280. The inputs to the reconstruction are the calibrated scintillators bar hits which are used to form 3D objects and attach a topology hypothesis.

3.4.1 Hit preparation

The initial ECal reconstruction stage takes the hits outputted by the calibration stage and prepares them for the downstream algorithms. The ECal hits arrive with timestamps but are not separated according to the bunch structure. So, the hits are ordered

in time and then grouped into buckets where a new bucket starts when a greater than 50 ns gap appears between two time-adjacent hits. A second filter is then applied to arrange the hits according to the sensor they occurred on. For double ended bars with both sensors activated, the time of the hit is re-estimated by averaging the timestamp of the two sensors. The two hits are then merged to form a single hit.
It is then necessary to apply extra calibrations to the charges of the hits. The effect of light attenuation along the WLS fibre is corrected for and a scaling is applied to convert the charge into MIP equivalent units (MEU).

3.4.2 Basic clustering

The time ordered hits are then passed to a set of clustering algorithms which attempts to form an object out of the ECal hits. The first stage of this is called basic clustering which is a nearest neighbour algorithm designed to form 2D clusters of hits for both views of an ECal module. This is initiated by forming a 30 ns window and searching for the highest charge hit contained within it to form a seed. The seed cluster is expanded by searching for and adding candidate hits which pass the following criteria:

- Is located in the 30 ns time window
- Is at most one bar away from a hit in the cluster
- Is at most two layers away from a hit in the cluster

To qualify as a basic cluster, any formed cluster must contain at least three hits. The successfully formed clusters in both views of the ECal modules are then passed to the next stage of the clustering algorithms.

3.4.3 Cluster combination

The second stage of the clustering algorithm takes the basic clusters in a given view and attempts to form merged clusters if a set of conditions are passed. The cluster with the highest number of hits is taken as a seed and compared with all other clusters in the same view. A Principal Components Analysis (PCA) is applied to all 2D clusters as the primary axis is used in one of the merging criteria. A candidate cluster is merged with the seed cluster if:

- The distance of closest approach, taken from the primary axes, of the candidate and seed cluster is less than 80 mm
- The charge weighted average hit times of the candidate cluster is within 40 ns of the seed cluster
- The charge weighted average distance of the candidate cluster to the seed cluster is less than 400 mm

All candidate clusters which pass the conditions are merged with the seed cluster. After all comparisons have been made, a new seed cluster is found and the process is repeated until no more merges take place. All clusters in each ECal view are then passed into the next clustering algorithm.

3.4.4 Cluster expansion

The next and final stage of the 2D clustering attempts to merge any unmatched hits with the already formed clusters if the comparison passes a set of conditions. As before, a PCA is applied to all 2D clusters to calculate the primary and secondary axes as these are used in one of the matching conditions. Let $\vec{n}_{\text{pri}}$ and $\vec{n}_{\text{sec}}$ be the primary and secondary axes of a cluster found by the PCA. The squared spread of the cluster along the primary or secondary axis is then defined as

$$\sigma_A^2 = \sum_{\alpha=x,y,z} \sigma_\alpha^2 n_{A,\alpha}^2, \tag{3.1}$$

where A refers to either the primary or secondary axis and σ_α refers to the cluster's spread along the cartesean axes. A metric for defining well the hit matches to the cluster can then be defined as

$$w = \sqrt{\left(\frac{\vec{H}\cdot\vec{n}_{\text{pri}}}{\sigma_{\text{pri}}}\right)^2 + \left(\frac{\vec{H}\cdot\vec{n}_{\text{sec}}}{\sigma_{\text{sec}}}\right)^2}, \tag{3.2}$$

where H is the vector joining the charge-weighted centre of the cluster and the centre of the hit. The unmatched hit is merged with the 2D cluster if:

- The unmatched hit is within 6 ns of one of the constituent hits of the 2D cluster

- The difference in bar number of the unmatched hit and at least one of the constituents hits in the 2D cluster is less than 11
- The difference in layer number of the unmatched hit and at least one of the constituents hits in the 2D cluster is less than 21
- The calculate value of w is less than 50

Unlike the previous stage where it is possible to merge every cluster in a view, an unmatched hit can be matched with one, and only one, 2D cluster. So, not only does the hit-cluster comparison have to pass the relevant criteria, the comparison has to be a better match than all other comparisons. In cases where the unmatched hit passes all of the conditions with more than one cluster, the comparison which provides the lowest w is selected as the best match.
The 2D clusters in each view are then passed onto the final stage of the clustering algorithms.

3.4.5 3D cluster formation

The final stage of the clustering algorithm attempts to form full 3D clusters information from both view of an ECal module. To do this, the algorithm takes the 2D clusters from one view and matches them with the 2D clusters in the other view. All 2D clusters from one view are compared to all 2D clusters in the other view and the two that are the best match form a 3D cluster. The metric used for the matching is a simple likelihood based on three input variables. The first variable is the ratio of the total charge in 2D matching candidate clusters, Q_{ratio}. Assuming the two 2D clusters come from the same particle, the amount of charge deposited in each view should be similar, so Q_{ratio} should be ~ 1. The Q_{ratio} value is then compared with a probability density function generated using ND280 MC to retrieve $\mathcal{L}_{Q_{\text{ratio}}}$. The second input variable to the likelihood compares the first layer used by the two matching candidate clusters which is closest to the ND280 tracker region, $\Delta_{\text{layer, first}}$. If the two candidate clusters are created from the same particle, the difference in starting layer number should ideally be one. As with with Q_{ratio}, the $\Delta_{\text{layer, first}}$ found for the matching candidates is compared to a probability density function to retrieve $\mathcal{L}_{\Delta_{\text{layer, first}}}$. The final input to the likelihood is almost identical to the second input. The variable, called $\Delta_{\text{layer, last}}$ compares the difference in layer number furthest away from the ND280 tracker region. Besides this difference, $\Delta_{\text{layer, last}}$ is treated in exactly the same manner as $\Delta_{\text{layer, first}}$

and so will not be described in any more detail.
Once the three inputs for the matching candidates have been calculated, the value of the matching likelihood is then

$$\mathcal{L} = \mathcal{L}_{Q_{\text{ratio}}} \times \mathcal{L}_{\Delta_{\text{layer, first}}} \times \mathcal{L}_{\Delta_{\text{layer, last}}}. \tag{3.3}$$

As mentioned above, all 2D clusters in one view are compared with all 2D clusters in the other view. The matching pair which produce the highest value of $\mathcal{L}$ are counted as a match. The matched pair are then removed from the pool and the process is repeated until no more matches can be made or the value of $\mathcal{L}$ falls below e^{-5}. All matched clusters are now counted as 3D objects and are passed onto the rest of the ECal reconstruction algorithms.

3.4.6 3D hit reconstruction

So far in the ECal reconstruction, it has only been possible to reconstruct 2/3 of the hit coordinates have been estimated. However, now that full 3D clusters have been formed, the information from both ECal views can be used to estimate the 3rd hit coordinate which is the coordinate along the length of the bar.
The hits are separated into their respective 2D view and then the charge-weighted average position of each view is calculated. Then, to calculate the final hit coordinate, the nearest four layers from the other view are found and a linear least-squares fit is applied to their 2D coordinates. The line found is then extrapolated into the layer of interest which provides the unknown hit coordinate.

3.4.7 Energy reconstruction

The ECal was designed to catch electromagnetic particles originating from the tracker region of ND280, particularly photons from π^0 decays. An integral part of particle shower reconstruction is estimation of the total energy of the incoming particle. So, the next phase of the reconstruction is particle energy estimation under the assumption that the incident particle created a full contained particle shower upon entering the ECal module. The range of the energy fitter is 25 MeV to 20 GeV which almost fully covers the range of energies seen in the ND280 ECals. The energy fitter works by minimising a likelihood function which takes the total charge, charge RMS and charge

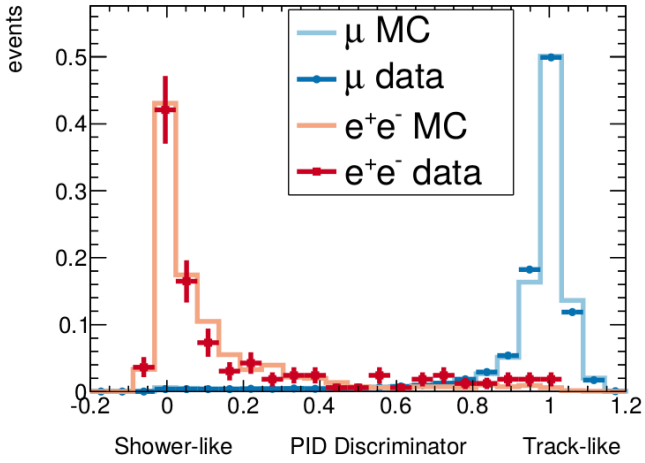

Figure 3.1: The track-like/shower-like discriminator for the DS-ECal. Solid lines show the control sample Monte Carlo and the point show the control sample data.

skew of the 3D cluster as inputs. For each of the three input variables, splines were generated using simple photon particle gun MC which relates the input variables to the true particle energy. The minimizer generates an estimate of the true energy and, via the splines, retrieves the expected values of the input variables. By minimising the distance between the measured input variables and those taken from the splines an accurate estimate of the particle energy is found. This energy estimate, along with the 3D cluster itself is then passed onto the final section of the ECal reconstruction.

3.4.8 Event classification

The final section of the ECal reconstruction is to attach a topology hypthosis to the 3D cluster. The routines currently separate the clusters into two catagories: those that are shower-like and those that are track-like using an Artifical Neural Network. This multi-variate analysis object takes multiple pieces of information about the 3D cluster which attempts to measure the energy deposition and shape of the cluster and returns a single number. The value of the returned number suggests whether the reconstructed object is track-like or shower-like. The discriminator was tested using ND280 electron and muon control samples, the results of which are shown for the DS-ECal in Fig. 3.1.

Chapter 4

Enhanced ECal reconstruction

The current implementation of the ECal reconstruction software was designed to reconstruct particles which originate from the ND280 tracker and enter the ECal. As section 3.4 shows, this was realised by only considering ECal hit clusters under the single track-like or shower-like hypothesis. It should be evident that a neutrino interaction occurring within the ECal does not well fit this topology. While it is true that there is some power in the current reconstruction to distinguish a neutrino interaction from an entering track or shower, there is little feature information available. How many final state particles propagated from the interaction? How much visible energy was deposited by each of the particles? Where in the ECal did the interaction occur? These basic questions can not be trivially answered when using the current reconstruction. To maximise the ability of distinguishing ECal neutrino interactions from entering backgrounds, the reconstruction must be revisited.

4.1 The Hough transform

The Hough transform is a popular method of machine pattern recognition used by, but is not limited to, high energy physics experiments. Originally designed for machine track recognition in bubble chamber pictures [19], the version most widely used throughout the world was developed in 1972 [20]. The Hough transform is used to isolate specific features or shapes from a digital image. The simplest implementation, which is of most interest in event reconstruction, allows the extraction of straight, 2D lines from a complex pattern. This is acheived by exploitation of a very simple feature of 2D geometry.

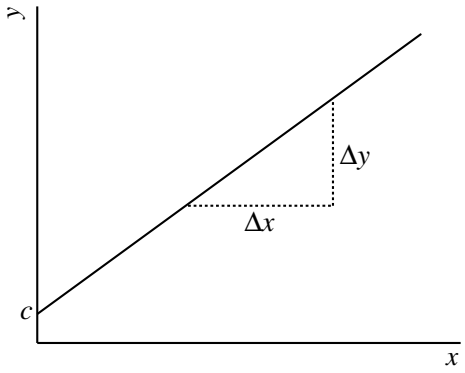

Figure 4.1: Line in 2D cartesian space.

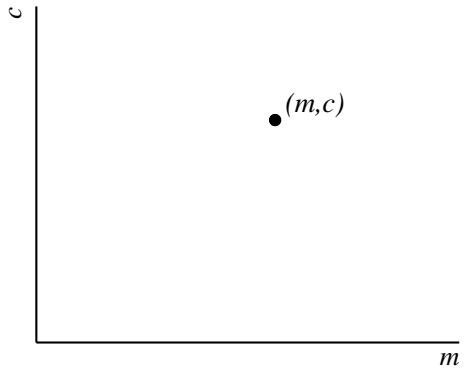

Figure 4.2: Point in 2D parameter space.

4.1.1 Line-point duality

Consider a straight line formed in a 2D cartesian space as shown in Fig. 4.1. The line is usually described by

$$y = mx + c \tag{4.1}$$

where y and x are used as coordinates, m is the gradient of the line and c is the intercept location of the line with the y axis. While it is not necessary to analyse this simple shape in great detail, it is important to note that m and c are the only parameters necessary to completely describe the line.

Now consider a new 2D space where the axes are defined by m and c rather than x and y (here after referred to as the parameter space). As this parameter space is described by the parameters of a general 2D cartesian line, there is an underlying symmetry between the two spaces. The parameters of the 2D line shown in Fig. 4.1 can be used to form a pair of coordinates (m,c) in the parameter space as shown in Fig. 4.2. It is important here to state clearly the general result; a straight line in cartesian space is represented by a single point in parameter space.

Now consider the 2D cartesian space again. Unlike before, we will define a single point rather than a straight line. Such a point is traditionally described by a pair of coordinates (x,y). However, an alternative description of the point is an infinite number of lines all of which pass through (x,y). This is highlighted by Fig. 4.3 where three lines of the infinite set are shown along with the point they represent. As the infinite line set are used to describe a single point, all lines in the set must follow a pattern. This relationship is revealed by simple algebraic manipulation of equation 4.1

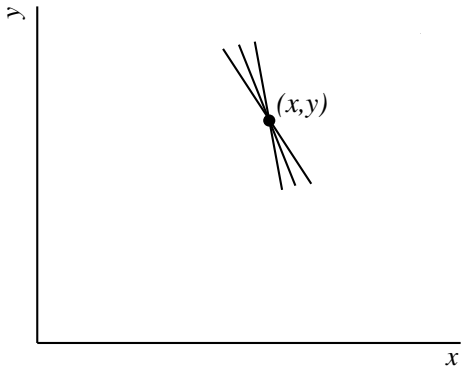

Figure 4.3: Point in 2D cartesian space.

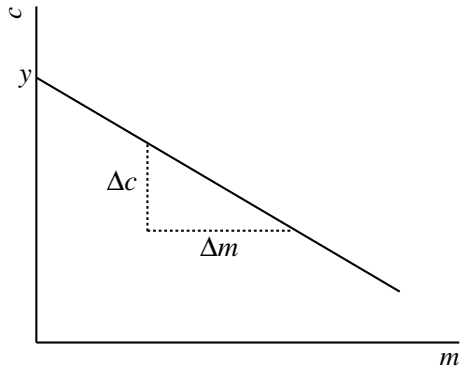

Figure 4.4: Line in 2D parameter space.

to give

$$c = -xm + y. \tag{4.2}$$

Despite the manipulation, equation 4.2 still resembles the equation of a 2D line, however the parameters are x and y and the coordinates are m and c. Specifically, equation 4.2 is represented by a line in the parameter space defined above. The gradient of this line is

$$x = \frac{\Delta c}{\Delta m} \tag{4.3}$$

and the intercept of the line with the c axis is y as shown in Fig. 4.4. As before, it is important to clearly state what has been shown; a point in cartesian space is represented by a line in parameter space.

4.1.2 The parameter space

As section 4.1.1 shows, there is a clear relationship between the cartesian space and the parameter space. Specifically, there is a symmetry between lines and points in the two spaces. This relationship between the cartesian and parameter spaces is not only interesting but also very powerful. Consider again the parameter line defined by equation 4.2 and shown in Fig. 4.4. As shown in section 4.1.1, equation 4.2 was derived by considering the infinite set of lines which pass through a cartesian point. As this infinite set represents the parameter line, it must also be true that the parameter line represents the infinite line set. Using one of the results from section 4.1.1, any point

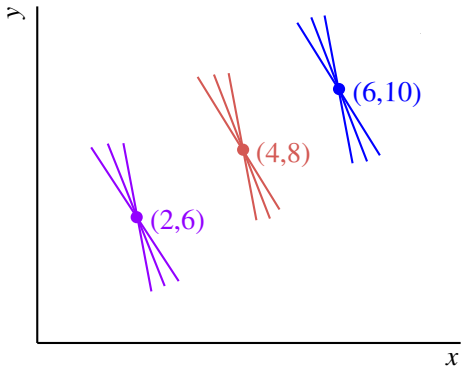

Figure 4.5: The three cartesian points defined in equation 4.4. The colour coding matches that of Fig. 4.6.

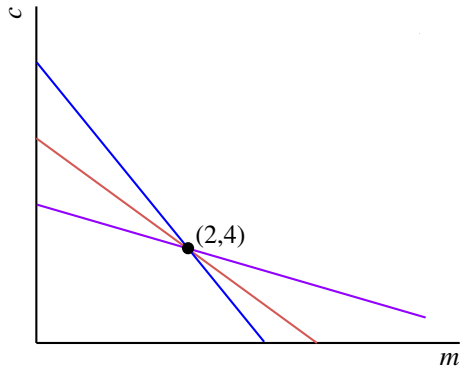

Figure 4.6: The three parameter lines defined in equation 4.5. The same colour coding as Fig. 4.5 is used.

along the parameter line represents one of the lines from our infinite set. This key feature of the parameter space is the central component of the Hough transform.
We will now return to the cartesian space for an example of how the Hough transform works. Let's define three points in this space,

$$\begin{aligned} p_1 &: (2,6) \\ p_2 &: (4,8) \\ p_3 &: (6,10). \end{aligned} \tag{4.4}$$

The three points defined in equation 4.4 are shown in Fig. 4.5. Using one of the results from section 4.1.1 and equation 4.2, the three points can be Hough transformed into the following parameter lines:

$$\begin{aligned} c &= -2m + 6 \\ c &= -4m + 8 \\ c &= -6m + 10. \end{aligned} \tag{4.5}$$

The three parameter lines are shown in Fig. 4.6. From Fig. 4.6, it is clear that the three parameter lines all cross at a common point with parameter space coordinates (2,4). Using the results from section 4.1.1, this common point in parameter space is represented by a line in cartesian space. In addition, as (2,4) is common to all three parameter lines, the cartesian line represented by (2,4) must also pass through all three

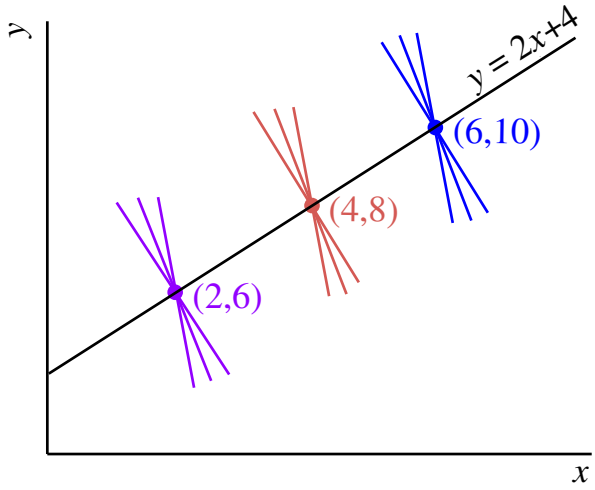

Figure 4.7: The line represented by the intersection in Fig. 4.6 with the cartesian points it intercepts.

cartesian points defined by equation 4.4. This new cartesian line is defined by

$$y = 2x + 4 \tag{4.6}$$

and is shown in Fig. 4.7 with the original points used to generate the parameter lines. While this example is relatively simple, it demonstrates the capability of the Hough transform to recognise linear patterns in sets of points.

4.1.3 Redefinition of parameters

Unfortunately, a complication in computation arises when $m \to \infty$. However, this complication can be removed by redefining the line parameters such that one is bounded. An alternative 2D line parameterisation is to specify a line in terms of the angle it makes with the x axis, θ, and the perpendicular distance of the line from the origin, ρ, as illustrated in Fig. 4.8. The functional form of the cartesian line becomes

$$y = x\tan\theta + \frac{\rho}{\cos\theta}. \tag{4.7}$$

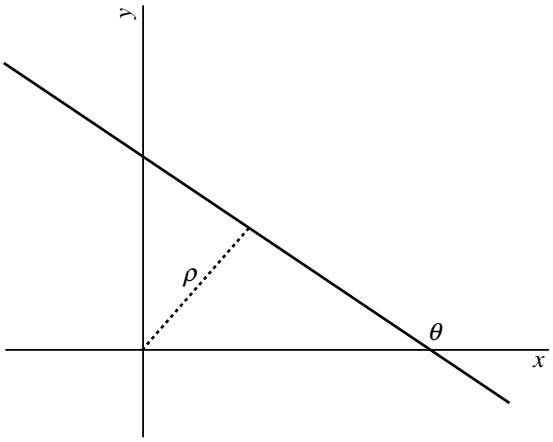

Figure 4.8: $\theta - \rho$ parameterisation of a 2D line.

Using this new parameterisation, we must also define a new parameter space with axes θ and ρ. Using equation 4.7, a line in this new parameter space is defined by

$$\rho = y\cos\theta - x\sin\theta. \tag{4.8}$$

By definition, the θ axis of the parameter space must be bounded to the interval $[0,2\pi]$. If the directionality of the line is meaningless to the analyser, then the interval can be restricted to $[0,\pi]$.
The disadvantage of this parameterisation is that parameter line generation now involves trigonometric calculations which can be computationally expensive. However, this problem is small when compared to the unbounded complication of the traditional parameterisation.

4.1.4 Discretisation of the parameter space

It must now be considered how the parameter space is analysed. When a large number of parameter lines are generated, it becomes computationally expensive to analyse the resultant parameter space. While approaches exist to analyse the parameter space with very high precision [21], it is often only necessary to extract parameters with finite resolution. In such a case, it is convenient to discretise the parameter space. Under this regime, the parameter space is split into $\theta - \rho$ bins. Then, a parameter line is generated by incrementing the value of each $\theta - \rho$ bin it passes through. After each line has been

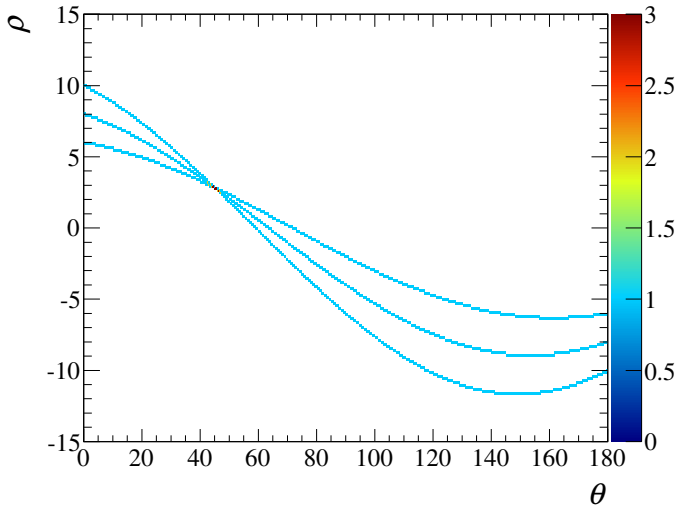

Figure 4.9: The discrete $\theta - \rho$ space. The plotted lines are those defined in equation 4.5 and reparameterised using equation 4.8.

added to the parameter space, the crossing locations can be readily found by searching for the $\theta - \rho$ bins with content larger than unity. An example of this discretisation is illustrated in Fig. 4.9, where the parameter lines defined in equation 4.5 have be re-parameterised using equation 4.8 and plotted. The content of each bin in Fig. 4.9 records how many of the three parameter lines pass through each bin. The bin with value 3 is the crossing point of the three parameter lines.
If a discretised approach is acceptable, which is the case in event reconstruction, construction and analysis of the parameter space is reduced to filling a 1D array N times, where N is the number of parameter lines, followed by a 1D grid search of the array to find the bin with the highest content.

4.2 ECal application of the Hough transform

We must now address how the Hough transform can be used as a reconstruction tool in the ECal. To do this, let's consider a neutrino interaction which occurs in the ECal as illustrated in 4.10. While the propagating neutrino is invisible to the ECal, the charged final state are definitely not. To first order, the final state particles propagate in straight lines depositing energy in the scintillator bars as they go. From this, we can infer that the hit bars arranged in straight lines should reveal the trajectory of the final states. As shown above, the Hough transform is capable of identifying straight lines from a

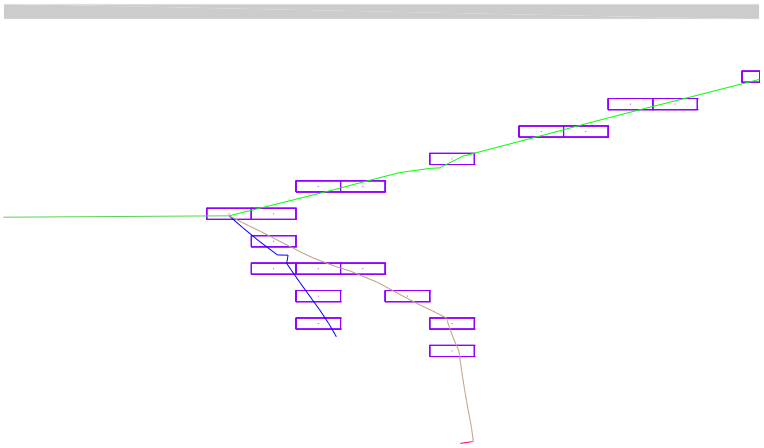

Figure 4.10: Simulated neutrino interaction with 3 final states in the side-right ECal. The green line entering from the left is the ν_μ. The green line exiting to the right is a μ^-, the brown line is a π^+ and the blue line is a proton. The purple rectangles represent the hit ECal bars.

set of coordinates. However, there are two complications in the ECal which the above sections have not addressed. We have only specifically discussed how to extract a single straight line from a pattern. As Fig. 4.10 shows, the number of final states can be, and is often, greater than one. This is merely a problem of computation which will be addressed in section 4.2.3. A much more severe problem is that the above demonstrations only deal with patterns constructed from infinitesimal points. While the centre of a scintillator bar can be used as a point for parameter line generation, it is unlikely that a final state particle will pass through the central point of the scintillator bars that it propagates through. If this is not addressed, the Hough transform will be of little use in trajectory reconstruction.

4.2.1 Modelling the ECal bar

To make the Hough transform viable as a reconstruction tool, the finite dimensions of the ECal bar need to be incorporated into the parameter space generation. This feature of the ECal bars would be very problematic if the parameter space was continous. However, it is only necessary to know the line parameters with finite resolution and a discrete parameter space can be used. This means that the ECal bar can be modelled as a set of cartesian points and each of said points can be Hough transformed in turn to build up the parameter line representation of the ECal bar.

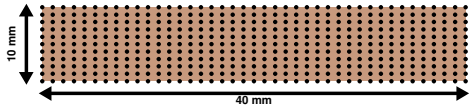

Figure 4.11: Grid representation of an ECal bar.

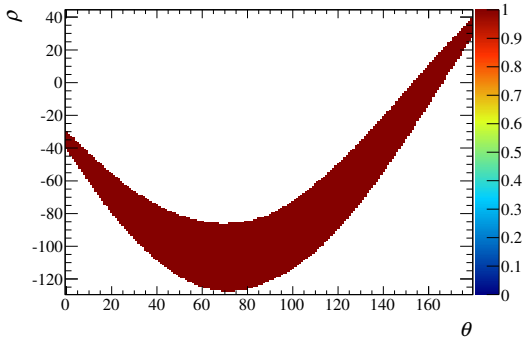

Figure 4.12: Single ECal bar Hough transform using the point configuration shown in Fig. 4.11.

There are now two steps to consider. Firstly, how should the points be arranged? Remember that the Hough transform of a point represents all of the lines that pass through that point. So, the points should be arranged in such a fashion that any line which passes through the 2D cross-section of the ECal bar also has to pass through one of the points in the configuration. Secondly, the spacings between the points should be small enough that no gaps appear in the generated parameter line.
Assuming that every bin of the parameter space will have a $1^\circ \times 1$ mm area, an obvious choice would be to use a rectangular grid of points with 1 mm spacing superimposed over the 2D cross-section of an ECal bar. The total number of points used to model the ECal bar is 451. To Hough transform the ECal bar, every point in the grid array can be Hough transformed individually with care taken to ensure that each $\theta - \rho$ bin is filled exactly once. The result is illustrated in Fig. 4.12. The finite size of the ECal bar is evident by the finite size of the resultant parameter line.
While the generated parameter line accurately represents every line which passes through the ECal bar, there are two problems with this approach. Firstly, the large number of points to be Hough transformed is very large which results in a long CPU time. Secondly, there is a very high number of redundant calculations involved in the parameter line generation. Consider an exactly vertical line which passes through one of the points in the grid array. This line also passes through 10 other points in the same column of the grid. This means that when the parameter line is being generated, this vertical line is calculated 11 times for each column. Bearing this in mind, there are many points along the parameter line which are repeatedly calculated and provide no extra information. This would mean that any algorithm which uses this approach would be very CPU inefficient.

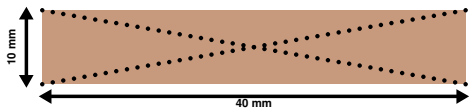

Figure 4.13: Cross representation of an ECal bar.

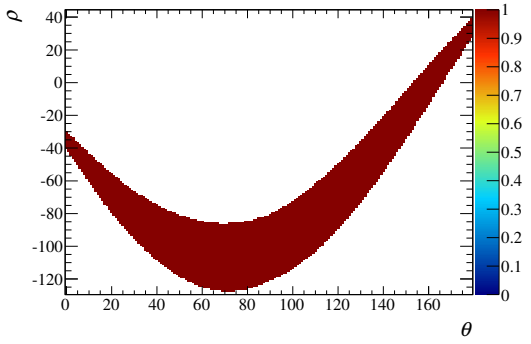

Figure 4.14: Single ECal bar Hough transform using the point configuration shown in Fig. 4.13.

An alternative is to model the ECal bar as a set of points arranged in a cross as shown in Fig. 4.13. Assuming that the spacing bettween the points on each line of the cross is infinitesimal, any line which passes through the ECal bar would also have to pass through one of the points in the configuration. As the parameter space is discrete, the spacing between the points need to be infinitesimal but only small enough to ensure that no gaps appear in the parameter line. Using 45 points on each line of the cross, the ECal bar can be Hough transformed by Hough transforming each point in the cross. An example of this result is shown in Fig. 4.14 using the same ECal bar used to generate Fig. 4.12. Clearly, Fig. 4.12 and Fig. 4.14 are identical showing that the cross model achieves the same result as the grid model. Comparing the two, the cross model uses a 90 point representation whereas the grid model uses a 451 point representation. This should mean that an algorithm utilising the cross model would be a factor of five faster than one using a grid model. In addition to this, the number of redundant calculations is heavily reduced.

4.2.2 Parameter space generation

As we have addressed how to Hough transform an ECal bar, we are now in suitable position to generate the full parameter space for an ECal cluster. As described in section 4.1.3, the parameterisation of the 2D lines requires some point in space to act as the origin. It is possible to use the origin defined by the global ND280 geometry, however this is located in TPC 1 which would mean ρ will usually be the order of metres. It is more convenient to define an origin in the vicinity of the ECal cluster being reconstructed. A simple option is to use the charge-weighted centre of the ECal

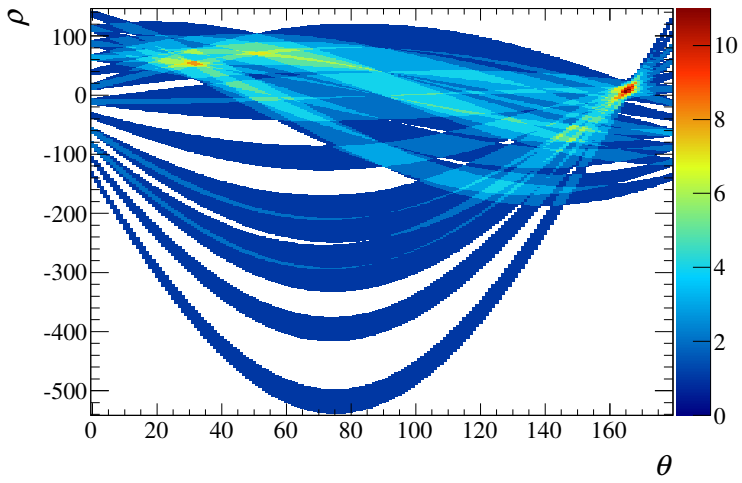

Figure 4.15: The full parameter space of the 2D cluster shown in Fig. 4.10.

cluster as the origin of the Hough transform. This location is simple to calculate and generally keeps ρ small.

The provided description of the Hough transform in all previous sections is strictly defined in 2D and so the ECal cluster should be split in such a way that this definition can be used. Fortunately, a 3D ECal cluster is built up using the two 2D views that the scintillator layers provide. So, it is relatively easy to split the 3D cluster into a pair of 2D clusters by collecting the cluster's hits into their respective 2D views.

We can now partly answer one of the problems raised in section 4.2 which is how to handle the track multiplicity aspect of the reconstruction? This is partly addressed by generating N parameter spaces with the same $\theta - \rho$ bin configuration where N is the number of hits in the 2D cluster. Each of the N parameter spaces will hold one parameter line generated by one of the 2D hits (in a similar fashion to Fig. 4.14). The final parameter space can then be generated by adding together each of the N parameter spaces. The parameter space of the 2D cluster in Fig 4.10 is shown in Fig. 4.15.

4.2.3 Parameter space analysis

The full parameter space can look arbitarily complicated. However, it contains a vast amount of trajectory related information about the cluster. Every $\theta - \rho$ bin of the parameter space describes a 2D track and the content value of said bin describes how many 2D ECal hits the track passes through. As described in section 4.2, a particles

trajectory should be straight in the ECal which means that the particles path should be revealed by finding the most hits arranged in a line. This hit arrangement can be found by finding the bin in the full parameter space with the highest value. The track candidate parameters can the be found by fetching the (θ,ρ) coordinate of the found bin.
While the preferred bin can reveal how many hits the track candidate passed through, it contains no information about which hits were contributors. However, this full parameter space was generated by summing the N parameter spaces discussed in section 4.2.2. So, the contributing hits can be readily found by looking at the same (θ,ρ) bin in each of the N parameter spaces and recording which have a non-zero value. We now have the track candidate's parameters and its contributing hits which is enough to describe the 2D trajectory.
A new search now needs to begin to find any other track candidates. However, repeating the same search of the full parameter space will return the track candidate that has already been found. To find the next track candidate, the presence of the previous track candidate must be removed. So, a reduced parameter space must be generated. The previous step found which of the N parameter lines contributed to the previous track candidate. So, this new parameter space can be formed by subtracting the contributing parameter lines from the full space. An example of this is shown in Fig. 4.16 where the reduced parameter space was formed by subtracting the contributing parameter lines to the highest bin in Fig. 4.15. The next track candidate can then be found by searching for the highest content bin of this reduced parameter space and said bin's contributing parameter lines.
This process can be repeated until some threshold is reached. This threshold is nominally set by demanding that at least three hits are required to form a track candidate.

4.2.4 3D track reconstruction

Section 4.2.3 describes the track reconstruction of a 2D ECal cluster. However, a 3D cluster consists of two sets of 2D clusters. So, the process described in section 4.2.3 must be performed on each of the 2D clusters. The result of this process is two sets of 2D tracks. To form full 3D tracks, the tracks from each view must now be matched together. This is acheived by making every pairwise comparison of the tracks from each view to find the pair which are most similar to each other. After such a pair is

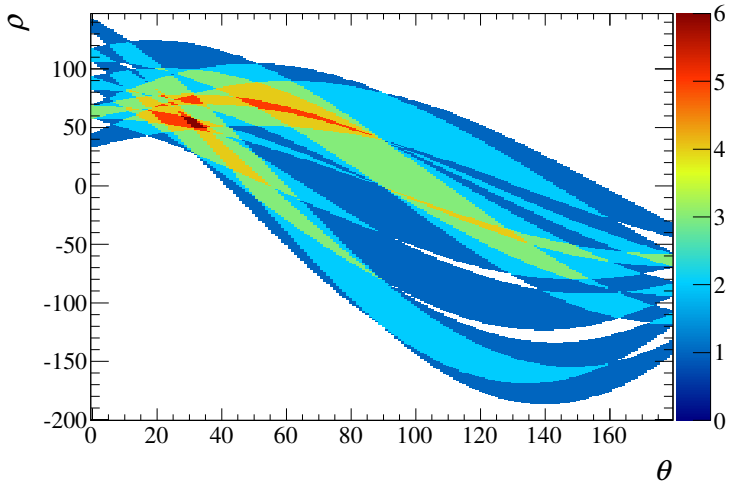

Figure 4.16: The reduced parameter space of the 2D cluster shown in Fig. 4.10.

found, the tracks are removed from the pool and the process is repeated to find the next pair until either no tracks are left or one 2D tracks are left. In the latter case the single 2D track is discarded. Every pairwise combination of tracks is used to form a likelihood $\mathcal{L}$. The pair which produces the highest $\mathcal{L}$ is declared the best match and removed from the pool. Three pieces of information about the matching pair are used to calculate $\mathcal{L}$, all of which make use of probability density distributions generated using beam Monte Carlo.

As should be expected, a vertex with one visible track will have different characteristics to a vertex with three visible tracks. So, to maximise the ability of the matcher, a different set of probability density distributions are used for the 1, 2 and 3 track cases. If the number of tracks in each view is not identical, the higher number of tracks is used to find the correct probability density distributions. Seperately, due to their geometrical differences, the reconstructed shape of vertices in the DS ECal will differ to those in the barrel ECals. This leads to a separate set of probability density distributions for the barrel and DS ECal modules.

The first input to the likelihood is the ratio of the total deposited charge on each track Q_{ratio}. The denominator is taken as the track which comes from the view with the most hits. Generally speaking, a particle propagating through an ECal module should deposit a similar amount of charge in each of the two views. So, Q_{ratio} should have a value close to 1 if the two 2D tracks are created by the same particle. An example of the Q_{ratio} distribution is shown in Fig. 4.17, taken from beam Monte Carlo in the barrel ECals for cases where the maximum number of tracks found in a given view is 2. In Fig. 4.17, correctly matched (in blue) shows the Q_{ratio} distribution for matching

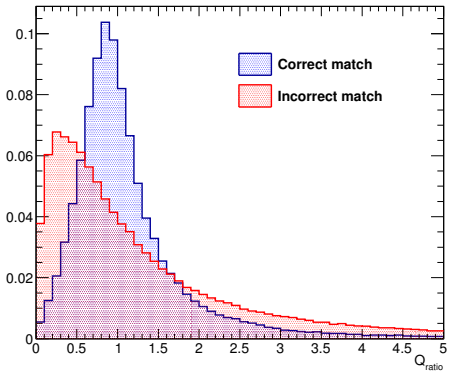

Figure 4.17: Q_{ratio} distribution in the barrel ECal for the two track case. The blue distribution refers to matching pairs which were matched to the same true particle. Both distributions are unit normalised.

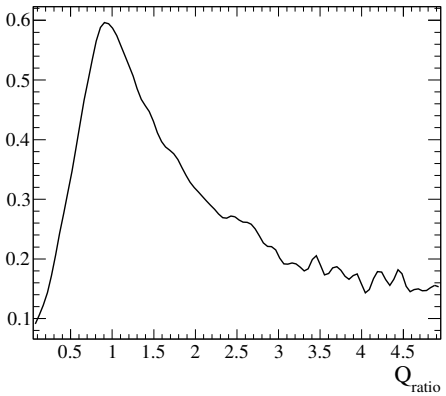

Figure 4.18: Q_{ratio} probability density distribution in the barrel ECal for the two track case.

pairs which come from the same particle and incorrectly matched (in red) shows the the Q_{ratio} distribution for matching pairs which were created by different particles. As Fig. 4.17 shows, Q_{ratio} well separates the two cases. To generate a probability density distribution for Q_{ratio}, the two distributions shown in Fig. 4.17 are used, but without applying any normalisation. By comparing the bins of each distribution, the probability for correctly matching two tracks in a given bin p_i can be formed by

$$p_i = \frac{s_i}{s_i + b_i}, \tag{4.9}$$

where s_i is the number of correctly matched tracks in bin i and b_i is the number of incorrectly matched tracks in bin i. A discreet probability density distribution for Q_{ratio} can then be formed by calculating p_i for every bin. The discreet probability density distribution is then interpolated with splines to create the final probability distribution. An example of this for the two track, barrel case is shown in Fig. 4.18. When a matching candidate pair is being considered, the value of Q_{ratio} is calculated and used in the spline to retrieve $\mathcal{L}_{Q_{\text{ratio}}}$.

The second input to the likelihood, called $\Delta_{\text{layer, first}}$, is the difference in the starting layer of each 2D track which forms the matching candidate pair, where starting layer refers to the layer closest to the ND280 tracker. For 2D tracks which should be matched together, $\Delta_{\text{layer, first}}$ should be 1. The separation ability of this variable for the two track,

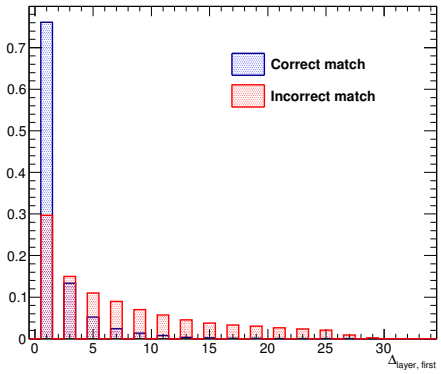

Figure 4.19: $\Delta_{\text{layer, first}}$ distribution in the barrel ECal for the two track case. The blue distribution refers to matching pairs which were matched to the same true particle. Both distributions are unit normalised.

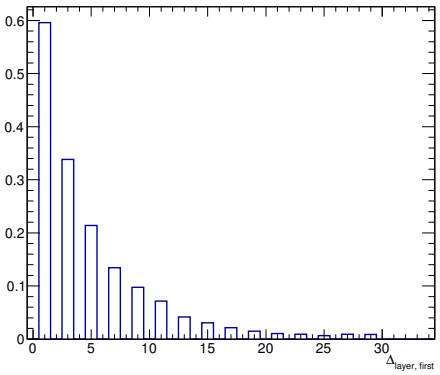

Figure 4.20: $\Delta_{\text{layer,first}}$ probability density distribution in the barrel ECal for the two track case.

barrel is shown in Fig. 4.19. The discreet probability density function was created using eqn. 4.9. It was not necessary to interpolate using splines as $\Delta_{\text{layer, first}}$ is itself discreet. The probability density function for $\Delta_{\text{layer, first}}$ is shown in Fig. 4.20 for the two track, barrel case. For each matching candidate pair, the value of Δlayer, first is calculated and the corresponding $\mathcal{L}_{\Delta_{\text{layer, first}}}$ is retrieved from the probability density function.

The third and final input to the likelihood, called $\Delta_{\text{layer, last}}$, is the differnce in the ending layer of each 2D track which forms the matching cadidate pair, where the ending layer refers to the layer furthest from the ND280 tracker. Functionally, how this function is used is essentially identical to $\Delta_{\text{layer, last}}$ so it will not be described in detail. The separation ability of this variable and its corresponding probabilty density function for the two track, barrel case are shwon in Fig. 4.21 and Fig. 4.22 respectively.

The matching likelihood, $\mathcal{L}$ for a matching candidate pair is then

$$\mathcal{L} = \mathcal{L}_{Q_{\text{ratio}}} \times \mathcal{L}_{\Delta_{\text{layer, first}}} \times \mathcal{L}_{\Delta_{\text{layer, last}}}. \tag{4.10}$$

As described above, $\mathcal{L}$ is calculated for every matching candidate pair and the pair which maximise $\mathcal{L}$ is selected as a match and removed from the pool. The process is then repeated until no more matches can be made.

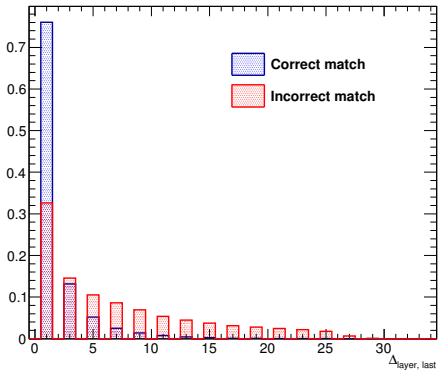

Figure 4.21: $\Delta_{\text{layer, last}}$ distribution in the barrel ECal for the two track case. The blue distribution refers to matching pairs which were matched to the same true particle. Both distributions are unit normalised.

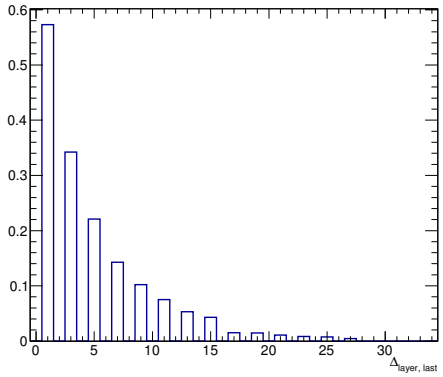

Figure 4.22: $\Delta_{\text{layer,last}}$ probability density distribution in the barrel ECal for the two track case.

3D tracks have now been formed, but the associated directions and positions of those tracks still need to be calculated. The track fitting process for the newly formed 3D tracks is very similar to that described in section 3.4.6. The tracks are briefly separated into their constituent 2D views and a charge-weighted average position of each layer is calculated using the track's constituent hits. Then, the hits in the opposing view are used to estimate the 3rd coordinate of a given layer using a least-squares fit. After all of the coordinates have been estimated, a full 3D least-squares fit of the positions in each layer is performed to estimate the 3D track's direction and position in that ECal layer.

4.2.5 Track pairwise crossing reconstruction

The final step of the reconstruction is to estimate where each of the 3D track's paths cross. As each track is reconstructed as a straight line in 3D, the final step is fairly simple. Using the track direction and position information calculated at the end of section 4.2.4, the position at closest approach for every pairwise combination of 3D tracks is calulated analytically. The distance of closest approach is also calculated. Six hits, the closest three from each 3D track, are then associated to the pairwise crossing.

4.3 Output of the reconstruction

The reconstruction is run over every 3D cluster found in the ECal. By applying the steps outlined above, for each 3D cluster the following output is given:

- A set of 3D tracks
- The pairwise crossings of all 3D tracks found in the cluster

Note that no vertex formation beyond the pairwise crossings is calculated at this stage, nor is any analysis of the 3D tracks performed. While this may seem like an oversight of the reconstruction, this approach was decided as no assumptions are made about what the tracks/crossings represent at this point, making the output more generic. Any analysis which wants to make use of the reconstruction is given enough information to apply more targeted reconstruction downstream.

4.4 Validation of the reconstruction

Someone do this plz

Chapter 5

Magnetic field simulation in ND280

Simulation of the B field

5.1 Magnetic field model in the ND280 flux return

Get a tube of flux

5.2 Effect of magnetic field on the ECal

Better data/MC

Chapter 6

Selection of neutrino interactions in the ECal

Gotta select neutrinos

6.1 Data selection

Data flags

6.2 Monte Carlo selection

Make the cuts

6.3 Properties of events in selection

Chapter 7

Evaluation of systematic uncertainties

Have to handle systematics

7.1 Flux systematics

7.2 Cross-section systematics

7.3 Detector systematics

EUGH

Chapter 8

Cross-section measurement

Measure it

8.1 Measurement method

How to do it

8.2 Validation of method

Does it work?

8.3 Applying the fit to ND280 data

Run it

Chapter 9

Discussion and conclusions

Discuss and conclude

9.1 Discussion

Discuss

9.2 Conclusions

Conclusions

Bibliography

[1] S. Weinberg, Phys. Rev. Lett. **19**, 1264 (1967).

[2] S. L. Glashow, J. Iliopoulos, and L. Maiani, Phys. Rev. **D2**, 1285 (1970).

[3] S. Willenbrock, (2004), hep-ph/0410370.

[4] K. Abe *et al.*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **659**, 106 (2011).

[5] (T2K Collaboration), K. Abe *et al.*, Phys. Rev. Lett. **112**, 061802 (2014).

[6] (T2K Collaboration), K. Abe *et al.*, Phys. Rev. Lett. **113**, 241803 (2014).

[7] K. Abe *et al.*, Phys. Rev. D **87**, 092003 (2013).

[8] R. Brun and F. Rademakers, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **389**, 81 (1997), New Computing Techniques in Physics Research V.

[9] A. Ferrari *et al.*, Fluka: a multi-particle transport code, in *CERN 2005-10 (2005), INFN/TC 05/11, SLAC-R-773*.

[10] R. Brun, F. Bruyant, M. Maire, A. McPherson, and P. Zanarini, (1987).

[11] N. Abgrall *et al.*, Phys. Rev. C **84**, 034604 (2011).

[12] Y. Hayato, Nuclear Physics B - Proceedings Supplements **112**, 171 (2002).

[13] G. A. Miller, Nuclear Physics B - Proceedings Supplements **112**, 223 (2002).

[14] O. Benhar, A. Fabrocini, S. Fantoni, and I. Sick, Nuclear Physics A **579**, 493 (1994).

[15] C. L. Smith, Physics Reports **3**, 261 (1972).

[16] D. Rein and L. M. Sehgal, Annals of Physics **133**, 79 (1981).

[17] T. Sjstrand, S. Mrenna, and P. Skands, Journal of High Energy Physics **2006**, 026

(2006).

[18] S. Agostinelli *et al.*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506**, 250 (2003).

[19] P. a. Hough, Conf.Proc. **C590914**, 554 (1959).

[20] R. O. Duda and P. E. Hart, Commun. ACM **15**, 11 (1972).

[21] J. Vuillemin, Fast linear hough transform, in *Application Specific Array Processors, 1994. Proceedings. International Conference on*, pp. 1–9, 1994.

List of Figures

List of Tables