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

2.1 Accelerator complex

These are words

2.1.1 Proton Accelerators

Stuff about the accelerator

2.1.2 Neutrino beamline

Stuff about the beamline

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 ECal event reconstruction and software

Lots of software stuff

3.1 Real data processing

Process the data. MAY NOT INCLUDE THIS

3.2 Monte Carlo event simulation

Process the MC

3.2.1 Neutrino flux prediction

I guess the flux is quite bit

3.2.2 Neutrino interaction simulation

NEEEEEUT

3.2.3 ND280 detector simulation

nd280mc is really good for this

3.2.4 Detector response simulation

Is elecSim there?

3.2.5 Detector calibration

oaCalib calibrates

3.2.6 Event reconstruction

oaRecon yo!

3.3 ECal event reconstruction

All about ecalRecon

3.3.1 Hit preparation

Hit prep

3.3.2 Basic clustering

Baaasic

3.3.3 Cluster combination

Combine dem clusters

3.3.4 Cluster expansion

Expand dem clusters

3.3.5 3D cluster formation

Match the clusters

3.3.6 3D hit reconstruction

Least square those fits

3.3.7 Energy reconstruction

blah

3.3.8 Event classification

Track or shower?

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 3.3 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 [4], the version most widely used throughout the world was developed in 1972 [5]. 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.

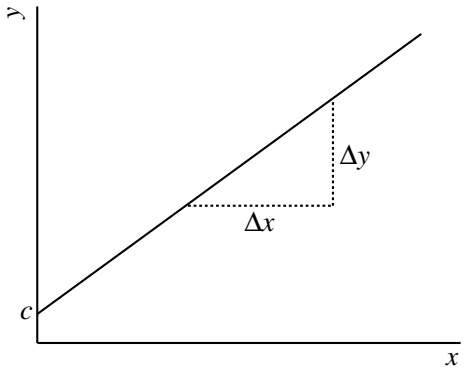

Figure 4.1: Line in 2D cartesian space.

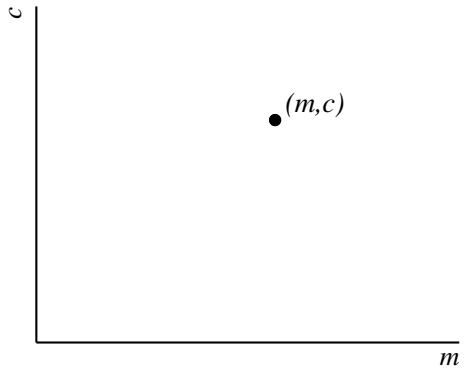

Figure 4.2: Point in 2D parameter space.

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

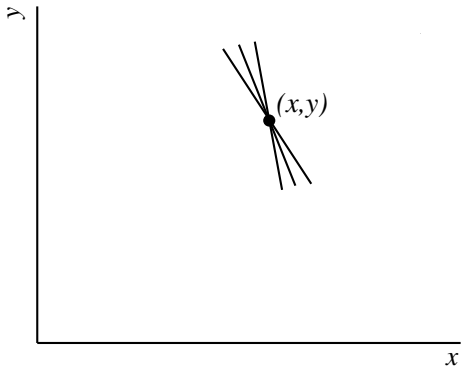

Figure 4.3: Point in 2D cartesian space.

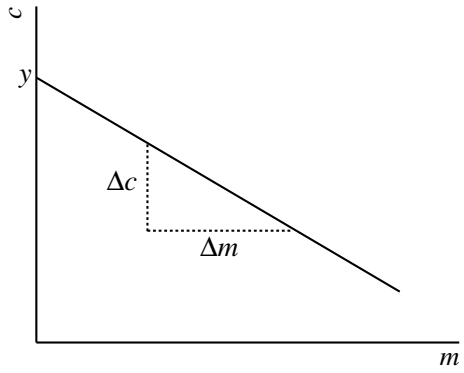

Figure 4.4: Line in 2D parameter space.

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

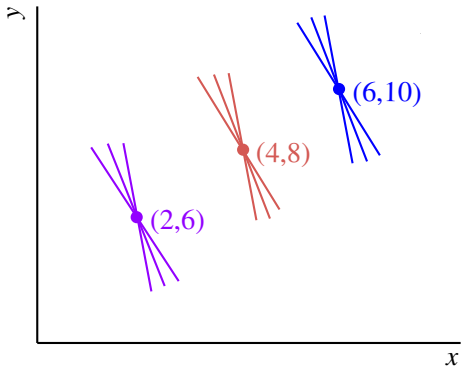

Figure 4.5: The three cartesian points defined in equation 4.4. The colour coding matches that of Fig. 4.6.

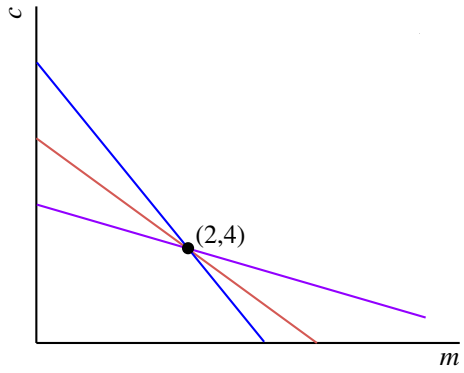

Figure 4.6: The three parameter lines defined in equation 4.5. The same colour coding as Fig. 4.5 is used.

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

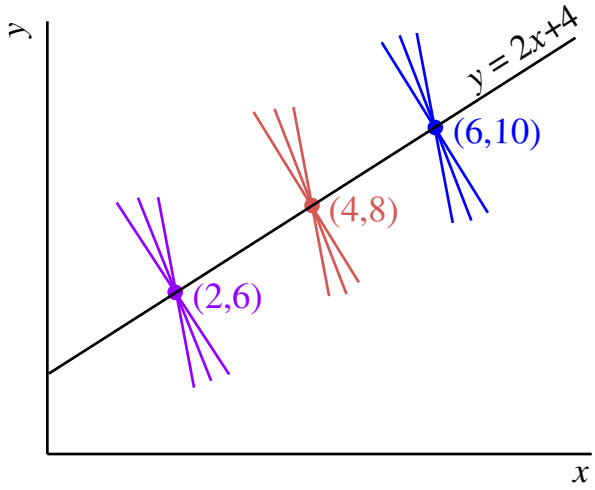

Figure 4.7: The line represented by the intersection in Fig. 4.6 with the cartesian points it intercepts.

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

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

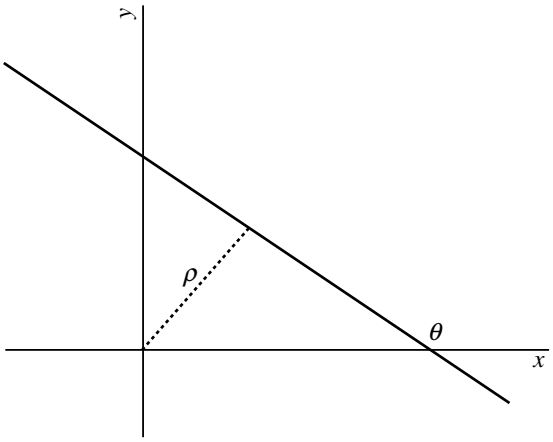

Figure 4.8: $\theta - \rho$ parameterisation of a 2D line.

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 [6], 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

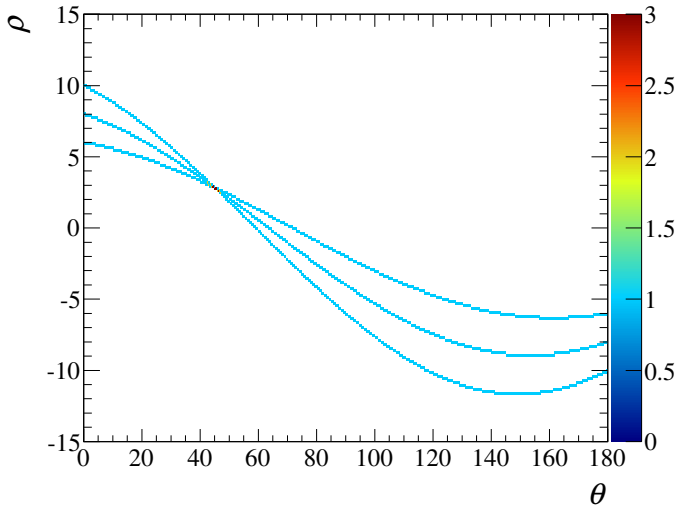

Figure 4.9: The discrete $\theta - \rho$ space. The plotted lines are those defined in equation 4.5 and reparameterised using equation 4.8.

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

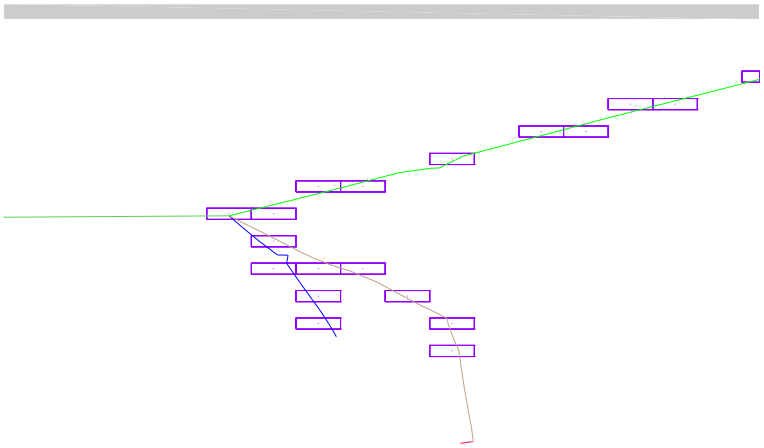

Figure 4.10: Simulated neutrino interaction with 3 final states in the side-right ECal. The green line entering from the left is the ν_μ. The green line exiting to the right is a μ^-, the brown line is a π^+ and the blue line is a proton. The purple rectangles represent the hit ECal bars.

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.

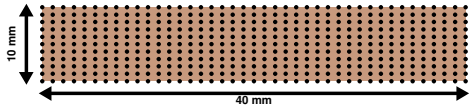

Figure 4.11: Grid representation of an ECal bar.

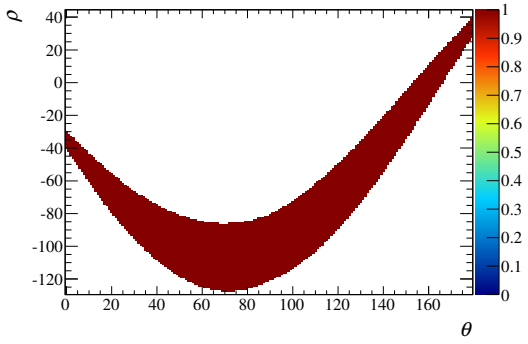

Figure 4.12: Single ECal bar Hough transform using the point configuration shown in Fig. 4.11.

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.

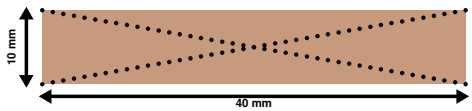

Figure 4.13: Cross representation of an ECal bar.

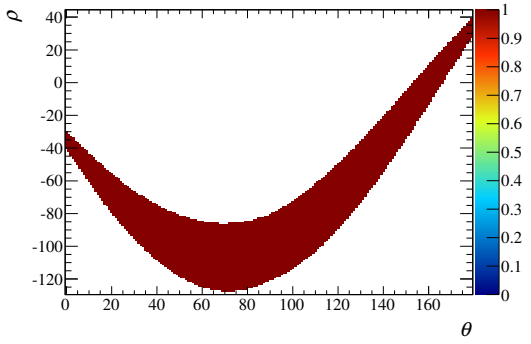

Figure 4.14: Single ECal bar Hough transform using the point configuration shown in Fig. 4.13.

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

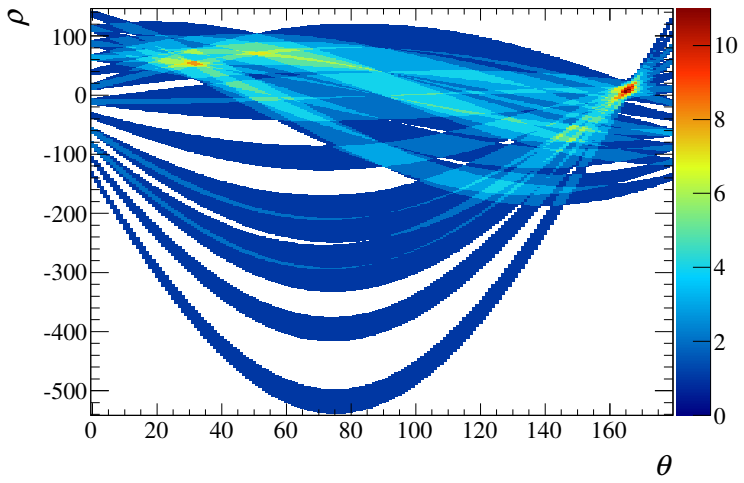

Figure 4.15: The full parameter space of the 2D cluster shown in Fig. 4.10.

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

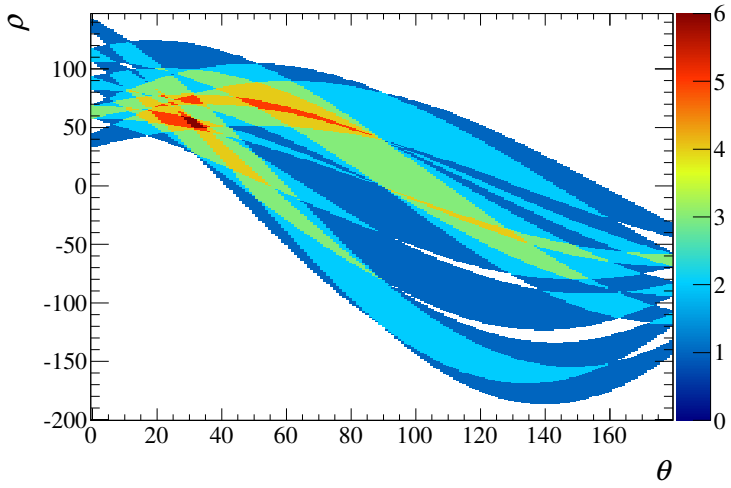

Figure 4.16: The reduced parameter space of the 2D cluster shown in Fig. 4.10.

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.

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

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

[6] 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