

An Algorithm for Particle Tracking and Analysis of Muons in the Main Injector Experiment ν -A (MINER ν A)

Senior Honors Thesis

Brandon Walker

Advisor: Professor Heidi Schellman

Northwestern University

May 7th, 2010

Abstract

An efficient algorithm for tracking particles is developed for use with the Main Injector Experiment ν -A (MINER ν A) at Fermi National Accelerator Laboratory. A technique using the Hough transform, energy weighted clustering of hits in planes, and standard least squares results in an accurate fit. The algorithm is particularly useful for tracking particles that follow a linear trajectory; e.g. rock muons from the MINOS beam. The program is capable of calculating a three-dimensional fit to such particles with standard deviations on the order of millimeters. This algorithm will hopefully be integrated into the final tracking code used in MINER ν A, and is a necessary first step in understanding neutrino nucleus interactions and neutrino oscillations.

Table of Contents

1. Introduction.....	4
2. The MINERvA Detector.....	6
3. The Hough Transform.....	8
4. ROOT Software.....	9
5. Finding Tracks.....	10
6. The NuMI beam is off axis?.....	13
7. Least Squares Best Fit.....	15
8. A Three Dimensional Track & Error Propagation.....	17
9. Conclusion.....	18
Acknowledgments.....	19
References.....	19
Appendix.....	20

1. Introduction

1.1 CP Violation and the Missing Antimatter.

Of particular importance in cosmology and particle physics today is the matter-antimatter asymmetry present in the universe. Why is it that everyday objects are composed of protons and electrons, and not antiprotons and positrons? The Big Bang theory supposes that equal amounts of matter and antimatter were created in the first instants of time, but it is blatantly obvious from astronomical observations that the universe is almost entirely composed of matter. During the birth of the universe, 13.8 billion years ago, something drastic happened—an unknown process or event nearly rid the universe of its antimatter component. One theory that may offer insight into this problem concerns Charge-Parity (CP) violation. CP violation is a phenomenon involving both a parity inversion and charge conjugation of a particle, and it is readily observed in the decay of neutral kaons. The long lived neutral kaon preferentially decays by means of positron emission $K_L^0 \rightarrow \pi^- + e^+ + \nu_e$ as opposed to electron emission $K_L^0 \rightarrow \pi^+ + e^- + \bar{\nu}_e$, despite the fact that a CP transformation converts the first product into the second¹. CP conservation would dictate that the two decays occur with equal probability, but this is not what is observed—consequently, this is an instance where CP is violated.

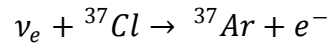
In search of explanations for the universe's preference for matter, CP violation is then a logical place to look. With the prior discussion of neutral kaons and the more recent confirmation of CP violation with B mesons (e.g. $B^0 \rightarrow K^+ + \pi^-$ occurs more frequently than $\bar{B}^0 \rightarrow K^- + \pi^+$), it is clear that CP violation occurs for weak interactions of quarks. Unfortunately, experiments have shown that CP violation from the weak interactions of quarks cannot account for all the matter-antimatter asymmetry in the universe¹. One proposed mechanism to account for this is CP violation with leptons. For example, if neutrinos had the ability to oscillate between flavors, then it is possible that CP violation in the lepton sector might prefer $\nu_e \rightarrow \nu_\mu$ to $\bar{\nu}_e \rightarrow \bar{\nu}_\mu$. A few theories propose that the addition of CP violation with leptons might be able to finally explain the reason why particle physicists aren't composed of antimatter. One may ask then, "Can an electron neutrino turn into a muon neutrino?"

1.2 Neutrino Oscillations.

The story of neutrino oscillations begins with the sun. Fueled by fusion through the *pp chain*, the sun is a prodigious supplier of neutrinos in the solar system. The route which the sun takes to go from hydrogen fuel to alpha particles and photons is a fairly complicated procedure, but the point of the matter is that electron neutrinos are copiously produced from this pp chain. In effect, trillions of solar neutrinos pass through the human body every second. Luckily for human beings, a constant barrage of

neutrinos is nothing to worry about, as they *rarely* interact with matter. Neutrinos are leptons that lack any charge, thus cannot interact via strong interactions or electromagnetic interactions. They can only interact via the weak force (and *gravitationally*, as we will see)².

In 1968, the first experiment to count solar neutrinos produced from the sun took place³. Deep underground in the Homestake mine in South Dakota, an enormous tank of chorine was placed to detect solar neutrinos. The experiment had to take place in an extremely shielded environment to prevent background radiation from altering the final results—thus, a mineshaft was as a good a place as any to start. A chlorine based detector was used in order to study the reaction in which a solar electron neutrino interacts with a chorine atom to produce an argon atom and electron¹:



Strangely, when all the argon atoms were counted at the end of the experiment, there was only one third of the predicted amount. Similar results have been found in more recent years by the Super-Kamiokande (SuperK) experiment using a water-based detector and at the Sudbury Neutrino Observatory (SNO) using a heavy water (D₂O) detector. Thus exists the solar neutrino problem—where did all the solar electron neutrinos go?

In the late 1960's, a theory was proposed to explain why the sun was not observed to be producing as many neutrinos as predicted. The solution was that all of the neutrinos from the sun were being produced, but they just weren't being detected. The Homestake mine experiment was only able to detect electron neutrinos, so the discrepancy could be explained if electron neutrinos could somehow transform into the other flavors of neutrinos (e.g. $\nu_e \rightarrow \nu_\mu$ or $\nu_e \rightarrow \nu_\tau$).

This notion of neutrino oscillation is built on quantum mechanics' description of mixed states. Suppose that the three neutrino flavors $\nu_e, \nu_\mu,$ and $\nu_\tau,$ were each actually a composition of three other orthogonal states. More formally, if the flavor states can transform into one another, then they are prohibited from being eigenstates of the free particle Hamiltonian. Each flavor state is instead a linear combination of the *mass* eigenstates $\nu_1, \nu_2,$ and ν_3 (why these are called “mass” eigenstates will be explained shortly)⁴.

Griffiths¹ offers a compelling simplification of the idea behind neutrino oscillation. Supposing for a minute that the world were only composed of two neutrinos ν_e and $\nu_\mu,$ the mass eigenstates could be written as a function of a mixing angle θ :

$$\nu_1 = \cos \theta \nu_\mu - \sin \theta \nu_e \text{ and } \nu_2 = \sin \theta \nu_\mu + \cos \theta \nu_e$$

The probability of an electron neutrino converting to a muon neutrino is then:

$$P_{\nu_e \rightarrow \nu_\mu} = \left[\sin 2\theta \sin \left(\frac{E_2 - E_1}{2} t \right) \right]^2$$

where the energy difference $E_2 - E_1$ is proportional to the difference of the masses squared $m_2^2 - m_1^2$. Consequently, if the neutrinos observed in experiments are actually a linear combination of orthogonal eigenstates with different masses, then neutrino flavors are not constant and they should oscillate between one another over time.

In 2002, SNO published experimental results confirming that solar neutrino oscillation is a real phenomenon. The conclusions from SNO and other experiments in the past 20 years have created an intense drive for neutrino oscillation experiments. Consequently, the Main Injector Neutrino Oscillation

Search (MINOS) began operating at Fermi National Accelerator Laboratory (Fermilab) in 2005. This experiment shoots an intense beam of muon neutrinos from the Near Detector at Fermilab to the Far Detector in the Soudan Iron Mine in Minnesota—a distance of 735 kilometers⁵. Experimental results from MINOS show that approximately half of the expected muon neutrinos were detected at the Far Detector, implying that many of the original muon neutrinos transformed to a different flavor during their journey.

1.3 MINERvA

The Main Injector Experiment v-A (MINERvA) was designed to improve the accuracy of neutrino oscillation experiments like MINOS by studying neutrino-nucleus interactions. The proper time required for a neutrino to oscillate into a different flavor has been measured to be approximately 400km/GeV. With engineering and detector design taken into consideration, accelerator based neutrino oscillation experiments are constrained to operate with neutrino energies greater than 0.6 GeV and with a maximum distance of 1000km between source and detector (referred to as the *baseline*). Consequently, the neutrino energies that are used in these experiments must lie between 0.6 and 2.5 GeV. Unfortunately, the transition from elastic to inelastic scattering takes place precisely in this energy regime, and the physics of this transition area with respect to neutrinos is poorly understood⁶. Because of this deficit of knowledge, the 2004 American Physical Society study on neutrino physics stated that of utmost importance in neutrino physics was the “determination of the neutrino reaction and production cross sections required for a precise understanding of neutrino-oscillation physics and the neutrino astronomy of astrophysical and cosmological sources. Our broad and exacting program of neutrino physics is built upon precise knowledge of how neutrinos interact with matter...”. This is precisely what MINERvA hopes to accomplish. With MINERvA just recently beginning to take data, much of the code for analyzing the data has yet to be written. A fundamental prerequisite for finding any new neutrino physics with the data is a method for particle tracking. My algorithm and code is able to take the data from the detector and decide whether or not a particle track is present. It is particularly effective for detecting muon tracks and is capable of finding a three dimensional fit to the tracks. Along the way, I was also able to show that the neutrino beam enters the MINERvA detector with a downward slope of approximately three degrees.

2. The MINERvA Detector

The MINERvA detector is located 106 meters underground in the NuMI Near Detector Hall at Fermilab. It is placed upstream of the MINOS near detector. The highest intensity neutrino beam in the world, the Neutrinos at the Main Injector (NuMI) beamline, runs right through the MINERvA detector. The NuMI beam is actually angled downward at 3.3 degrees relative to the main axis of the detector⁷. I take this main axis of the detector, the axis in the direction of travel of an entering particle, to be the z-axis. MINERvA is largely a fully active detector, with the inner regions of the detector being

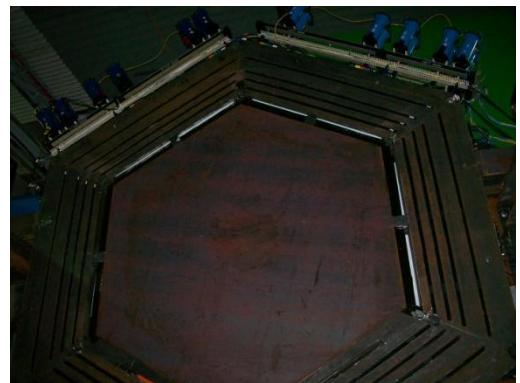
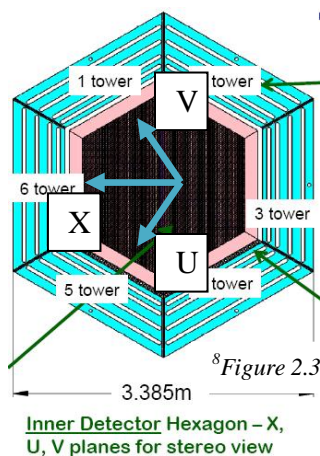
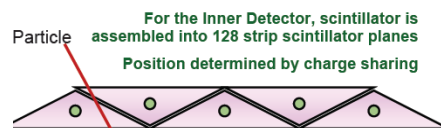


Figure 2.1

⁸Figure 2.2



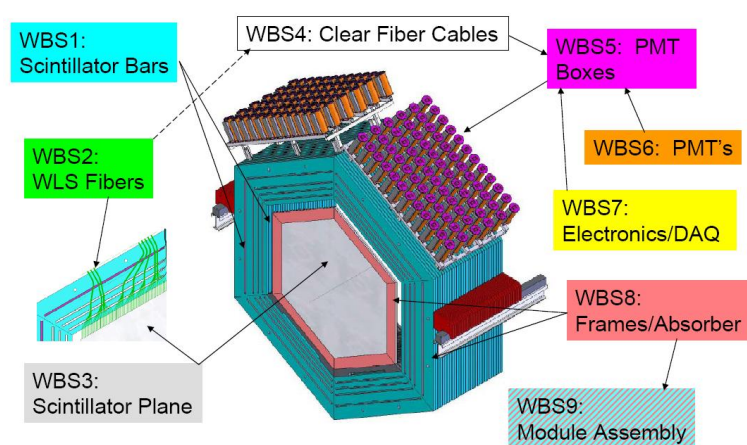
composed of triangular strips of plastic scintillator. The detector is composed of hexagonal *planes* standing transverse to the z-axis, with each plane being surrounded by a hexagonal outer detector. All of the planes combine to form the inner detector, while the outer detector is composed of the hexagonal outer shell. Each plane is formed by 128 overlapping scintillator strips. All of the strips are oriented the same way within a plane, but not necessarily with respect to the strips of other planes. The planes come in three orientations, X, U, and V. The orientation of these planes is shown in the figure to the left. The z-axis is into the page, the V plane is rotated 60 degrees clockwise from the X plane, while the U plane is



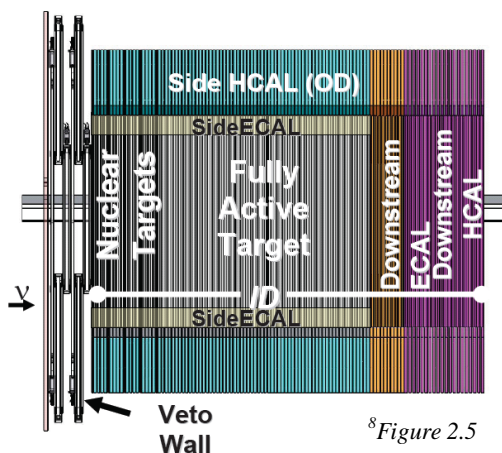
rotated 60 degrees counter-clockwise from the X plane. It should be noted that the length of any given scintillator strip runs perpendicular to the arrows in the figure, so that strip positions correspond to measurements in the corresponding plane. When considering strip positions within an X-plane, one is looking at measurements in the “X-view,” with similar terminology for the “U-view” and “V-view.” Two planes are attached to form a *module*, one of the planes always being an X-plane, and the other plane alternating between U-planes and V-planes. For example, the planes composing 3 modules might be in the following order: X, U, X, V, X, U, etc.

Each of the triangular scintillator strips has a Wave Length Shifting (WLS) fiber running through a center hole. The light that is emitted by the scintillating plastic is sent down the length of the strip via the WLS fibers, through optical connectors to clear fibers, and eventually to a photomultiplier tube (PMT). The active inner detector is surrounded by the outer detector, which is divided into six parts. Lead alloy absorbers act as electromagnetic calorimeters (ECALs) and surround the inner detector volume. Steel hadronic calorimeters (HCALs) surround the ECALs.

Data is collected in $10\mu\text{s}$ intervals, referred to as *gates*. Each gate is then split up into time *slices*. The code that generates slices within a gate takes small time intervals with a large amount of data to be valid slices that most likely contain events (particle interactions). The MINERvA electronics read out a photoelectron (PE) count. If the electronics read out a large enough PE count after calibration for a particular strip, then we say there was a *hit* in that strip. Consequently, each hit has a



⁸Figure 2.4



⁸Figure 2.5

corresponding module position, strip position, plane type (hence the hit is in a particular view), and calibrated PE count. Some of these variables can be seen in the event display below.

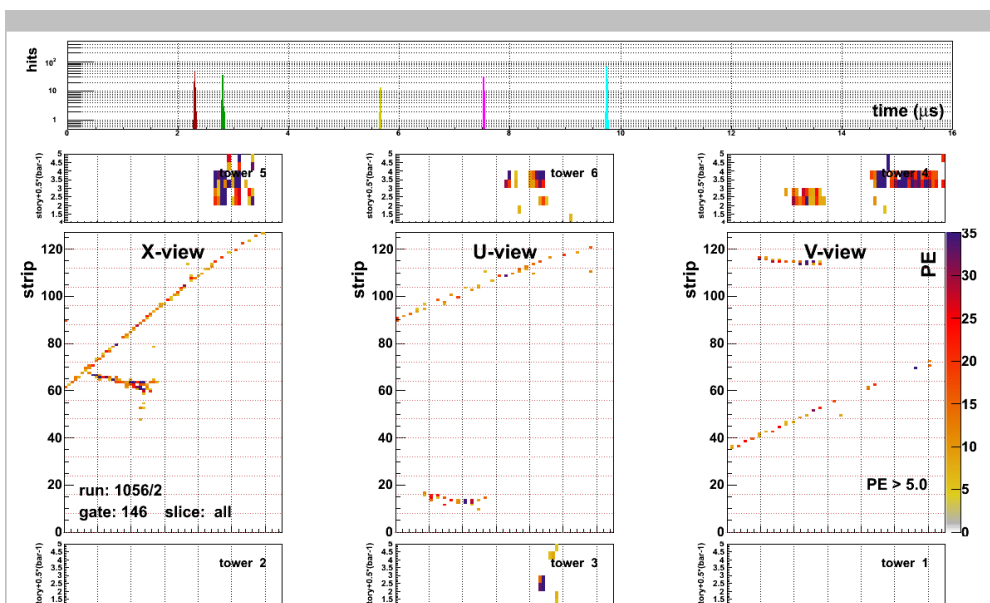


Figure 2.6

3. The Hough Transform

In the 1960's, the Hough transform was introduced as a means for detecting patterns and shapes from a set of points⁹. The Hough transform assumes that a pattern or shape (e.g. a line) will have a definite set of parameters. A pattern that spans a large region of the image space is then mapped to a parameter space, where the problem is reduced to finding maxima in the parameter space.

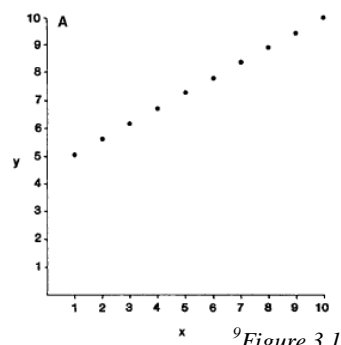
For example, consider the set of points shown in *Figure 3.1*. Each of these points lies in the image space and has a corresponding x-coordinate and y-coordinate. It looks as if these points all lie along the same line, so we consider the relationship each point has to the parameters of a line:

$$y - mx - b = 0$$

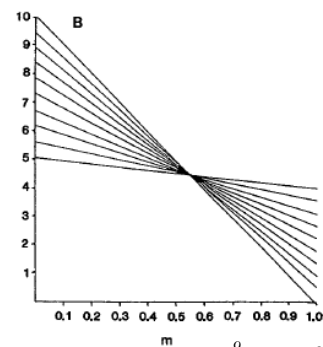
where m and b stand for the slope and y-intercept, respectively. In effect, each point with coordinates (x,y) corresponds to an infinite number of lines in parameter space. By plugging in slopes, leaving x and y fixed, a series of y-intercepts are generated:

$$b = h(m) = y - mx$$

The function $h(m)$ is then a line in parameter space, and this process is often referred to as back-projection of the image point. Each point in image space maps to a line in parameter space. The intersection of two lines corresponds to two collinear points in parameter space.

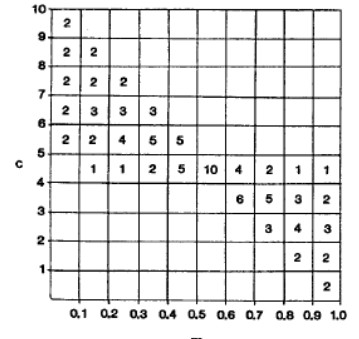


⁹Figure 3.1



⁹Figure 3.2

If the points in image space are not all exactly collinear, but instead lie very nearly on a line, then the back-projection of the image points will not result in one intersection in parameter space. For this reason, the image space is often mapped to an accumulator space with a finite number of pixels or bins. For this example, a two-dimensional histogram suffices, and is shown in *Figure 3.3*. Computationally, the problem of identifying a line in image space has been reduced to finding the maximum bin in the two-dimensional histogram in accumulator space.



⁹Figure 3.3

The careful reader will notice a problem with back-projection as it has been defined—for image points forming lines with near vertical slopes, the resulting slope parameter is unbounded. As the line nears a vertical slope, ever larger values of m must be passed to $h(m)$ for increasingly smaller increments in parameter space. As the near-term goal of my algorithm has been to track muons from the NuMI beam, this has not been a major concern—most muons are traveling more or less in the in the z-direction, and even particles scattering quasi-elastically tend to not deviate by more than 70 degrees from the z-axis of the detector. However, if my algorithm were to be implemented for a wider range of particles with more chaotic trajectories, a simple solution might be to parameterize the line in image space by using polar coordinates. Letting r represent the least distance from the origin to the line, and θ represent the angle from the x-axis to the vector from the origin to the point of closest approach, each point then has the following relationship to the parameters:

$$y = \frac{-\cos \theta}{\sin \theta} x + \frac{r}{\sin \theta}$$

and the back-projection could be performed with the function:

$$r = h(\theta) = y \sin \theta + x \cos \theta$$

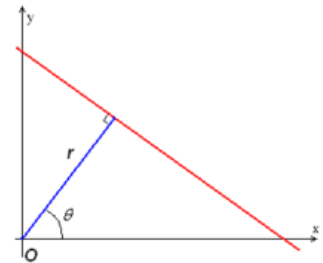


Figure 3.4

4. ROOT Software

Before any effort can be made to actually find a particle's trajectory, there must be data and a way for an algorithm to access the data. In order to efficiently package and provide access to all the necessary variables for neutrino physics, a common tool in high energy physics, called ROOT, was utilized. This object-oriented program was developed at the European Organization for Nuclear Research (CERN) and is a framework for data processing¹⁰. One of the key reasons ROOT is commonly used in high energy physics is its ability to provide quick access to extremely large amounts of data. ROOT uses a data structure called a *tree*, which allows access to smaller data members called *branches* and *leaves*. In our case, the hits in each plane, along with their calibrated photo electron count, correspond to leaves in a MINERvA tree.

ROOT is a framework for data processing in the sense that it provides an enormous amount of infrastructure and built in utilities to the user. For example, ROOT offers: a built in histogram utility, a graphical user interface (GUI), 2 dimensional graphics, input/output, a command line interpreter, and a

script processor. The command line interpreter is a C++ interpreter called CINT, from which source and header files written in C++ can be uploaded and executed. Additionally, CINT allows a quick and easy way for the user to run the tracking algorithm multiple times and on different gates.

5. Finding Tracks

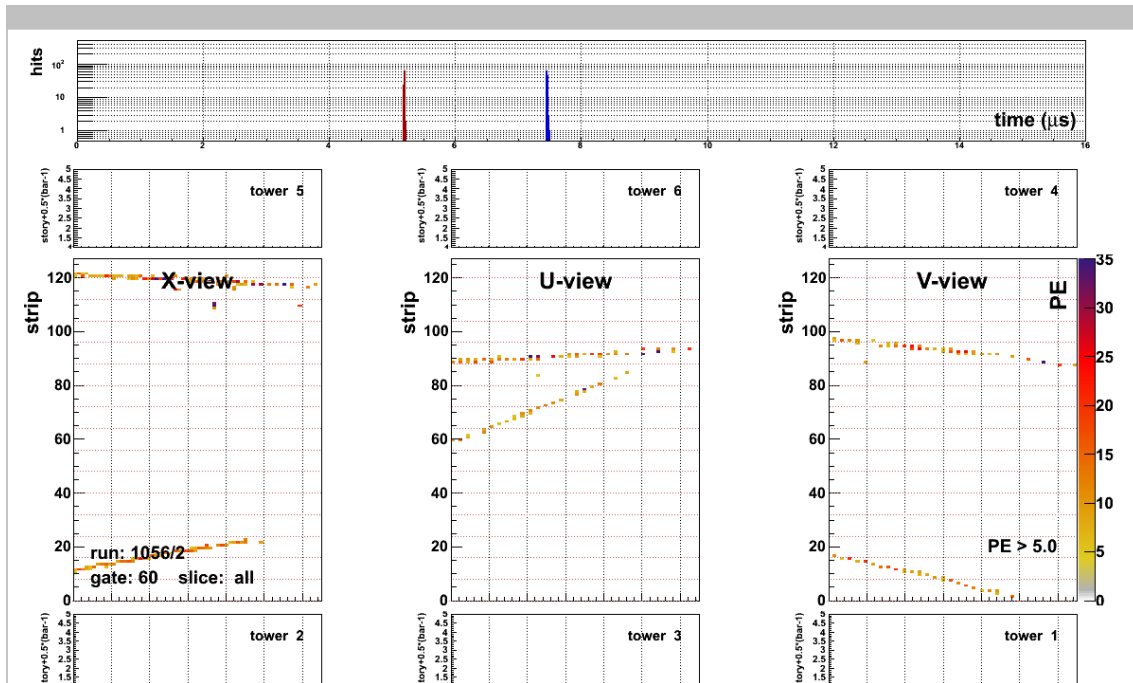


Figure 5.1

5.1 Hough Transform Applied to MINERvA

My algorithm is written in C++ as a series of functions. Each of these functions is able to interact with the C++ code already written for MINERvA to display the events in a user friendly form. All of this code is implemented within ROOT, and utilizes much of ROOT's histogram functionality, I/O, and graphics displays. Before I began writing the tracking algorithm, MINERvA's code contained a function `Pause_Event_Display()` that displayed the strip position of the hits as a function of module number in a 2-D histogram. This is separated into the three views of the inner detector, X, U, and V, depending on what plane the hit took place in. The hit is displayed in a particular bin within the 2-D histogram and is color coded based on the calibrated photo electron count of that hit. If the user would like to make a PE cut (i.e. only display hits that are greater than or equal to the PE cut), the PE cut is displayed in the lower right corner of the V-view. The NuMI beam run and subrun are displayed in the lower left corner of the X-view with the format `<Run>/<Subrun>`. The gate number and slice are displayed below the NuMI run number. It is oftentimes the case where one wants to view all of the slices of an entire gate at once, in which case slice has `<all>` displayed. Above and below each view are the towers of the outer detector. The outer detector is not used in my algorithm, but it is useful to know what this part of the display corresponds to. Above the inner and outer detector displays is a plot of hits versus time for the entire gate. Note that the gate is only 10 μ s long, so the hits will always be shown

within the first 10 μ s of the plot. The slices of the gate are short time intervals around each of the spikes in the plot. An example is shown in *Figure 5.1* above for Run 1056, Subrun 2, Gate 60, slice All.

To the human eye, it seems clear that there are two particle tracks moving down the detector in the z-direction in *Figure 5.1*. My algorithm teaches the computer how be clever enough to come to the same conclusions. The function that is given the task of finding the tracks is called `track_display()` (somewhat of a misnomer, `track_display()` doesn't actually display the tracks, it merely finds them). As input variables, `track_display()` takes the gate number, slice, view, and a PE cut from the MINERvA event display. `track_display()` also has access to the MINERvA tree in ROOT, thus knows about each individual hit within the gate and its corresponding variables.

The maximum number of tracks that `track_display()` looks for is set to five, but this number could easily be modified based on the statistics of the number of tracks per gate. The MINERvA event display does not have access to the exact position of hits in millimeters, so the strip number and module number are used to locate a hit. Each strip is approximately 16.5mm across, and each module is separated by approximately 40mm, so the appropriate conversions allow `track_display()` to calculate a two dimensional position of the hit in the plane with respect to the z-axis.

A Hough transform is used to find the most likely track candidates. The hit coordinate with a strip position and z-position is back-projected to a slope-intercept parameter space. The parameter space is actually a 2-D histogram that functions as an accumulator space. For example, for a hit in the X-view, the function that accomplishes back-projection is:

$$b = h(m) = x - mz$$

where b is the x-intercept at the $z = 0$ position of the detector, x is the strip position of the hit in mm, m is the slope, and z is the z-position of the module the hit took place in. It is important to note that we are centering the origin of our coordinate system such that the center of each plane corresponds to the origin in the X, U, and V views (i.e. $x = 0$ at strip 64 in an X-plane). We pass slopes ranging from -5 to +5 to $h(m)$, corresponding approximately to a 78.7° window above or below the z-axis. The range of intercepts allowed in accumulator space is the width of 3 detectors (i.e. 384 strips), to account for particles that are scattered at large angles. The bins in accumulator space have a slope width of 0.025 ($\cong 0.39^\circ$ for small slopes), and an intercept height of 1 strip (i.e. 16.5mm).

The `track_display()` function is first called for the X-view—the U and V views will need to use the information from the tracks in the X-view later on. A global variable, called `ntracksX`, is used to store the number of tracks found in the X-view. When `track_display()` is called for the X-view, it cycles through all of the hits in the given gate and slice from the MINERvA tree object. Various characteristics of the hits need to be checked before the hit is used in back-projection. `track_display()` checks that a hit is in the proper view (the X-view in this case), that a hit is in the proper time slice, and that the calibrated PE count of the hit is not less than the PE cut.

After the hit has passed the aforementioned tests, it is assigned a weight based on its calibrated PE count. To prevent one hit with an extremely high PE count from distorting the data, anything above 15 PE is counted the same. The point is that hits with a higher PE count may be more accurate representations of the particle's location, but we do not want to end up basing the entire track on one heavily weighted hit. For any hit below 15 PE, the weight is simply the PE count. Finally, we take the

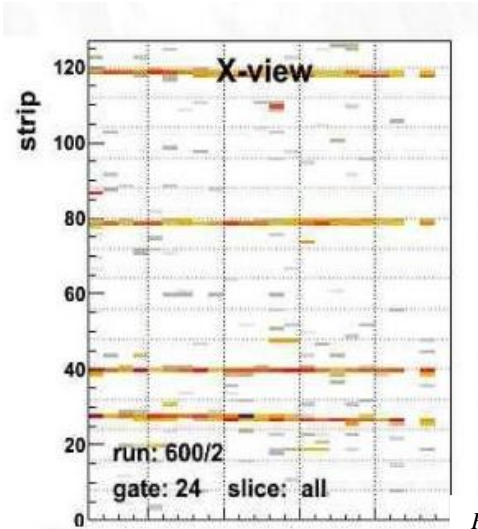


Figure 5.2

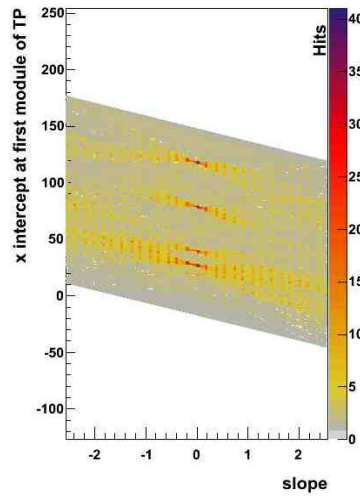
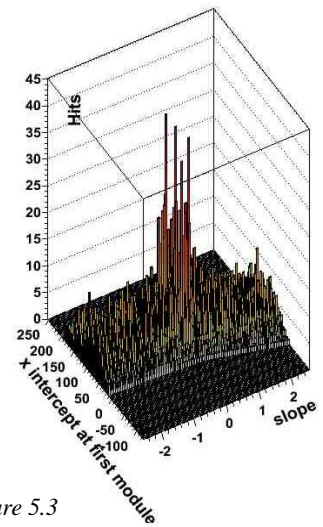


Figure 5.3



logarithm of this weight to be the final weight used, as the logarithm makes sure that the low PE hits are not disregarded. With the weight for the hit being calculated, the hit is back-projected using $h(m)$ into accumulator space, and the appropriate bins are filled in the 2-D histogram with the corresponding weight (e.g. Figure 5.3 is the accumulator space for Figure 5.2).

After cycling through all of the hits in the slice and filling the accumulator histogram, `track_display()` looks for the maximum bin in the histogram. If the maximum bin content is less than a predefined minimum (which I define to be 25), the program returns and prints out that no tracks are present in the slice. If the maximum bin content is greater than the predefined minimum, then this track candidate is accepted as a valid track. The program then marks which hits are assigned to this track by using the slope and intercept parameter from the maximum bin, and calculating whether or not a hit is within a small window around the track. It does this by cycling through all of the hits, and finding the x-intercept for each hit using the accepted slope from the maximum bin. If the calculated x-intercept is within a two strip window of the accepted intercept from the maximum bin, then this hit is assigned to the track. This assignment is stored in a global array outside of `track_display()` so that it may be used by other functions. After finding one valid track, the program cycles through all of the hits again and fills the accumulator histogram with unmarked hits. It once again looks for the highest bin in the accumulator histogram and checks whether such a bin corresponds to a valid track. This time around, however, hits that were already assigned to the first track cannot be reassigned to the second track. This entire process repeats for up to five tracks, marking the hits used along the way. The accepted track slopes and intercepts in the X-view are stored as global variables.

After `track_display()` has been called for the X-view, it is called one more time to try to find tracks in the Y-view (the Y-view corresponding to the traditional y-axis in a Cartesian coordinate system). The detector does not have a plane that measures hit positions in the Y-view, so my algorithm uses the X-view to correct the U and V-views to the Y-view with the following geometrical relations:

$$y = \frac{2}{\sqrt{3}}v - \frac{1}{\sqrt{3}}x \quad y = \frac{-2}{\sqrt{3}}u + \frac{1}{\sqrt{3}}x$$

Aside from the reassignment of each hit in the U-view and V-view, `track_display()` basically performs the same operations. It is important to note, however, that data from both the U-view and V-view are

used simultaneously to find tracks in the Y view. An X-plane occurs in every module, but the U and V planes alternate modules, thus both views must be used to provide sufficient data for track finding. After performing a Hough transform on the Y-view data, we once again use the accumulator histogram to find the maximum bin, look for valid track candidates, and assign hits to the proper track in the Y-view. Up to five tracks are found in the Y-view. The accepted track slopes and intercepts in the Y-view are stored as global variables.

5.2 Verify the Tracks

After applying the Hough method to the MINERvA data, I wanted to verify visually that the results of track_display() were reasonable. In order to do this, I defined a new function called draw_tracks() to use ROOT's built-in function classes to draw the accepted tracks on the event display. In order to do this, I used the following relationships to convert the Y-view slopes and intercepts to back to the U-view and V-view:

$$v = \frac{\sqrt{3}}{2}y + \frac{1}{2}x \quad u = \frac{-\sqrt{3}}{2}y + \frac{1}{2}x$$

The resulting plots matched extremely well, as can be seen by applying draw_tracks() to *Figure 5.1*:

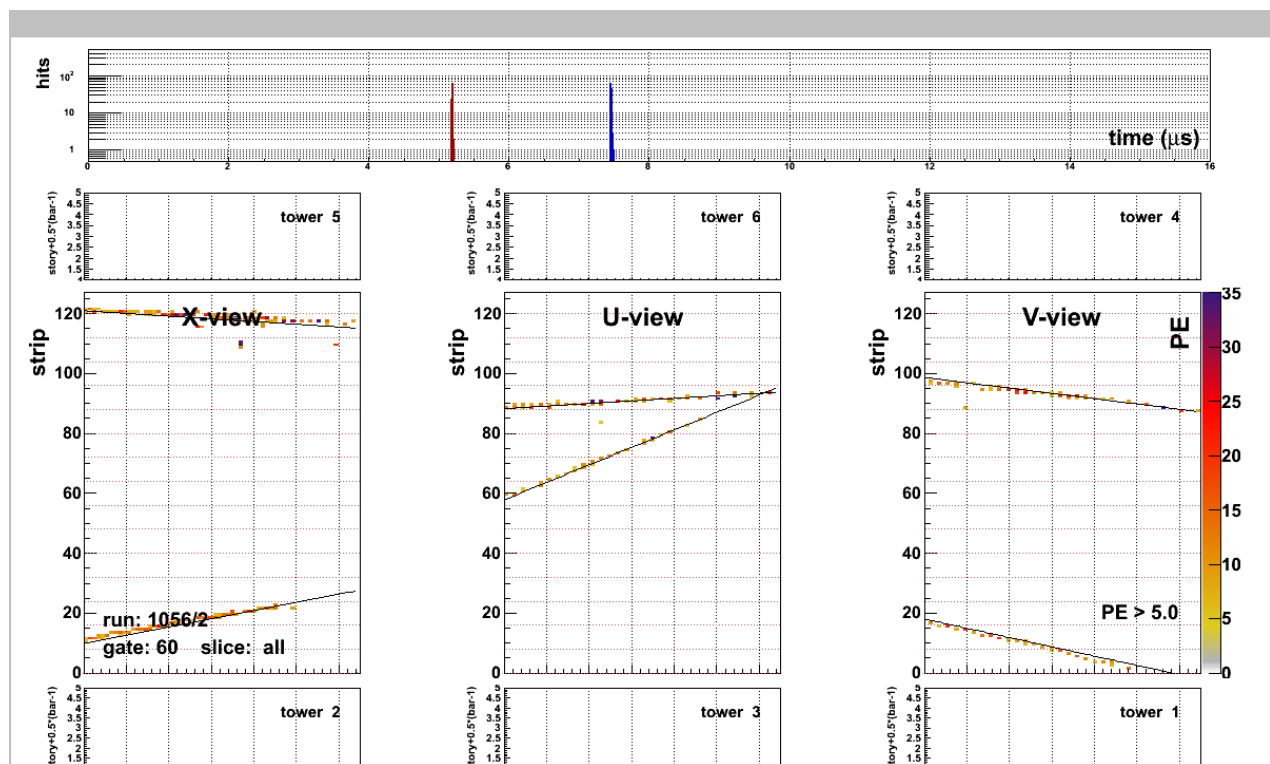
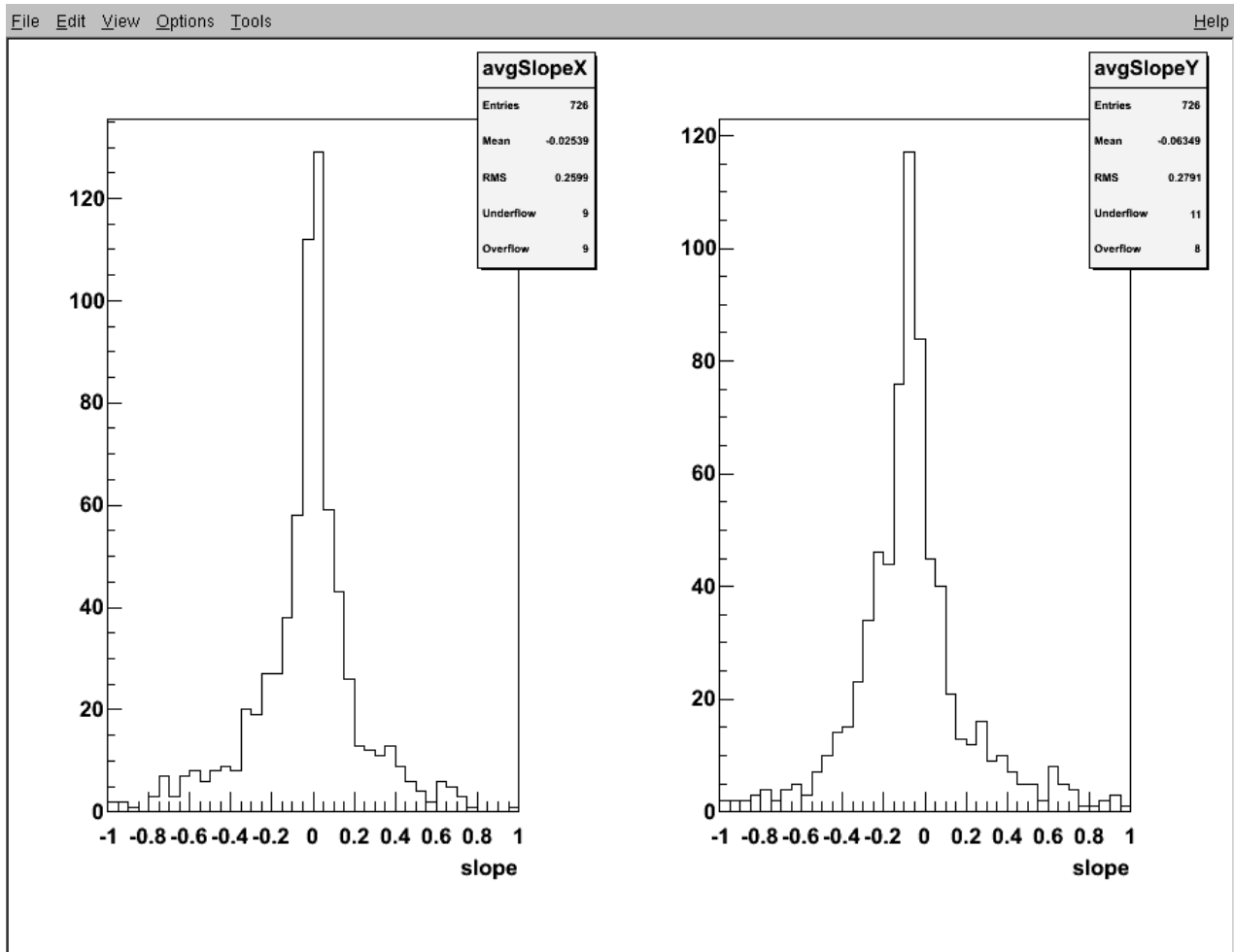


Figure 5.4

6. The NuMI beam is off axis?

It was stated earlier that the NuMI beam entered the MINERvA detector with a 3.3° downward slope. Consequently, it should be possible to observe this downward slope as an effect on the average

slope in the Y-view over many events. I designed a function, called `AnalyzeTracks()`, to look for this effect. Some precaution was necessary, however, as we only wanted to use tracks whose average slopes in the Y-view would otherwise come out to be zero if the NuMI beam were parallel to the z-axis of the detector. If we only consider slices in which there are two tracks and furthermore that they intersect at a common point, it is quite likely that we are seeing a quasi-elastic interaction of a muon neutrino and a neutron $\nu_\mu n \rightarrow \mu^- p^+$. In such an interaction the neutron can be treated as stationary and the incoming neutrino is not observable by MINERvA. The outgoing muon and proton should form two tracks diverging from the same vertex. Momentum in the X-view and Y-view should be approximately conserved in each event, but over many events it should be observable that the average X-view slope is zero and the average Y-view slope is -3.3° . By using the slopes and intercepts of each track from the `tracks_display()` and vector calculus, I was able to calculate a measurement of closest approach between two tracks in three dimensions. `AnalyzeTracks()` cycled through all of the slices within 4000 gates of a few NuMI subruns and histogrammed the resulting average slopes in the X-view and Y-view. The result was comforting—the program found an average Y-view slope of $-3.63 \pm 0.59^\circ$ and an average X-view slope of $-1.45 \pm 0.55^\circ$. These results are not perfect nor error free, but the data sample over 4000 gates was not large; only 726 suitable events were found. It is also important to notice that not every two prong event is necessarily a quasi-elastic event described above. A more rigorous analysis using each particle's momentum and energies would be much more accurate. Note that more details about the tracks used in this calculation (i.e. χ^2 of the tracks in the X-view, U-View, and V-views) are included in Appendix B.



7. Least Squares Best Fit

In order to find a best fit to the tracks found by the Hough transform, I wrote a function called `cluster_best_fit()`. The name of this function suits it well, as this function clusters all of the hits in one plane together. Moreover, I used an energy weighted average of the hits in each plane to find a single average hit per plane. The `cluster_best_fit()` function first cycles through every hit in the current time slice of a gate, summing up the calibrated PE count in each module for a given track and view. It then finds the energy weighted average hit in a plane using the calibrated PE count of the hit as the energy.

To find a best fit line to the tracks using the energy weighted average hit in `cluster_best_fit()`, I use ordinary least squares as described in Bevington¹¹ and Numerical Recipes¹². The goal is to find the following parameters for each track in its corresponding view:

$$x = \alpha_x + \beta_x z \quad u = \alpha_u + \beta_u z \quad v = \alpha_v + \beta_v z$$

A weighted fit is used by assuming the weighting of each hit is equal to the variance. This variance of a measured hit corresponds to the experimental error measured for MINERvA—a standard deviation of 4mm for any given hit measured in a plane. Thus for any given hit i , $\sigma_i = \sigma = 4mm$.

In order to find the parameters for a track, we minimize the weighted sum of the residuals χ^2 . All sums are from 1 to N unless otherwise stated. Beginning with the X-view¹¹,

$$\chi_x^2 = \sum \frac{1}{\sigma_i^2} (x_i - \alpha_x - \beta_x z_i)^2 = \frac{1}{\sigma^2} \sum (x_i - \alpha_x - \beta_x z_i)^2$$

We define a term Δ to help simplify the equations:

$$\Delta = \sum \frac{1}{\sigma_i^2} \sum \frac{z_i^2}{\sigma_i^2} - \left(\sum \frac{z_i}{\sigma_i^2} \right)^2 = \frac{1}{\sigma^4} \left(N \sum z_i^2 - \left(\sum z_i \right)^2 \right)$$

The parameters are found by setting:

$$\frac{\partial}{\partial \alpha_x} \chi_x^2 = 0 \quad \frac{\partial}{\partial \beta_x} \chi_x^2 = 0$$

Resulting in the parameters:

$$\alpha_x = \frac{1}{\Delta} \left(\sum \frac{z_i^2}{\sigma_i^2} \sum \frac{x_i}{\sigma_i^2} - \sum \frac{z_i}{\sigma_i^2} \sum \frac{z_i x_i}{\sigma_i^2} \right) = \frac{1}{\sigma^4} \frac{1}{\Delta} \left(\sum z_i^2 \sum x_i - \sum z_i \sum z_i x_i \right)$$

$$\beta_x = \frac{1}{\Delta} \left(\sum \frac{1}{\sigma_i^2} \sum \frac{z_i x_i}{\sigma_i^2} - \sum \frac{z_i}{\sigma_i^2} \sum \frac{x_i}{\sigma_i^2} \right) = \frac{1}{\sigma^4} \frac{1}{\Delta} \left(N \sum z_i x_i - \sum z_i \sum x_i \right)$$

Similarly, for the U-view and V-view the parameters are:

$$\alpha_u = \frac{1}{\Delta} \left(\sum \frac{z_i^2}{\sigma_i^2} \sum \frac{u_i}{\sigma_i^2} - \sum \frac{z_i}{\sigma_i^2} \sum \frac{z_i u_i}{\sigma_i^2} \right) = \frac{1}{\sigma^4} \frac{1}{\Delta} \left(\sum z_i^2 \sum u_i - \sum z_i \sum z_i u_i \right)$$

$$\beta_u = \frac{1}{\Delta} \left(\sum \frac{1}{\sigma_i^2} \sum \frac{z_i u_i}{\sigma_i^2} - \sum \frac{z_i}{\sigma_i^2} \sum \frac{u_i}{\sigma_i^2} \right) = \frac{1}{\sigma^4} \frac{1}{\Delta} \left(N \sum z_i u_i - \sum z_i \sum u_i \right)$$

$$\alpha_v = \frac{1}{\Delta} \left(\sum \frac{z_i^2}{\sigma_i^2} \sum \frac{v_i}{\sigma_i^2} - \sum \frac{z_i}{\sigma_i^2} \sum \frac{z_i v_i}{\sigma_i^2} \right) = \frac{1}{\sigma^4} \frac{1}{\Delta} \left(\sum z_i^2 \sum v_i - \sum z_i \sum z_i v_i \right)$$

$$\beta_v = \frac{1}{\Delta} \left(\sum \frac{1}{\sigma_i^2} \sum \frac{z_i v_i}{\sigma_i^2} - \sum \frac{z_i}{\sigma_i^2} \sum \frac{v_i}{\sigma_i^2} \right) = \frac{1}{\sigma^4} \frac{1}{\Delta} \left(N \sum z_i v_i - \sum z_i \sum v_i \right)$$

Each hit that was used to find the parameters has an error associated to it, and this error must be reflected in the error of the parameters. Using propagation of errors, it can be shown that the error for any parameter f of a track in the X-view is related to the hits x_i by the formula¹²:

$$\sigma_f^2 = \sum \sigma_i^2 \left(\frac{\partial f}{\partial x_i} \right)^2$$

As a result, the errors in the parameters along with the covariance are¹²:

$$\begin{aligned} \sigma_{\alpha_x}^2 &= \frac{1}{\Delta} \sum \frac{z_i^2}{\sigma_i^2} = \frac{1}{\sigma^2} \frac{1}{\Delta} \sum z_i^2 \\ \sigma_{\beta_x}^2 &= \frac{1}{\Delta} \sum \frac{1}{\sigma_i^2} = \frac{1}{\sigma^2} \frac{N}{\Delta} \\ \sigma_{\alpha_x \beta_x}^2 &= -\frac{1}{\Delta} \sum \frac{x_i}{\sigma_i^2} = -\frac{1}{\sigma^2} \frac{1}{\Delta} \sum x_i \end{aligned}$$

The equations for the errors in the U-view and V-view are identical, except for the replacement of every “ x ” by a “ u ” or “ v .”

Consequently, error propagation can be used again to find the error involved with actually using the fit. The errors are:

$$\begin{aligned} \sigma_x^2 &= \sigma_{\alpha_x}^2 + \sigma_{\beta_x}^2 z^2 + 2\sigma_{\alpha_x \beta_x} z \\ \sigma_u^2 &= \sigma_{\alpha_u}^2 + \sigma_{\beta_u}^2 z^2 + 2\sigma_{\alpha_u \beta_u} z \\ \sigma_v^2 &= \sigma_{\alpha_v}^2 + \sigma_{\beta_v}^2 z^2 + 2\sigma_{\alpha_v \beta_v} z \end{aligned}$$

I used the parameter values for each track to plot the best fit lines on top of the MINERvA event display in ROOT. Applying the `cluster_best_fit()` function to the gate and slice in *Figure 5.1*, I come up with the following event display, shown in *Figure 7.1*. The fits are matched across views by color.

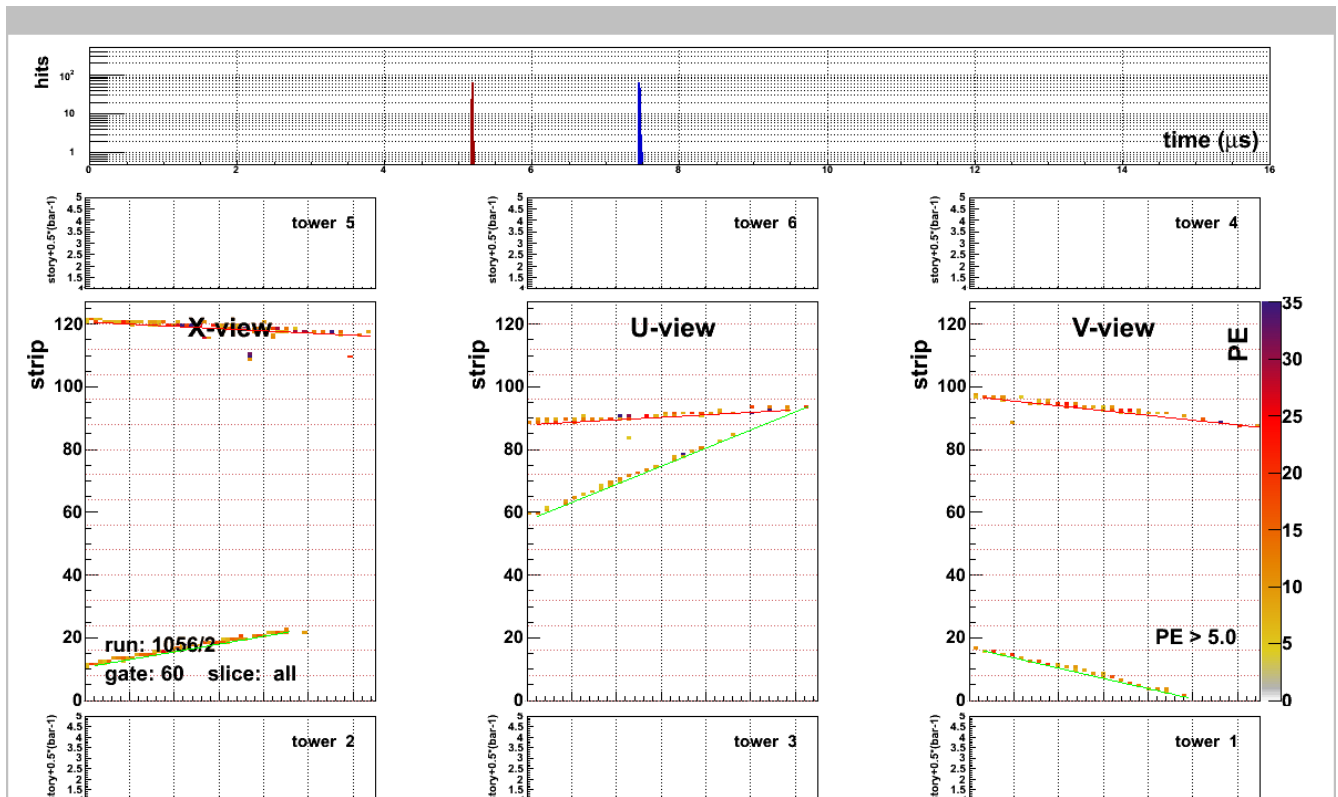


Figure 7.1

8. A Three Dimensional Track and Error Propagation

To find a full three-dimensional fit to each track, one needs to use error propagation and relate the U-view and V-views to the Y-view. Taking note of the fact that parameters from different views are not correlated and using the geometrical relationship described earlier, the Y-view parameters are as follows.

$$y = \frac{1}{\sqrt{3}}(\alpha_v + \beta_v z) - \frac{1}{\sqrt{3}}(\alpha_u + \beta_u z)$$

Therefore,

$$\alpha_y = \frac{1}{\sqrt{3}}\alpha_v - \frac{1}{\sqrt{3}}\alpha_u \quad \beta_y = \frac{1}{\sqrt{3}}\beta_v z - \frac{1}{\sqrt{3}}\beta_u z$$

The errors are then:

$$\begin{aligned} \sigma_{\alpha_y}^2 &= \frac{1}{3}(\sigma_{\alpha_v}^2 + \sigma_{\alpha_u}^2) \\ \sigma_{\beta_y}^2 &= \frac{1}{3}(\sigma_{\beta_v}^2 z^2 + \sigma_{\beta_u}^2 z^2) \\ \sigma_{\alpha_y \beta_y}^2 &= \frac{1}{3}(\sigma_{\alpha_v \beta_v}^2 + \sigma_{\alpha_u \beta_u}^2) \\ \sigma_y^2 &= \frac{1}{3}(\sigma_{\alpha_v}^2 + \sigma_{\beta_v}^2 z^2 + 2\sigma_{\alpha_v \beta_v} z + \sigma_{\alpha_u}^2 + \sigma_{\beta_u}^2 z^2 + 2\sigma_{\alpha_u \beta_u} z) \end{aligned}$$

The results of the fits found in *Figure 7.1* as printed out by ROOT are shown below. Two tracks were found in each view, thus the track indices range from 0 to 1. Distances are measured in millimeters for the parameters.

Calling track_display for view = 1. The number of slices is: 2. And pe_cut is set to 5. The current slice is: -1 track_display has finished and found 2 tracks.

Calling track_display for view = 2. The number of slices is: 2. And pe_cut is set to 5. The current slice is: -1

Chi2 on track 0 in X-view is 92.2545. Num modules used in track is 50.
 Chi2 on track 1 in X-view is 72.6113. Num modules used in track is 44.
 Chi2 on track 0 in U-view is 33.1781. Num modules used in track is 25.
 Chi2 on track 1 in U-view is 41.067. Num modules used in track is 22.
 Chi2 on track 0 in V-view is 64.5023. Num modules used in track is 26.
 Chi2 on track 1 in V-view is 54.3303. Num modules used in track is 23.

x-view alpha for track 0 is 919.808 with a standard deviation of 1.11899
 x-view beta for track 0 is -0.0290423 with a standard deviation of 0.000836924
 x-view covariance for track 0 is -0.000620804
 x-view alpha for track 1 is -900.821 with a standard deviation of 1.25617
 x-view beta for track 1 is 0.101515 with a standard deviation of 0.0011633
 x-view covariance for track 1 is 0.00108891

u-view alpha for track 0 is 377.072 with a standard deviation of 1.57454
 u-view beta for track 0 is 0.0325069 with a standard deviation of 0.00122486
 u-view covariance for track 0 is -0.000619709
 u-view alpha for track 1 is -132.489 with a standard deviation of 1.63211

u-view beta for track 1 is 0.23817 with a standard deviation of 0.00129068
u-view covariance for track 1 is -0.000207069

v-view alpha for track 0 is 530.835 with a standard deviation of 1.60542
v-view beta for track 0 is -0.0630844 with a standard deviation of 0.00113524
v-view covariance for track 0 is -0.000583808
v-view alpha for track 1 is -787.57 with a standard deviation of 1.76863
v-view beta for track 1 is -0.135088 with a standard deviation of 0.00155421
v-view covariance for track 1 is 0.00222987

y-view alpha for track 0 is 88.7753 with a standard deviation of 1.29827
y-view beta for track 0 is -0.0551896 with a standard deviation of 0.000964199
y-view covariance for track 0 is -0.000401172
y-view alpha for track 1 is -378.211 with a standard deviation of 1.38946
y-view beta for track 1 is -0.215501 with a standard deviation of 0.00116639
y-view covariance for track 1 is 0.000674268

9. Conclusion

The tracking algorithm composed of `track_display()` and `cluster_best_fit()` is able to accurately track muons from the NuMI beam entering the detector (oftentimes referred to as *rock* muons). A three dimensional fit can be found for such tracks with errors on the order of millimeters. While the method is particularly effective at tracking particles with a linear trajectory, a more advanced data processing technique using the Hough transform could theoretically be performed in the future to discover particles with nonlinear tracks (any particle whose track bends significantly in the vicinity of the MINOS magnetic field).

Additionally, the algorithm completes its task extremely quickly. The method's effect on the MINERvA event display code is hardly noticeable. This algorithm went through many different versions and what is seen today is the end product of many attempts to find an efficient way to track muons. This code is already on its way to being integrated into the final MINERvA tracking code, and opens up the door for future analysis of particle interactions requiring three dimensional trajectories.

Acknowledgments

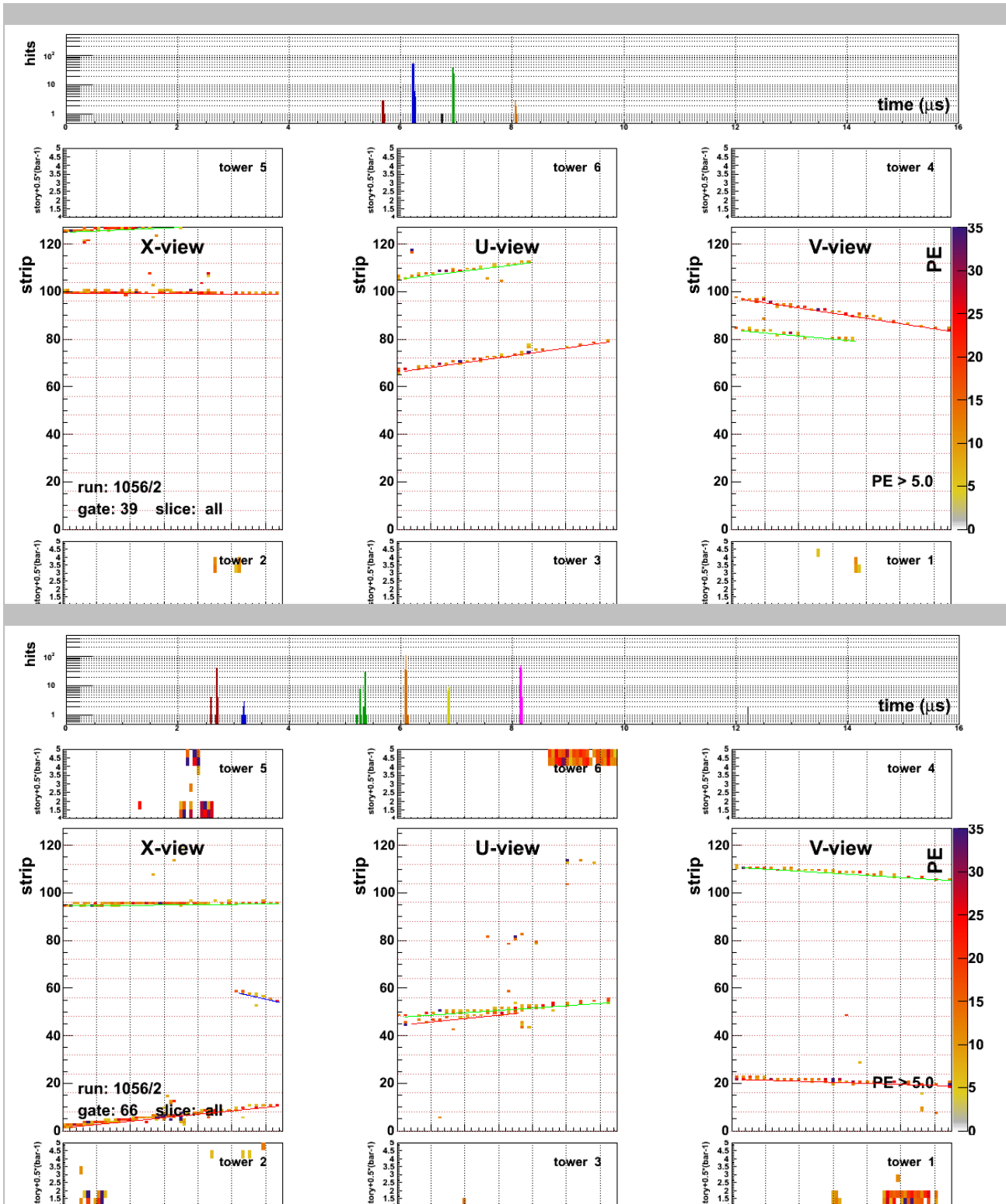
First, I would like to thank my advisor, Heidi Schellman, for providing me with a world of opportunities to get involved in high energy physics, for teaching me my first programming language, and for allowing me the chance to work side by side with particle physicists at Fermilab. Without her guidance, my undergraduate education would not be nearly as exciting as it has been. Thank you to André de Gouvêa for agreeing to be a second reader of this thesis, and for his *outstanding* lectures in classical dynamics. Thank you also to Bob Bradford and Dan Ruggiero of the University of Rochester for being wonderful mentors and a joy to work with at Fermilab during the construction phase of MINERvA. Finally, thank you to my girlfriend, Danielle, for being so understanding when I was spending countless hours 300 feet underground in a mine shaft exploring the wonders of neutrino physics. As it turns out, it is exceedingly difficult to get cell phone reception in such a place.

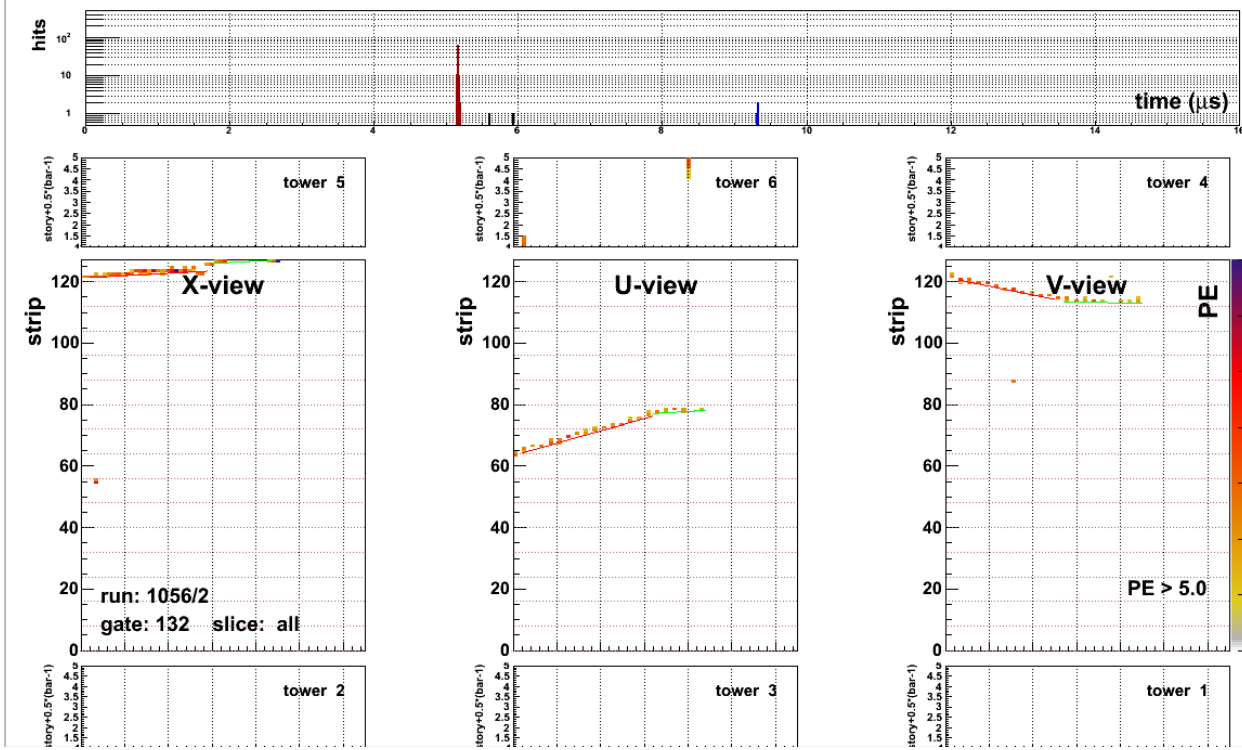
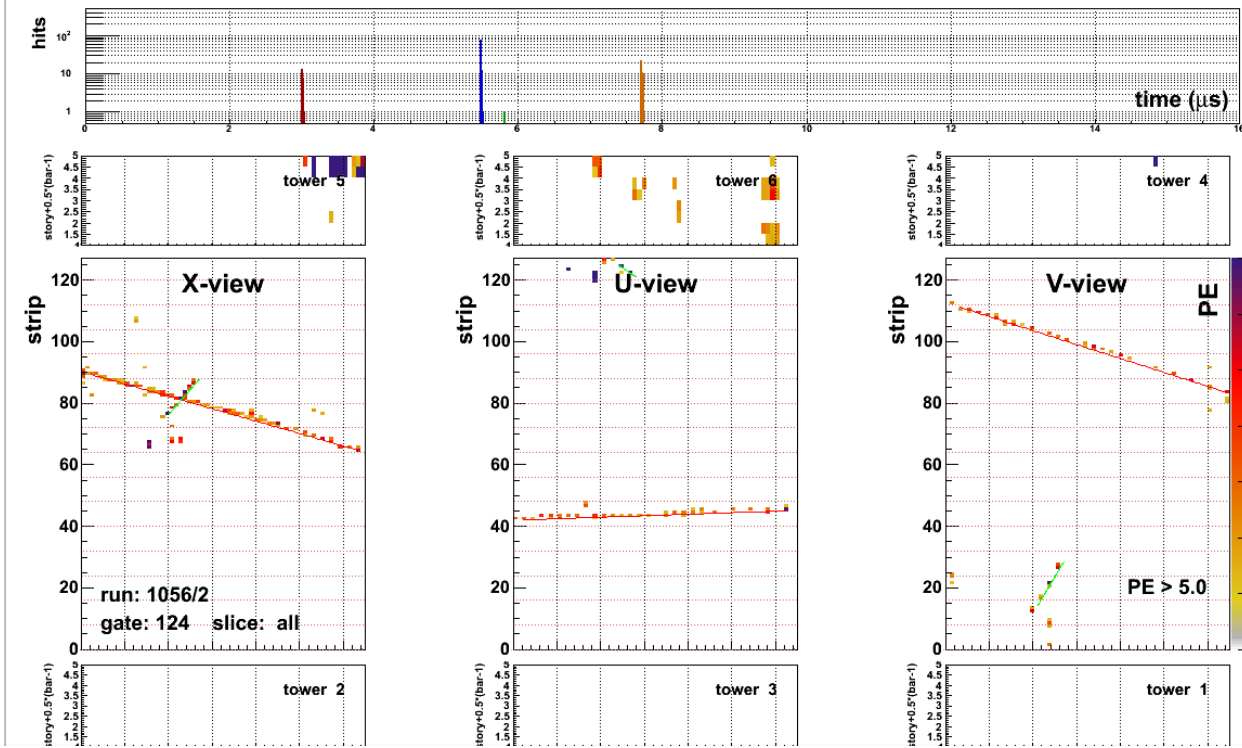
References

- [1] David Griffiths. *Introduction to Elementary Particles*. Wiley-VCH, 2008.
- [2] Mark Alpert. A New Neutrino Hunt. *Scientific American* 299:3 (2008): 32.
- [3] Arthur B. McDonald. Solving the Solar Neutrino Problem. *Scientific American* 288:4 (2003): 40.
- [4] David Griffiths. *Introduction to Quantum Mechanics*. Pearson-Education, 2005.
- [5] Mark Alpert. The Neutrino Frontier. *Scientific American* 295:2 (2006): 20-22.
- [6] The MINERvA Collaboration. *MINERvA Project Conceptual Design Report*. Fermi National Accelerator Document Database, 2006.
- [7] The MINERvA Collaboration. *MINERvA Technical Design Report*. Fermi National Accelerator Document Database, 2006.
- [8] Deborah Harris. *MINERvA Document 4731-v4: MINERvA in 20 Words or Less*. Fermi National Accelerator Document Database, 2010.
- [9] J. Illingworth and J. Kittler. A Survey of the Hough Transform. *Computer Vision, Graphics, and Image Processing* 44:1 (1988): 87-116.
- [10] Rene Brun et al. *ROOT User's Guide v5.26*. <http://root.cern.ch>.
- [11] Phillip Bevington. *Data Reduction and Error Analysis for the Physical Sciences*. McGraw-Hill, 1969.
- [12] William H. Press et al. *Numerical Recipes: The Art of Scientific Computing 3ed*. Cambridge University Press, 2007.

Appendix

A. Tracking Examples





B. Additional Information for Tracks Used in Average Slope Calculation

