First Deep Learning based Event Reconstruction for Low-Energy Excess Searches with MicroBooNE

MICROBOONE-NOTE-1042-PUB

The MicroBooNE Collaboration

This paper describes algorithms developed to isolate and accurately reconstruct two-track ν_μ-like events that are contained within the MicroBooNE detector. This reconstruction has applications to searches for neutrino oscillations and measurements of cross sections using events that are charged-current quasi-elastic-like, among other applications. The algorithms we discuss will be applicable to all detectors running in Fermilab's SBN program, and any future LArTPC experiment with beam energies $\sim 1\,\mathrm{GeV}$.

I. INTRODUCTION

The MicroBooNE experiment, currently taking data in the BNB neutrino beam at Fermilab since October 2015, 450 m downstream of the target, is a liquid argon time projection chamber (LArTPC) [1]. The detector cryostat has a total capacity of 170 tons of liquid argon, with an active region of $(2.6\times2.3\times10.4)$ m^3. The system comprises two major sub-detectors: a time projection chamber (TPC) for tracking, and a light collection system for trigger and reconstruction of the precise interaction time. The detector has been described in detail in Ref. [2].

Events which are contained within the detector, and which are consistent with the signature of one muon and one proton, are of interest to a number of physics studies on MicroBooNE. These are primarily measurements of exclusive cross sections, including charged-current quasi-elastic (CCQE). This group of events is also important for charged current π^0 events (CCπ^0) and the investigation of the MiniBooNE low energy excess [3] in MicroBooNE. While the signal for those events is hypothesized to consist of an electron and at least one proton, the normalization sample consists of a muon and at least one proton. In this case, the reconstruction is identifying candidate events that may be selected as "$1\mu1p$" (one muon and one proton) events after particle ID. To obtain a large sample of contained $1\mu1p$ events, the MicroBooNE collaboration has developed a specialized reconstruction package specialized for contained two-track events. The purpose of this reconstruction is to identify and reconstruct those events with two tracks emanating from a vertex that are both longer than a specified length; any number of shorter tracks may be attached to the same vertex.

The physics analyses that will make use of this code package employ both the TPC and light collection subsystems. However the three-dimensional reconstruction code described here uses only the TPC information, and so we describe only this subsystem. In the TPC, electrons from the ionization tracks produced by charged particles in the interaction with liquid argon, due to a $273\,\mathrm{V/cm}$ electron field, drift towards three sense wire planes that provide the charge read-out The signals from the three wire planes form three views, U (wires at +60 degrees from vertical), V (-60 degrees) and Y (0 degrees). The U and V planes detect signals via induction, while the Y view is the collection plane. The wire spacing in each plane is 0.3 cm. The wire waveforms are read out with a sampling time of $0.5\,\mu\mathrm{s}$, and with a shaping time of the ASICs of $2\,\mu\mathrm{s}$. This results in highly detailed event information that we exploit by treating the time versus wire hit plots from each of the three planes as images with pixels, as described below. This use of high resolution images allows the analysis chain to make use of deep learning algorithms. Indeed, in this note, we present an algorithm-based reconstruction approach leveraging the output of a pre-processing performed using a deep neural network.

Our reconstruction package is discussed within the context of MicroBooNE analyses. However, the approaches are generic to LArTPC detectors that run in a $\sim 1\,\mathrm{GeV}$ neutrino beam. Future examples of such experiments are SBND and ICARUS, which will run in the same BNB neutrino line as MicroBooNE in the near future [4]. The approach is also appropriate to reconstructing atmospheric neutrino events in the DUNE far detector [5], although in this case, complications due to cosmic rays will be substantially reduced compared to the surface-based detectors on the BNB beam-line.

We will present results based on simulated events at energies relevant to the MicroBooNE beam and using MicroBooNE's simulation package. MicroBooNE uses GENIE [6] to simulate neutrino interactions and CORSIKA [7] to simulate cosmic rays in events. The particles from these simulations are fed to a GEANT4 simulation [8–10] of the detector that has been tuned by comparison to a cosmic ray data set.

II. DATA PRE-PROCESSING

An essential and very difficult problem to solve in reconstruction is identification and removal of cosmic rays from the events. With no overburden and given the 2.3 millisecond readout window, MicroBooNE averages 12 cosmic rays per readout period. The high rate of cosmic

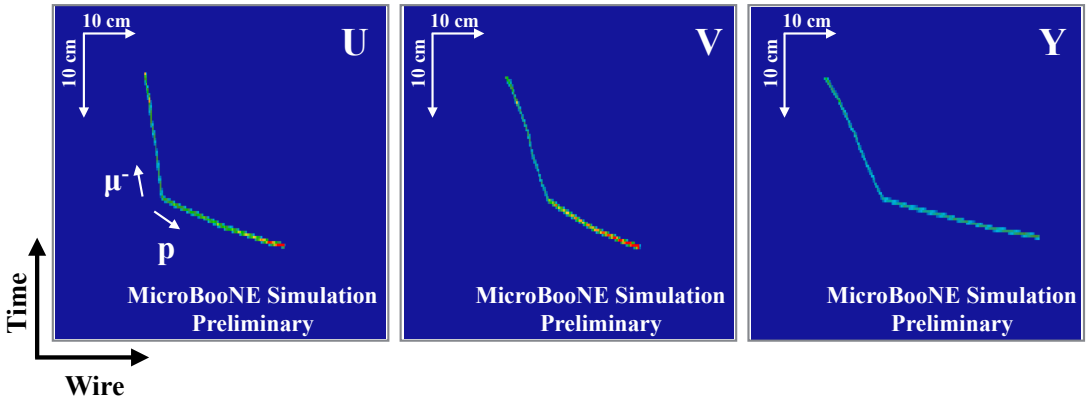


FIG. 1. A simulated ν_μ event shown in the three wire planes that illustrates the features of interest for this reconstruction package. The muon neutrino MC truth energy is 483 MeV. A single proton (deposited energy 266 MeV) and muon (deposited energy 73 MeV) are produced. The image-like nature of the drift-time versus wire-plane plots is apparent, and explains why we use this language. Each two-dimensional bin on these plots is called a "pixel" and, in this case, contains the information on the ADC count. See text for explanation.

rays, in conjunction with un-responsive wires can cause cosmic rays to look like neutrino events. To address this issue, prior to three-dimensional reconstruction, an algorithm is applied to tag pixels corresponding to cosmic rays. This code will be described in a future separate paper, and so is only briefly described here. Cosmic rays are identified by the boundary-crossings at the edges of the active region. Through-going cosmic rays will cross two boundaries, while stopping cosmic rays will cross only one boundary. Exiting muons from charged current neutrino events also cross one boundary, but this is not an issue, since the analyses for which this code is used employ only contained events. The cosmic ray tagging algorithm starts at the boundary and works inward, into the detector, labeling the consecutive charge. Once all charge that is connected to a boundary is identified, the remaining untagged charge clusters on each planes are included in a 3D-consistent volume called "contained regions of interest" (cROIs). These cROIs are then fed into the three-dimensional reconstruction code. Typically about 10 cROIs are found per event. Often tagged-cosmic charge will also appear within a cROI.

The cROIs are then fed into a deep-learning algorithm called Semantic Segmentation Network (SSNet) [12, 13]. The uses of Semantic Segmentation in the analysis, along with tests of its efficiency, will be described in detail in a coming article. Here, we will suffice to say that the Semantic Segmentation identifies all pixels in the cROI image as either background, track-like, or shower-like, where background refers to empty pixels with little to no charge deposition. Typically, the Semantic Segmentation Network will classify muons, charged-pion and protons as track-like, and electrons and photons as shower-like.

In summary, the inputs to the code we describe here are cROIs that have all charge tagged as cosmic-ray, track-like, shower-like or background-like. This information is used to find a three-dimensional vertex using the sets of images from the three planes. At that point a determination is made as to whether there are two, and only two, track-like chains of charge with lengths that pass our requirement. The pixels associated with the track-like chains are clustered. The three-dimensional vertex is then fed to reconstruction of the three-dimensional tracks. For the track-reconstruction stage, the pixel identification tagging is not used. We describe each of these steps below, and present information on the reconstruction efficiency. However, first, we describe the input images that are utilized by this package.

III. USING IMAGES IN THE RECONSTRUCTION PACKAGE

This reconstruction package makes use of MicroBooNE data and Monte Carlo treated as "images." By this, we mean that the TPC data are represented on a 2-dimensional plot, with wire number along the x axis and drift time along the y axis. The choice to analyze the detector in an image-format allows the use of widespread and very powerful computer vision tools such as "OpenCV" (Open source Computer Vision) [15], which is a C/C++ based framework for computer vision that provides useful classes/functions for image processing. It is a widely used application for cutting-edge pattern recognition. Using images also allows for the implementation of deep learning algorithms at two points in the analysis. The first is the SSNet, which precedes this reconstruction package, and was discussed above. The second is a convolution-neural-network-based particle identification, which follows this reconstruction package, and is beyond the scope of this article. Ref. [16] describes how deep-learning-based particle identification can proceed, given the output of the three-dimensional reconstruction package we describe here. We note that likelihood-based particle identification that follows this package is also under development on MicroBooNE.

However, the value of using images goes beyond the applications to deep learning. When constructing the simulated events, it is easy to overlay the pixels associated with a simulated neutrino event onto a real out-of-beam image or an image from the cosmic ray simulation. Also, crucially for the three-dimensional reconstruction, information can be straightforwardly associated with each pixel at each stage of the algorithm, and then carried through to the following stages.

The reconstruction package will primarily make use of two kinds of images. The first is the "ADC-image," which contains information on the charge in each pixel. The second is the "SSNet-image" which contains information on whether pixels representing connected chains of charge are track-like, shower-like or neither. In both cases, the cosmic-ray tagged pixels are masked, and so are not visible in the image. Two additional pieces of information are also provided to the algorithm. The first are images that show the cosmic-tagged pixels. The second are images indicating dead wires.

A. ADC-images

In the case of the ADC-image, the intensity of each "pixel" is determined by summing the amplitude of the noise-filtered, deconvolved signal [17] from one or more wires over one or more time ticks. The amount one sums, either over time or in wires, must be defined when making an image. We sum over six ticks and keep wires individually. This choice comes from the fact that, at 0.5 microseconds per tick, six time ticks is 3 microseconds, only slightly larger than the 2 microsecond shaping time of the ASICs. Also, at the current drift field, the drift velocity of the electrons is 0.11 ± 0.01 centimeters per microsecond. Therefore, the drift distance spanned over six time ticks is about 0.33 cm, which is similar to the detector's 0.3 cm wire pitch. The images are created using the LArCV code available on GITHUB [18]. There is an image for each cROI in each of the wire-plane views.

Figure 1 shows an example event of interest for this reconstruction package. The ADC-Images are made in each plane, as shown. The size of each image is set by the cROI-finding algorithm. The top plot shows a ν_μ CCQE simulated event. This event has true neutrino energy of $E_\nu^{\text{true}} = 483$ MeV, the kinematic energies of the emitted muon and proton are respectively 73 MeV and 266 MeV, which is typical of the kinematics that we aim to reconstruct with this package.

B. SSNet-images

The SSNet-images, which are also constructed for each cROI and for each wire plane view, are created by feeding the ADC-images to a SSNet. The SSNet identifies the pixels based on their surroundings into three categories:

- track pixels
- shower pixels
- background pixels

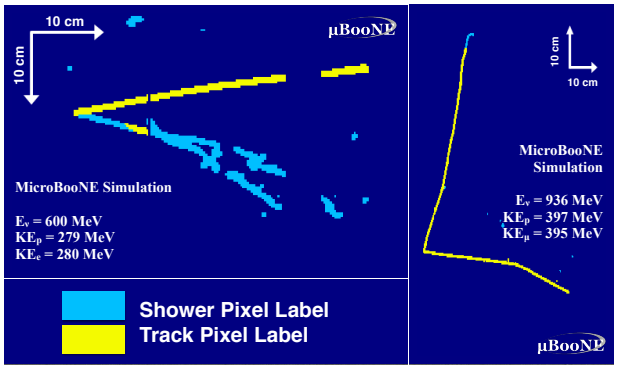


FIG. 2. Two examples illustrating the SSNet-image pixel labeling. The left panel shows a $1e1p$ type event from a $\_e$ interaction. The pixels corresponding to the proton are here here correctly classified as track by the SSNet. The pixels corresponding to the electron are here mostly classified as shower by the SSNet, except for a small portion mistakenly labeled as track. The right panel shows a $1\mu1p$ type event from a ν_μ interaction. Here both the proton and muon tracks are correctly labeled as track pixels. The muon decays into a Michel electron, classified as shower pixels. Dark blue pixels correspond to empty pixels, without charge deposition.

Two examples of SSNet outputs are shown in Fig. 2. The left panel shows a view of $1e1p$ ν_e interaction in the Y plane, of a 600 MeV neutrino producing a 279 MeV proton and a 280 MeV electron. The proton track is correctly classified as containing only track-like pixels (in yellow), and the electron shower is mostly classified as shower pixels (in light blue). A small fraction of the shower pixels are mistakenly labeled as track-like. Background pixels, corresponding to pixels without charge deposition are shown in dark blue. The right panel shows a view in the Y plane of a $1\mu1p$ ν_μ interaction of a 936 MeV ν_μ producing a 397 MeV proton and a 395 MeV muon, that decays into a Michel electron. The Michel electron is here classified as shower pixels, while the proton and muon tracks are correctly labeled as track-like.

In this note, we will focus on the $1\mu1p$ topology, therefore we will be looking for two-track vertices, neglecting the track-shower interface vertices.

C. Cosmic-tagged-images

The reconstruction algorithms described in the rest of this article are also supplied an additional image with cosmic ray information. The through-going muon pixels

in this image are tagged, but visible. These pixels can be optionally removed from the ADC, track, and shower images to help reduce the probability that the reconstruction algorithms will reconstruct a cosmic background.

D. Dead-wire-images

In addition, the vertexing algorithm is supplied with an image marking the spatial location of dead wires. The list of dead wires is run-dependent. Providing this information allows the algorithm to know precisely which region in the image represents pixels which contain no charge, and make a decision about whether to veto this region, and potentially neighboring regions, for vertex-finding and particle clustering.

IV. 3D VERTEX FINDING AND PARTICLE CLUSTERING

This reconstruction step finds the 3D vertex and then clusters pixels belonging to individual particles. In this algorithm, the pixels tagged as cosmic rays are removed from the images. For each of the three views, a set of three images are provided to this algorithm. The first image contains all pixels in the cROI (ADC image), the second contains pixels labeled as track (track image) after the SSNet correction, and the third image contains pixels labeled as shower (shower image). In many, but not all, of the cases of interest to the two-track reconstruction, the shower image will be entirely blank. For finding the vertex, the algorithm focuses the track features in its own image separately.

Along with use of LArOpenCV, described in the previous section, this code makes use of a custom OpenCV package called Geo2D. This package has convenient tools for 2D geometrical analysis to supplement and extend OpenCV built in data types.

This step makes use of only the track-identified pixels (the track image) to reconstruct a 3D vertex. The algorithm searches for a coincident "vee" shape feature as shown in Figure 2, right, across the three wire planes which could indicate the presence of a $1\mu1p$ interaction. The algorithm begins by identifying, per plane, a collection of vertex "seeds". Vertex seeds are pixel locations in the image where a likely vertex may be present, for example at the location where two tracks meet at a kink point. The algorithm identifies vertex seeds by breaking down continuous sets of track clusters into smaller clusters which contain straight segments of charge.

First, a distinction between pixels in the low charge (LC) and high charge (HC) regime as shown in Figure 3 is performed. The division between LC and HC ADC count is a constant threshold per plane and is determined from a study of the pixel intensities of MC protons. The minimum HC value for the U, V, and Y planes are set at 140 ADC, 120 ADC, and 80 ADC respectively. These

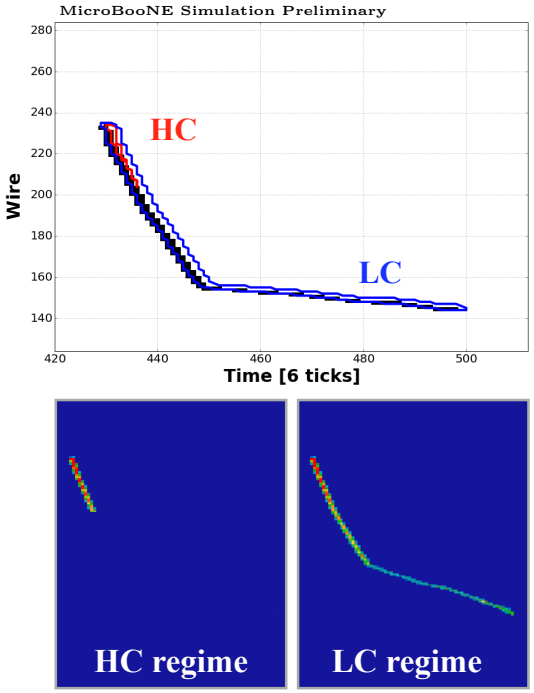


FIG. 3. Low charge (blue) and high charge (red) contours are found in the example $1\mu1p$ event. The HC contour clusters a collection of proton pixels which have a high pixel intensity. The single LC contour encloses all pixels in this track image as they are all above the 20 ADC threshold.

values correspond to 10% of the average pixel value for a proton track on each plane. Pixels with values below 20 ADC count are not considered. Once the pixel ranges are separated, the algorithm finds groups of LC and HC pixels by applying the OpenCV contour finder. This step defines the LC and HC clusters. Pixels that satisfy to the HC condition are also included in the LC clusters, to avoid hallow clusters. HC clusters are therefore a subset of the LC clusters.

Next the algorithm performs a shape analysis by breaking down LC and HC clusters which are not linear. For example, the blue LC contour shown in Figure 3 has an obvious bend or "kink" in it. For each cluster, the algorithm computes the "convex hull" which the smallest convex polygon which bounds the original cluster. Figure 4, top image, shows an example of convex hull (purple polygon). The algorithm identifies the sides of the convex hull which are far away from their corresponding sides on the contour.

The point on the contour that is farthest away from the corresponding hull side is called the "defect point", and is a location where the cluster is potentially bending and changing direction. If the convex hull side is far enough away (5 pixels) from the defect point, the contour is then broken into two at the defect point. The bottom images shows the three clusters obtained after this stage, 1 HC cluster and 2 LC clusters. The algorithm then iteratively breaks down all clusters into linear segments until no

defects point remain.

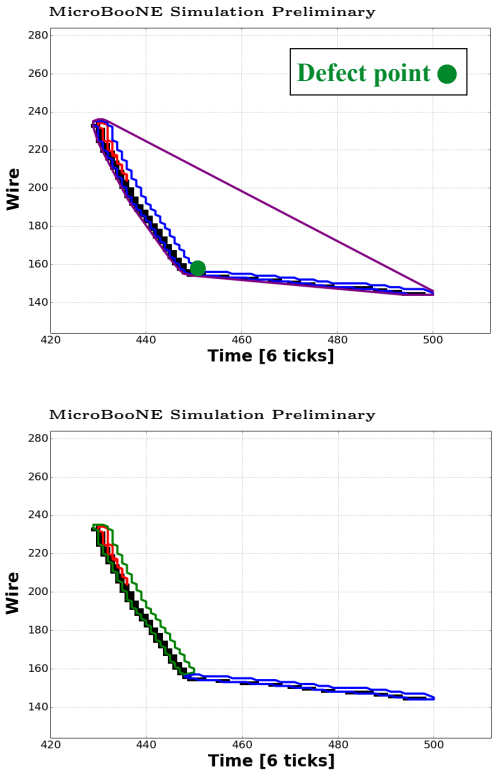


FIG. 4. The convex hull (purple) is computed for the LC contour (blue). A defect is found on one of the convex hull edges. Top : A defect point (green) is found on the LC contour, more than 5 pixels away from the corresponding convex hull edge. The defect point indicates the location where the cluster is bending. Bottom : The LC cluster is then broken into two clusters at the defect point using the defect point. The breaking procedure is carried out for each contour until no defect points remain on the cluster.

The collection of defect points are the first set of vertex seeds.

The second set of vertex seeds is found using a Principal Component Analysis (PCA) procedure which fits the clusters to a straight line hypothesis. The PCA is a linear approximation which minimizes the perpendicular distance between the data (the pixel points), and the estimated line. A PCA is calculated for each broken cluster separately. Since all clusters have been broken into linear segments by removing defects, a linear approximation is suitable. The algorithm then computes the intersection of all possible PCA lines on the plane. If the lines intersect near a location on the image with charge, the point is saved and is added to the set of vertex seeds. Intersection points far from any charge are ignored. The top image in Figure 5 shows the three PCAs found in the event example. Although three intersections points are found, only two correspond to pixels with charge and are then kept (middle image).

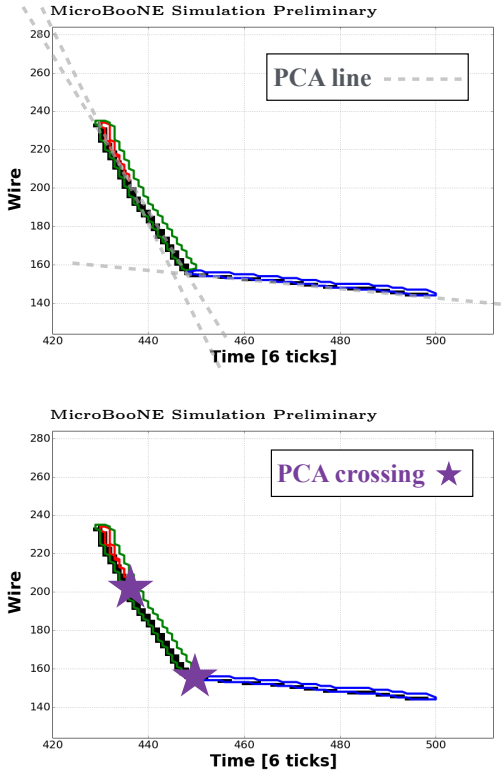


FIG. 5. Top: For each broken cluster, a 2D PCA line is fit to the cluster. Three PCA lines (grey dashed) are found in this example, one per cluster. Bottom: The points where the PCA lines cross are shown as purple stars and are called PCA crossing points. Only two PCA lines cross near charged pixels producing two PCA crossing points. The third PCA crossing point is not located on charged pixels and is ignored. Vertex seeds found in the example $1\mu1p$ event. The purple stars are vertex seeds from PCA crossing point locations.

This type of vertex seed helps find the 2D location where tracks which may be changing direction in the image. Also, using a linear approximation for the clusters gives an additional set of points than the defects points, increasing the efficiency to find the actual vertex.

The final set of vertex seeds is composed of both the breaking down of clusters and the PCA intersections. The proximity of the seeds from cluster breakdown and PCA crossing is a strong indication of the actual vertex location. Each of these 2D points are considered a vertex "seed" and serves as a starting points for 3D vertex search.

The algorithm makes use of the fact that a correct vertex will appear near the same time tick in each view to reduce the seed sample to the time-coincident ones. The X position of these candidates can be determined by using the trigger time and the known drift speed to match the time tick to a X position.The Y and Z positions can then be determined by using wire coincidence between

two or three planes.

The algorithm then performs an exhaustive search for a 3D vertex around each seed by minimizing a quantify called the "angular metric". This angular metric is a single quantity which measures the likeliness of two tracks being emitted radially outward at the same position across two or more planes. This metric will be minimized when a point in 3D space is found where the 2D projections indicates that particles coming out of a single point.

The algorithm begins a search for a 3D vertex by combining images across two or three wire planes. The following operation is applied to every vertex seed on each plane. Circles of radius 6 and 12 pixels are drawn with the given vertex seed at the center. The circle size with the greatest number of clusters coming out is used. In cases where the same number of clusters cross the circles, the largest circle is kept. The algorithm then identifies the points at which the out-going clusters cross the circle. For each pair of out-going clusters, the algorithm evaluates two angular quantities:

1. The first angular quantity, θ, is the smallest angle between the center of the circle, and each pair of cluster-circle intersection points.

2. The second quantity, ϕ, is calculated in the following way. At each cluster-circle intersection point a small region of 7×7 pixels is identified. In this region, a PCA of charge carrying pixels is computed, and provides an estimate of the local cluster directions. ϕ is the smallest angle between each local PCA pair.

The definitions of the two angles are summarized at the top of Figure 6.

Starting at the vertex seed, two lines segments are drawn from the initial local PCA approximations. The circle center is then stepped in increments of 1 pixel along the straight line segments. When the local track direction on the circle boundary (ϕ) matches the direction between the center and crossing points (θ) the difference is small and indicates that the circle is at a location where the tracks are coming out straight from the center point, likely indicating a kink feature. The algorithm then computes the *magnitude difference* $d\Theta = |\theta - \phi|$. At each step $d\Theta$ is evaluated and stored per time tick. The stepping stops when the algorithm has scanned a 40x40 pixel region around the initial vertex seed. This procedure is repeated for each vertex seed in the plane. The evolution of $d\Theta$ as a function of the corresponding time tick is represented at the bottom of Figure 6. If two vertex seeds happen to scan the same time tick, the lowest $d\Theta$ value is stored.

Next, the $d\Theta = |\theta - \phi|$ versus time-tick maps for the three views are summed to produce a single distribution of $d\Theta$ values. The summed magnitude difference is the angular metric to be minimized to find the best

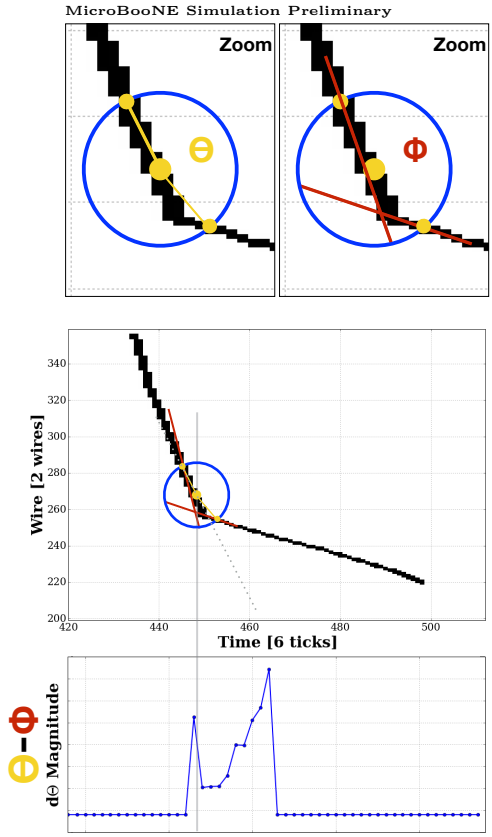


FIG. 6. For each candidate vertex position, a circle is drawn centered at the vertex position, the tracks intersects the circle boundary at two points. The yellow lines link the track-circle crossing points to the circle center. The angle between the two yellow lines is θ. The red lines represent the local PCA approximation at the track-circle intersection. The angle between the red lines is ϕ. The gray dashed lines represent the initial PCA lines. The circle is scanned along the initial straight line PCAs and the magnitude difference of θ and ϕ is recorded per time tick. The bottom plot shows the resulting spectrum for the scanning procedure applied centered at the vertex seed near the kink point. The graph is $d\Theta$ versus time. A dip in the magnitude difference is observed at the kink point when the two angles agree. The spectrum is flat in the left and right regions where no time tick is scanned.

vertex seed. The distribution is smoothed using a rolling mean approximation of 6 time ticks. Finally the algorithm searches for local minima in the spectrum to find regions where a coincident vertex feature appears across multiple planes as shown in Figure 7. In each plane, the circle at the local minima time is examined and matched across planes using wire coincidence. If coincident wires are found, then a 3D vertex is claimed. An equivalent procedure is carried out by performing the search for local minima in the angular spectrum in overlapping wire regions.

To obtain the 3D vertex position, the vertex time provides the X-coordinate. The Y and Z spatial information of the vertex is extracted from wire coincidence across

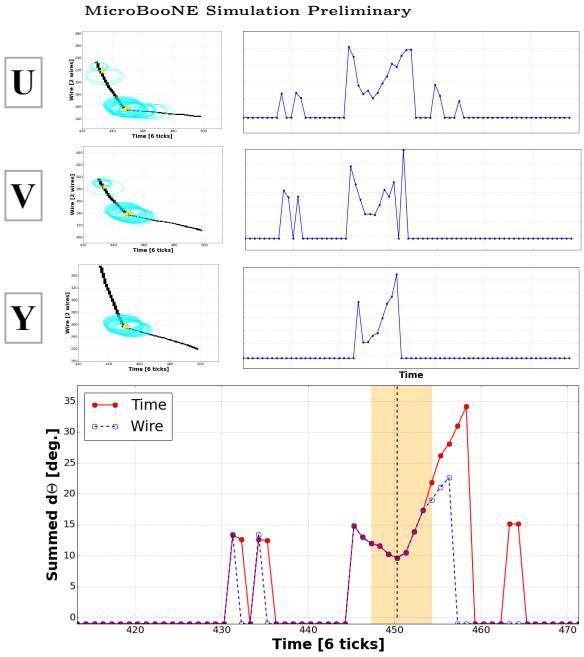


FIG. 7. The scanning procedure is carried out for each vertex seed across three planes. The top left three images show the history of circles scanned at each vertex seed. The top right three spectra show the angular metric as a function of time tick. The bottom plot shows the sum of the three spectra summed together after a rolling mean approximation is applied. A gradient algorithm finds a single local minimum present in the yellow band. This time tick is searched for on each plane to estimate a 3D vertex.

any pair of U, V, and Y planes. Finally, the resulting 3D vertex is refined using a 3D volume scan in a $(4\times4\times4)\text{cm}^3$ region around the vertex. For each location in that space, the 2D projections of the point in 3D space are estimated, and the $d\Theta$ variable is estimated. This stage allows to further improve the estimation of the best 3D location of the vertex.

V. CORRELATING PARTICLES ACROSS THE THREE VIEWS

Once the 3D vertex is identified, we must correlate the images of each particle that emanates from the vertex across the three views. This is done in two steps. First, in each view, unique particles are identified using 2D clustering. Then, these 2D clusters are matched across views. The event is then analyzed for a gap at the vertex. In the case of the two-track analysis, for example, two close, untagged cosmic rays may be misreconstructed through the other steps in the analysis chain, but will have an identifiable gap that allows tagging.

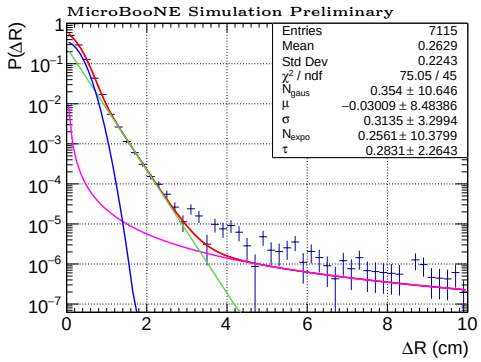


FIG. 8. Vertex resolution for well reconstructed $1\mu1p$ events. The 3D distance between the Monte Carlo neutrino vertex and the reconstruction vertex is shown. Each bin ΔR_i is weighted to the volume of the spherical shell of radius ΔR_i and thickness 0.2 cm, to obtain the probability density function of the vertex resolution. The distribution is described by three populations: a Gaussian centered on the actual vertex position, and with a resolution of $\sigma = 0.3$ cm, the size of a pixel, an exponential population, of decay length $\tau = 0.3$ cm; vertices in that population tend to be reconstructed on pixels nearby the pixel the true vertex projects to and a low statistics tail described by the $1/R^2$ power law, due to the scaling applied to each bin; these vertices are mistakenly placed on nearby cosmics, and with no correlation to the actual neutrino.

VI. 3D VERTEX STUDIES

In this section, we discuss the quality of the vertex reconstruction code. We consider the resolution of the 3-D vertex, optimization of vertex-finding scanning-radius is changed, and the efficiency of vertex finding for the MicroBooNE detector.

A. Vertex Resolution

The quality of the track image vertex-finding can be assessed using MC by considering the difference between the 3D distance of the simulated true neutrino vertex to the reconstructed vertex. This is shown in Figure 8. The Probability Density Function is obtained by scaling each bin by the volume of space around the vertex with a given ΔR : $\pi\Delta R^2 \cdot \text{dR}$ where dR is the thickness of the bins. Three populations appear in the distribution:

- a Gaussian, centered on the true vertex position, and of resolution $\sigma = 0.3$ cm, corresponding to the size of a pixel, corresponding to events that are being reconstructed on the same pixel the true vertex projects to,

- an exponential of typical length $\tau = 0.3$ cm, corresponding to vertices reconstructed on pixels nearby the pixel the true vertex projects to,

- a low statistics tail described by the $1/R^2$ power law, due to the scaling applied to each bin; these vertices are mistakenly placed on nearby cosmics, and with no correlation to the actual neutrino.

It is to be noted that the first three orders of magnitude of the distribution lie within the first 2 cm from the vertex, with an overall resolution of ∼0.3 cm, the size of a pixel.

B. Vertex Efficiency

The vertexing algorithm is sensitive to four upstream factors which impact the ability to find a consistent 3D vertex across planes. We analyze each factor separately to determine their impact on vertex reconstruction efficiency. The four factors are:

- Dead wires
- Cosmic pixel tagging
- cROI placement on image
- SSNet pixel classification

For this study, we use a "1μ1p Golden Sample"– simulated events within the fiducial volume that is 10 cm from any edge of the active volume, that have only one muon and one proton with kinetic energy greater than 35 MeV and 60 MeV respectively at the generator level, and that have a muon contained in the active volume.

We decouple the efficiency into two steps: first, the efficiency to find the correct cROI. A found cROI is correct if it contains the true neutrino vertex position. Then, once the cROI is found, the efficiency to find the vertex. Many vertices can be found per event, due to the possibility to place vertices on remaining un-tagged cosmic rays, or possible kinks in neutrino-related tracks. We consider the vertex correctly reconstructed if a vertex is found within 5 cm of the true vertex position.

As described above, the vertex reconstruction algorithm is particularly sensitive to four upstream factors. First, if the neutrino vertex lies in a dead wire region across two or more planes then no neutrino induced pixels will appear in the image and no vertex can be found. Second, the neutrino can only be searched for inside a reconstructed well reconstructed cROI, such that the cROI contains the neutrino vertex in at least two planes. Third, the SSNet pixel classification network has inefficiency at labeling the electron and gamma type particles as shower and the muon and proton type particles as track. This can cause an inefficiency in the vertex finding algorithm where a shower-on-track-end or kink in the track image may not be present and consistent across two or more planes. Finally, the cosmic ray pixel tagging algorithm for marking through-going muons in the image may tag neutrino induced pixels, removing them from the image.

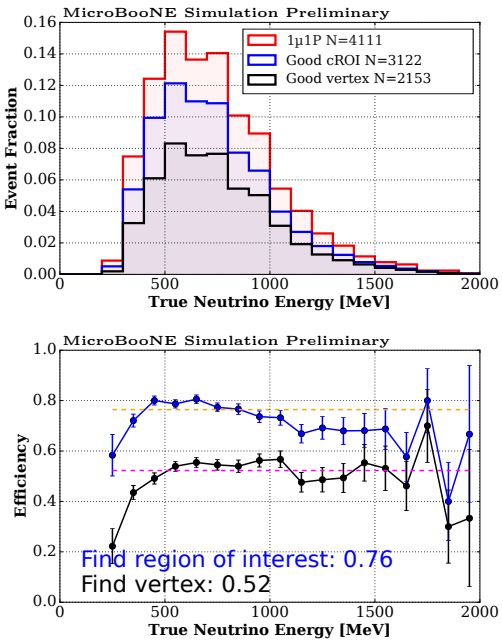


FIG. 9. Good cROI efficiency and overall vertexing efficiency as a function of true neutrino energy for 1μ1p events.The red histogram corresponds to all generated events, the blue histogram are events which have a Good cROI, and the black histogram are events which are well reconstructed. The ratio of histograms is shown in the bottom plots. The orange and magenta lines are one parameter fits to the cROI-finding and vertex-finding efficiencies respectively.

Figure 9 shows the efficiency for finding a cROI, and then the efficiency finding the vertex. The average efficiency for finding the good cROI is 76%, and the averaged efficeincy for finding the vertex, from a found cROI is 69%. The overall efficiency, of finding a cROI, then a vertex, from the original generated MC events is 52%.

Figure 10 shows the spatial dependency of the vertex finding efficiency when applying all upstream stages of the reconstruction. Two major un-responsive regions can be identified that have a noticeable impact on the vertexing algorithm. There is a band of dead wires shown on the Z plane around 700 cm on the Z axis where no reconstructed vertex is found. In addition, there is a band of low efficiency region starting at Y value of approximately −100 cm sloped upward to Z value of approximately 200 cm. Both these regions are consistent with two large dead wire regions within the MicroBooNE detector.

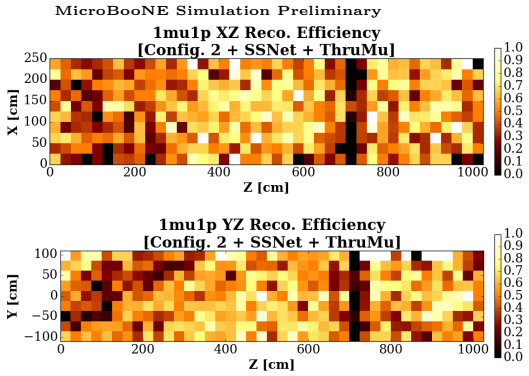


FIG. 10. Efficiency for finding a well reconstructed $1\mu 1p$ vertex in the XZ (top) and YZ (bottom) planes. Each cell is the ratio of number of events with well reconstructed vertex to the number of neutrino events in that cell.

VII. 3D TRACK RECONSTRUCTION ALGORITHM

A. 3D track finding

The reconstruction of 3D tracks is required to obtain the track kinetic energy. We convert the 3D length of the track to kinetic energy using the known stopping power of each type of particle in liquid argon. The full track length cannot be accurately inferred from the 2D images, thus 3D reconstruction is necessary. Rudimentary particle identification is also necessary in order to apply the appropriate stopping-power conversion.

Track reconstruction is particularly sensitive to the quality of the image on a large scale, and to data-Monte Carlo differences. An interruption of the charge deposition along the track, due to dead or noisy wires, waveform deconvolution artifacts, etc., may lead to a wrong reconstructed length and ultimately to the reconstruction of an unphysical energy.

Because of the sensitivity of the OpenCV-based pixel clustering to dead wires and SSNet labeling continuity, only the position of the vertex is used for track reconstruction, not the particle-by-particle clustering. Moreover, the location of the vertex within the cROI may cause the tracks and shower to exit the cROI. If this happens, although the vertex can be identified, the particle clustering may not reach the end of the tracks and showers, causing a misreconstruction of the tracks and showers. Searching for the 3D path followed by the track all the way to the end of the track is thus performed by a separate tracking algorithm.

The track finding algorithm takes as input the full ADC images for each of the three planes and the vertex 3D point. No other information is used. The ADC image is then "thresholded" by bringing to zero all pixels of value less than 15 ADC.

The reconstruction of a track finds a set of 3D points that belong to a given track by performing a stochastic search in the neighborhood of previously found 3D points, starting with the vertex position found in the previous section. A regularization is then performed to find a minimal set of ordered 3D points that describe the track at the required spatial resolution. Finally, observables such as length, local and average charge deposition, and angles can be estimated.

Given a 3D point on a track (the vertex, or an already found 3D point), finding the neighboring points, and following the track as far as possible, is achieved by iterating the following steps as long as new 3D points can be found:

- The seed of the reconstruction step is placed at the last found point
- A set of random 3D points is picked inside a sphere of radius $2 + 4 * e^{(-N/3)}$ cm, where N represents the number of points already found in that track. The purpose here is to allow a wider search around the vertex point, but once a track is found, the restricted radius helps to prevent the track reconstruction from jumping to a nearby track.
- Only the points that project back on pixels with non-zero charge deposition on all planes are kept, with at most one plane on which the point projects on a dead wire.
- New points can only be added if the sum of the ADC values of the deposited charge on the pixels on which they project is greater than that of the already placed points of that iteration.

At this point, we have a set of neighbors to the seed. Some of these points are not relevant, because they are too close to an already found point or track. They are rejected based on the following criteria:

- New points cannot be placed closer than 0.5 cm from an already placed point
- New points cannot be placed closer than 3 cm from an existing track.

All the remaining points at this stage are added to a proto-track, a cloud of un-sorted 3D points that correspond to non-zero pixels. The point within the new found points that is the furthest away from the current seed is now used as the new seed and that phase is iterated as long as new points can be found. This phase ensures that explored region is pushed as far as possible along the track.

However, the points in the proto-track are not ordered and do not follow a linear path. They zig-zag back and forth, and within the thickness of the track. The next step is to order the points by linking each one to its most

likely neighbor. This next neighbor is found my minimizing a global score :

$$\begin{aligned}\text{global score} = \; & 5 \cdot L_1 + 0.1 \cdot L_2 \\ & + 2 \cdot (2 - \cos\theta) \\ & - 10 \cdot (2 - \cos\phi)\end{aligned}$$

The distances L_1 and L_2, as well as the angles θ and ϕ are summarized in the cartoons on Figure 11. The dots represent the set of 3D points in the proto-track. The black dots are the points that have been sorted through, and the light blue dots are the remaining un-sorted points. The green and red dots correspond to the vertex and end of track respectively. The end point is selected as the 3D point the furthest away from the vertex. The red broken line corresponds to the path found within the sorted 3D points. For each candidate within the points that are still un-sorted (here the dark blue point) the two lengths and angles are computed : L_1 is the distance to the last selected 3D point, L_2 is the distance to the end of the proto-track, θ is the angle from the last two sorted points to the candidate, and ϕ is the angle between the candidate, the last sorted point, and the end of the proto-track. Once the points in the proto-track have been ordered, there is a logical path from one point to the next, and some points are rejected as they are never the best candidates. However, at this point, the track still zig-zags and is formed by too many 3D points to be a good representation of the particle path, so a second stage is required.

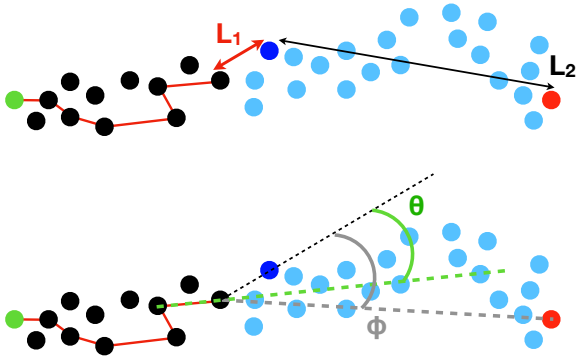


FIG. 11. Once the set of uncorrelated 3D points is found, a sorting algorithm finds a logical path. From a sorted point, the other candidates (here dark blue point) are evaluated based on the distance to already sorted points (black points), the remaining distance to the end of the track (red point) and the two angles, with respect to the last sorted points (θ) and to the end of the track (ϕ). The green point represents the vertex and the light blue points the points of the found set that have not been sorted through yet.

The second stage smooths the path and makes it more direct. We loop through the set of 3D points, rejecting superfluous points based on several criteria:

- a new set of point is created by performing a rolling average of two consecutive point,
- the new set is ordered by moving to the closest neighbor,
- the new point must be closer to the end point than the previous one,
- the distance from previous point cannot be more than 5 cm, this indicates a possible jump to another near-by track,
- the points that deviate by less than 0.5 cm from the line between points n-1 and n+1 are removed.

B. Finding the other tracks

These operations are then iterated as long as a new track is found. To prevent the algorithm from finding the same track multiple times, the pixels corresponding to a found track are masked in the ADC image. Two regimes are used to mask the pixels.

- If the 3D points are within 2 cm of the vertex : pixels within a 3-pixel sleeve around the projected track on each plane are erased.
- If the 3D points are beyond 2 cm of the vertex : pixels within a 6-pixel sleeve around the projected track on each plane are erased.

Pixels are erased on a smaller sleeve close to the vertex in order to allow the algorithm to be efficient at finding tracks that overlap, i.e. that would have a small projected angle, in one of the three planes.

Once no new track is found, we iterate the process to the end points of the tracks already found. Indeed, the end points were selected as the point the furthest away from the vertex, but in some cases, if multiple scattering cases the track to curl up, the actual end of the track is not the furthest point. Starting at the end of a found track and looking for a missing portion of track helps reducing these cases. The two portions of the same tracks are then put together in a single new track.

C. Self-diagnostic

Once all the tracks associated with a vertex have been found, it is important to recognize and possibly reject cases where the reconstruction failed.
This cross-check relies on a set of random points thrown on a spherical shell of radius 3 cm at the end point of each track. Only the forward going points with a solid angle of

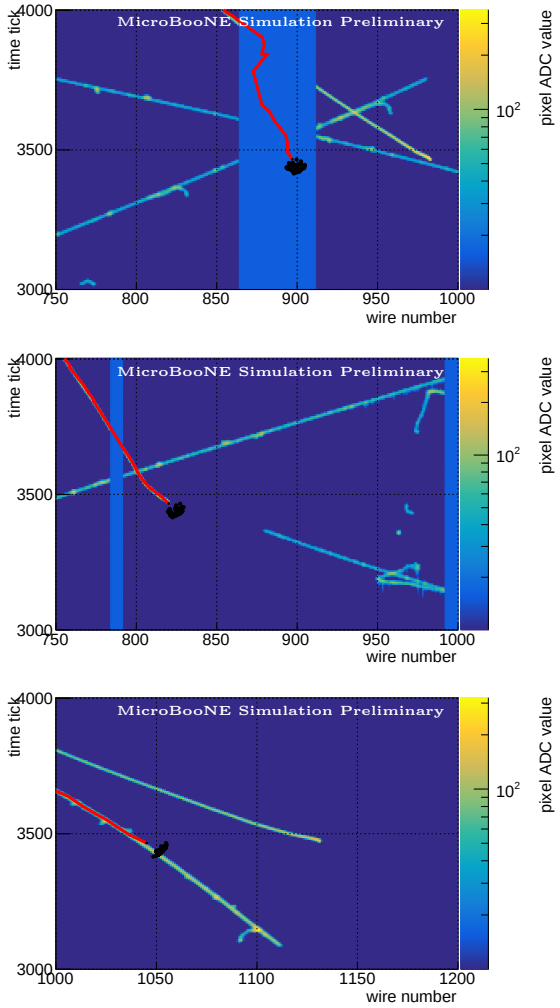


FIG. 12. Once the end of a track is reached, random points are thrown on a spherical shell to estimate if the end of track has actually been found or if the reconstruction ended early because of dead wires or a faint track on an induction plane. The three cases are represented here: (top) the track ends while on a dead region, (middle) the track ends at the end of the charge deposition on that plane, and (bottom) the track ends while the charge deposition continues on that plane.

65° are kept. For each track, the fraction of points that project on pixels corresponding to dead wires, empty pixels and pixels with charge deposited is evaluated, and a label is attributed to the end point on each plane. Figure 12 shows the three possible case we distinguish. The black dots and line correspond to the projection on a given wire plane of the reconstructed 3D points, the colored pixels are the charge deposition, and the uniformed blue region correspond to dead wires.

- **(top)** the reconstructed track ends in a region with dead wires,
- **(middle)** the reconstructed track ends at the end of the charge deposition on that plane,
- **(bottom)** the reconstructed track ends while the charge deposition continues on this plane.

In the case where the charge deposition seems to continue, it is clear that the reconstruction failed to find the actual end of the track, but in the cases where the track ends at the end of the charge deposition or on an un-responsive region, one needs to have a more global approach and look at the other planes.
Indeed, a track that reaches the end in an un-responsive region in two planes but for which the third plane shows that, at least in that plane, the end of the charge deposition is reached, can be considered as a good reconstruction. However, in the case where the charge deposition seems to continue in the third plane, this indicates that the reconstruction did not reach the end of the track.

Another case that can be addressed in this way is when the track local direction is almost co-planar to a wire in the induction planes, i.e. the track becomes locally vertical in one of the images corresponding to the U or V wire planes. The signal in these planes being bipolar, the tracks can appear faint, or interrupted. In that case, the reconstruction will sometimes stop at the interruption point. This end point is going to be classified as correct since it appears that, indeed, the track has stopped. In the other two planes however, the end points will be classified as bad because the charge deposition seems to continue.

Based on the label of the end points in the various planes, a label is attributed to the vertex as shown in Table I.

Dead	Empty	Track	label
3	0	0	Dead Wire
2	0	1	Dead wire
2	1	0	OK
1	2	0	OK
1	1	1	Faint track
0	2	1	Faint track
0	1	2	Faint track

TABLE I. Vertex label attribution depending on the labels of the end points in the various planes.

In the rest of the paper, a well reconstructed vertex is a vertex that satisfies all these conditions: a vertex with exactly two tracks of more than 5 cm that both ends at the end of the charge deposition.

From the reconstructed 3D-path of each particle exiting a vertex, several key observables can be estimated. As we describe these observables in the subsections below, we will characterize the results using a $1\mu 1p$ MC sample. In this sample, all events are generated with exactly one proton above 60 MeV and exactly one lepton above 35 MeV. A fiducial volume selection of 10 cm is applied on the vertex position, and a containment criterion applied to the lepton only. In addition to the neutrino interaction, a cosmic background from CORSIKA is overlaid onto the images.

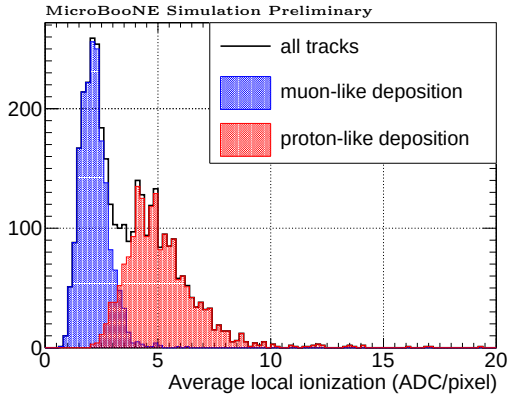


FIG. 13. Average charge deposition along each reconstructed track. The red and blue distributions represent respectively the tracks with the highest and lowest average ionization in a given vertex.

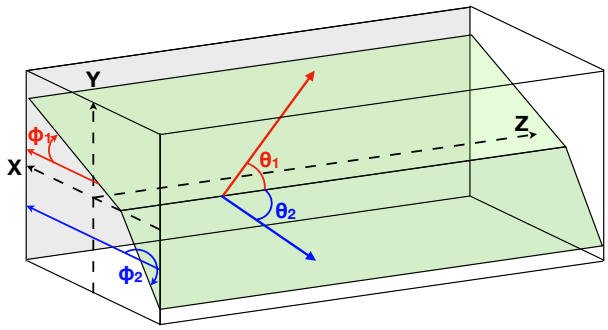


FIG. 14. Description of ϕ and θ angles for each particle in MicroBooNE. ϕ is the angle of a track projected in the (X,Y) plane with respect to the X axis, and θ is the angle of a track with respect to the beam axis (Z axis).

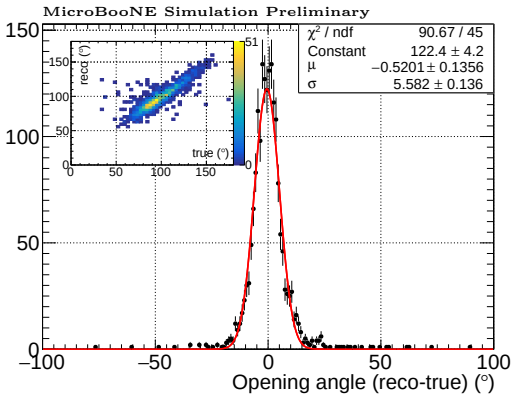
VIII. OBSERVABLE ESTIMATIONS AND PERFORMANCE EVALUATION

A. Local ionization

For each 3D point, the local ionization for a given plane is computed by integrating the values of non-zero pixels in a 2 pixel radius around the projection of the 3D point on that plane. The values measured on the three planes can then be used individually, or summed across planes. A scale factor $3/N$ is applied where N is the number of planes on which a non-zero value was found. This scale factor allows to correct for a possible plane in which the 3D point projects onto an un-responsive region.

Once the local charge deposition around each 3D point has been acquired for each of the three planes, one can compute the Average ionization as the local ionization averaged over the reconstructed 3D points of a given track.

The average ionization reconstructed for $1\mu1p$ simulated ν_μ events in MicroBooNE is shown in Figure 13. At this stage, no particle identification has been performed, the blue and red populations have been separated by identifying the muon as the track with the lowest average ionization within the pair of reconstructed tracks (blue distribution) and identifying the proton as the track with the highest average ionization (red distribution). It is important to note that these particle identifications are relative within a pair of reconstructed particles, assuming one is a proton and the other a muon. All vertices with two reconstructed tracks will have tracks identified as muon or proton with that method. A more definitive particle identification will be performed later on in the analysis chain.



(a)

FIG. 15. The difference between the reconstructed opening angle and the true opening angle shows a $(6.0 \pm 0.1)^\circ$ resolution. The insert shows the linearity of the reconstruction.

B. Angle Estimation

For each individual track, the 3D points within 15 cm of the vertex are averaged. The vector from the vertex to that mean point describes the path of the particle at short range. The angles ϕ (projected angle in the (X,Y) plane) and θ (angle with respect to the beam axis) are evaluated for each track as described in Figure 14.

Once the angle of each track has been computed, an opening angle can be evaluated. Figure 15(a) shows a comparison of the reconstructed opening angle and the true opening angle. An overall resolution of $(5.5 \pm 0.1)^\circ$ is found. The insert shows the linearity of the opening angle reconstruction.

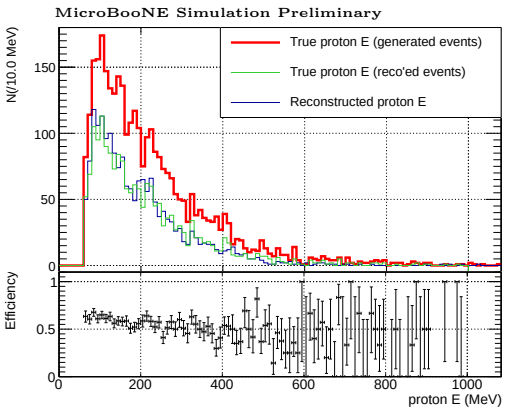
C. Energy Estimation

1. *Individual tracks*

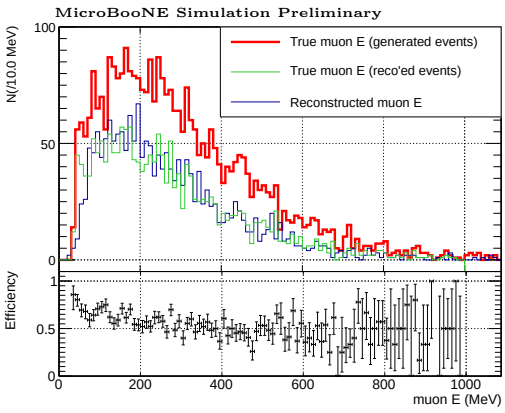
The length of each track is the sum of distances between two consecutive points. From the length of a track, a kinetic energy can be obtained, assuming a given particle identification, based on the stopping power of muons and protons in liquid argon [19] [20]. As no particle identification has been performed yet, energies for both hypotheses are estimated for all tracks. It is left to the analyzers, downstream, to decide which one to use based on more rigorous particle identification.

For the sake of this discussion, and evaluating the tracker's performances, the attribution of proton or muon identification is performed as previously, by using the average ionization and calling muon the particle with the lowest average ionization and proton the one with the highest.

The dependency of the reconstruction efficiency with respect to the energy of the individual particles in the vertex, after absolute truth-based calibration, is shown in Figure 16. 16(a) shows the proton energy distributions and 16(b) shows the muon energy distributions. The red and green histograms show the true distributions for the generated and well reconstructed populations, and the blue histograms show the reconstructed distributions. The bottom plot corresponds to the bin-to-bin ratio of the green to red histograms. A clear down-going trend is visible for the muon distribution. The lower efficiency at high energies comes from the fact that longer tracks have a higher probability of encountering dead region, or be interrupted. This probability increases linearly with the length, hence the steady, linear behavior of the efficiency. The proton distribution shows a lower slope, as the proton length remains shorter than the muon one, and thus plays a less dominant role in driving the efficiency behavior.



(a)



(b)

FIG. 16. Single particle energy distributions for protons (a) and muons (b). The red histograms show the true generated distribution for $1\mu1p$ events, the green histograms show the true energies of the well reconstructed events, and the blue histograms show the reconstructed energies of the well reconstructed events. The bottom plots show the dependency of the global track reconstruction efficiency, defined as the bin-to bin ratio of the green to red histograms.

2. *Neutrino energy estimation*

A length-based estimation of the neutrino energy (E_ν^{vis}) can be achieved assuming a simple $1\mu1p$ CCQE interaction by summing the kinetic energies of the reconstructed muon and proton, accounting for the energy necessary to create a proton and a muon out of a neutron, and using an effective nuclear binding energy B:

$$E_\nu^{\text{vis}} \sim \text{KE}_p + \text{KE}_\mu + m_\mu + B \qquad (1)$$

Figure 17 shows the length-based energy using the true muon and proton energy compared to the true neutrino energy. An offset of $\sin 146\,\text{MeV}$ is required to align the linear population of the distribution on the identity line, corresponding to the muon mass and $40 \pm 10\text{MeV}$ of effective nuclear binding energy [21].

In the rest of this discussion, we will use true E_ν^{vis} as reference to the true visible energy as it is the variable we can approach best in a truly perfect reconstruction.

Figure 18 shows, with the same color code as Figure 16, the visible energy spectrum of all the generated $1\mu1p$ neutrino events and the evolution of the efficiency with respect to the neutrino visible energy. The average efficiency of the vertex reconstruction is $(56 \pm 1)\%$, however, a clear, linear, decreasing trend is visible, consistent with the behavior observed in the muon and proton single-particle efficiencies from Figure 16.

Figure 19(a) shows a comparison of the true and re-

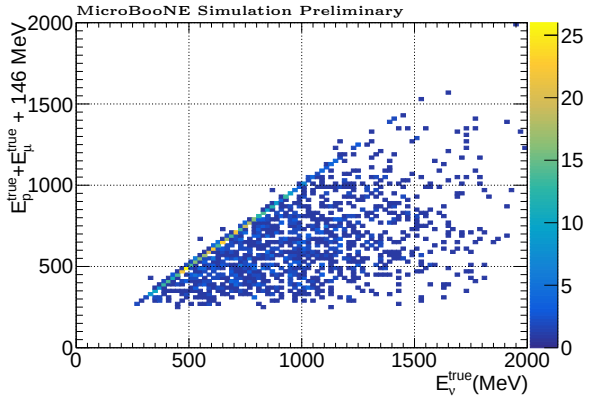


FIG. 17. Comparison of the true caloritmetric neutrino energy to the true neutrino energy from the Monte Carlo simulation. An offset of ~ 146 MeV is required to align the linear population of the distribution on the identity line, corresponding to the muon mass and ~ 40 MeV of effective nuclear binding energy.

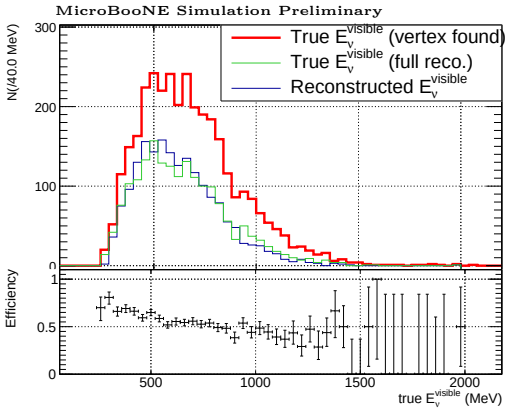
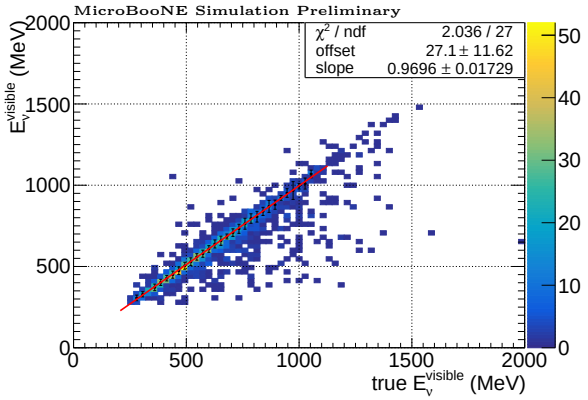


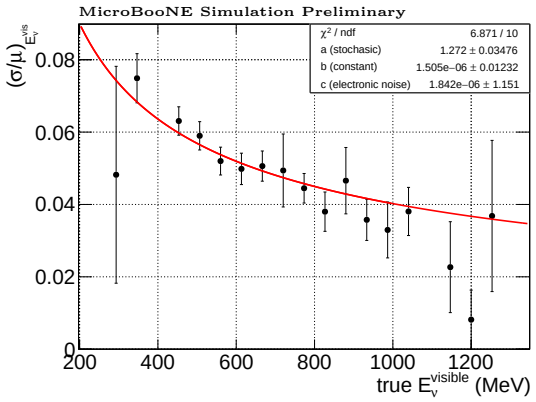
FIG. 18. Relative efficiency as a function of the true energy for events that are being well reconstructed, i.e. for which the reconstruction reaches the end of the tracks and with only two found tracks above 5 cm.

constructed visible energies. Each slice in true energy is fitted by a Gaussian around its mean value. The fit results are shown as the black dots, and the errors on these dots correspond to the σ of the fitted Gaussian. A linear fit performed on the result shows a linearity with a slope factor of 0.97 ± 0.01 and an offset of (23 ± 12)MeV, for a neutrino energy range of $[200 - 1000]$MeV.
Figure 19(b) shows the evolution of the fractional resolution (σ/μ from the previous Gaussian fits) as a function of energy. The errors are the errors on the parameters estimated by the Gaussian fit. The relative resolution is fitted by the function:

$$\frac{\sigma}{E} = \sqrt{\frac{a^2}{E} + b^2 + \frac{c^2}{E^2}} \qquad (2)$$



(a)



(b)

FIG. 19. (a) : Comparison of the reconstructed energy to the true length-based energy. (b) Evolution of the resolution as a function of the true length-based energy.

where a represents the stochastic term of the resolution, b is a constant fractional error, and c characterizes the impact of the noise.

Figure 20 shows the relative error made in reconstructing the full energy of the neutrino. The distribution is fitted with two Gaussians, one describes the well reconstructed events, while the second one describes the events for which the reconstructed energy does not reflect the true neutrino energy. The peak of the distribution shows a bias in reconstructed energy of $\sim$ 1%, with a resolution of $4.0 \pm 0.2\%$. The events in the second Gaussian are also labeled as complete track reconstruction by the self-diagnostic tools, the apparent energy loss comes from non-ionizing energy loss processes such as neutron scattering, proton-muon mis-identification or possible imperfections in identifying failed tracks reconstructions.

The event shown in Figure 21 is a 974.8 MeV neu-

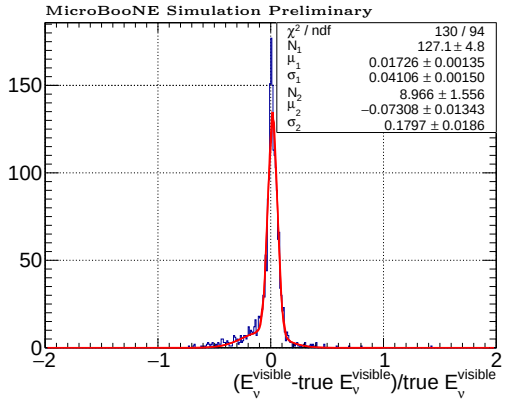


FIG. 20. Relative difference between the energy reconstructed for the interaction (E_ν^{vis}) and the true length-based energy from the simulation (true E_ν^{vis}). A fit by Gaussian functions allows to obtain an estimation of the global fractional resolution of $4.0 \pm 0.2\%$.

trino (true E_ν^{vis}), producing a 602.9 MeV muon and a 225.9 MeV proton. Figure 21(a) shows the three ADC images corresponding to the view of each plane cropped around the neutrino interaction. Figure 21(b) shows the projections of reconstructed tracks for each plane overlaid on top of the corresponding ADC images. The straight vertical light blue lines correspond to the unresponsive wires. The reconstructed energies are respectively 626.8 MeV for the muon track (red dots) and 220.6 MeV for the proton track (black dots). The reconstructed length-based energy is 993.4 MeV and constitutes an error of +2% from the true visible energy.

3. *Spatial dependency of the efficiency*

Figure 22 shows the spatial dependency of the efficiency based on the position of the vertex, projected on the (X,Z) plane (22(a)), and on the (Y,Z) plane (22(b)). On the projection on the (Y,Z) plane, regions with lower efficiency corresponding to large regions with un-responsive channels are visible. The sharpness of these regions, however, is lower than for the vertexing stage itself, as the track reconstruction has moderate abilities to track across un-responsive wires provided that the two other planes have a 3D-consistent non zero charge deposition. However, due to the spatial extension of the tracks, and the global forwardness of the event, a vertex a meter upstream of an un-responsive region can fail reconstruction if one of the tracks cross into that region. Indeed, the un-responsive region around $Z \sim 700$ cm in figure 10 has now moved to $Z \sim 650$ cm.

IX. CONCLUSIONS

We have presented a reconstruction method for three-dimensional event reconstruction of two-track events in LArTPCs. We have discussed the algorithms within the context of reconstruction of events in the MicroBooNE detector. This reconstruction uses computer vision and clustering tools to find 3D-consistent vertices, and a 3D stochastic best neighbor search to reconstruct tracks emerging from these vertices. Because the future experiments of the Fermilab SBN program have similar LArTPC design and run in the same BNB neutrino beam-line, the code is easily adaptable for SBN use. The off-beam DUNE program, which will reconstruct atmospheric neutrinos, will also find aspects of the code to be applicable. The code that can be used to perform this reconstruction can be found publicly on GITHUB [18].

The main parameters that affect the performance of the vertexing algorithm are the out-going proton and muon energies and their opening angle. The vertexing algorithm is also affected by the performance of up-stream reconstruction stages such as the SSNet labelling, the cROI finding, and the cosmic pixel tagger. From the output of the vertexing stage, the track finding algorithm is mainly affected by the un-responsive regions. The dependency of the efficiency on the energy is simply an increased probability of crossing such a region as the track length increases.

The performance of the vertex reconstruction was optimized to precisely place vertices on 3D-consistent kinks with a maximum efficiency. The track reconstruction is then ran for all the found vertices. The performances of the track reconstruction are optimized with the objective to find nearby tracks with small opening angle, and to maximize the length of the reconstructed prong but following a 3D-consistent path of non-zero charge deposition. A layer of smoothing algorithm is then applied to the found set of points and finally, a self-diagnostic is performed to ensure that the reconstructed variables are relevant, and reject the vertices for which the reconstruction failed to follow the track to its end to increase the purity of the final sample of events. The efficiencies of the vertex finding and the track reconstruction are $52 \pm 1\%$ and $56 \pm 1\%$ respectively. The spatial resolution of the vertex finding algorithm is of the order of the wire spacing of MicroBooNE, and the track reconstruction achieves an energy resolution of $4.1 \pm 0.1\%$. The spatial reconstruction tools presented here are exploiting the excellent resolution capabilities of LArTPCs.

Further work will include allowing the tracker to recover tracks by improving its ability to cross through un-responsive regions. The local ionization will also be exploited to better pin-point the end of muon tracks and separate the Michel electron. A spatial correction of the reconstructed points based on a precise measurement of the space-charge effect inducing inhomogeneities of drift field and distorting the reconstructed tracks will also be performed, further improving the performances of this

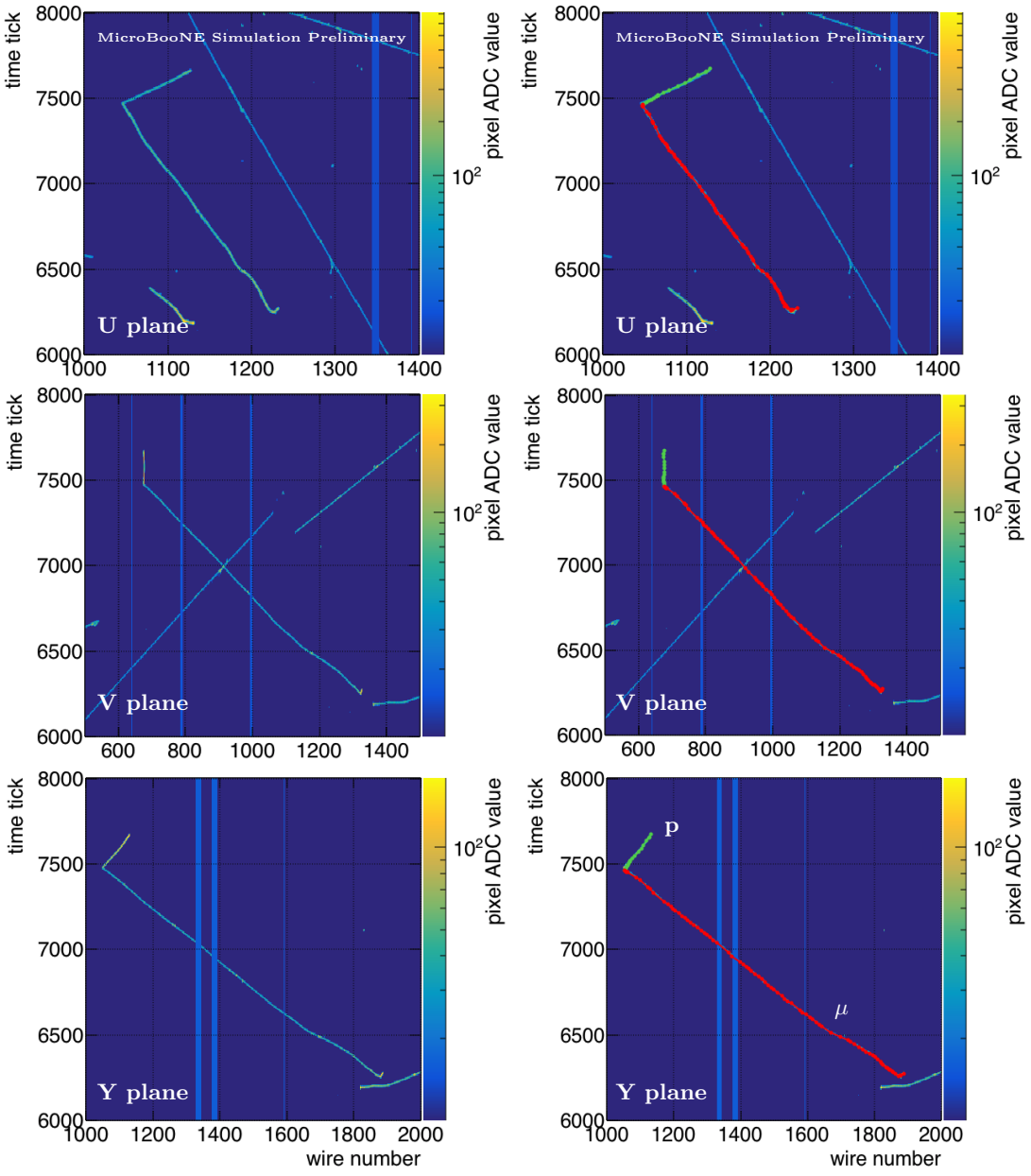
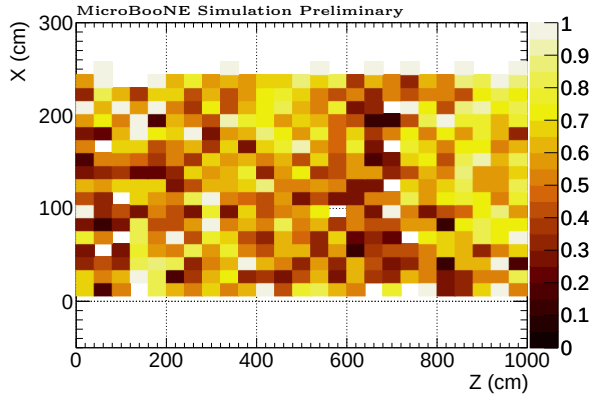


FIG. 21. Example of a reconstructed MC event : (a) ADC image of a 974.8 MeV simulated $1\mu 1p$ neutrino event producing a 602.3 MeV muon and a 225.9 MeV proton. (b) Reconstructed tracks are overlaid on top of the ADC image. The event is reconstructed as a 626.8 MeV muon (red) and a 220.6 MeV proton (black), for a reconstructed E_ν^{vis} of 993.4 MeV.

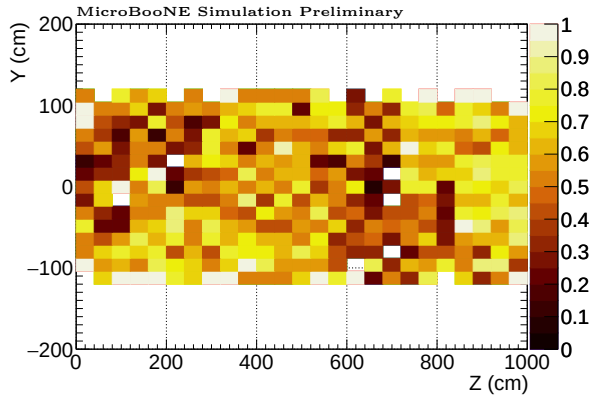
reconstruction.

ACKNOWLEDGEMENTS

This material is based upon work supported by the following: the U.S. Department of Energy, Office of Science, Offices of High Energy Physics and Nuclear Physics; the U.S. National Science Foundation; the Swiss National Science Foundation; the Science and Technology Facilities Council of the United Kingdom; and The Royal Society (United Kingdom). Additional support for the laser calibration system and cosmic ray tagger was provided by the Albert Einstein Center for Fundamental Physics. Fermilab is operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359



(a)



(b)

FIG. 22. Spatial dependency of the tracking efficiency. The color scale represents the fraction of well reconstructed events for a vertex in a given location in the detector. As for the vertexing efficiency, the effects of the two major un-responsive regions are visible in the (Y,Z) plot (bottom).

with the United States Department of Energy.

[1] B. Fleming, "The MicroBooNE Technical Design Report," 2012.

[2] R. Acciarri *et al.*, "Design and construction of the microboone detector," *Journal of Instrumentation*, vol. 12, no. 02, p. P02017, 2017.

[3] A. A. Aguilar-Arevalo *et al.*, "Improved search for $\overline{\nu}_\mu \to \overline{\nu}_e$ oscillations in the miniboone experiment," *Phys. Rev. Lett.*, vol. 110, p. 161801, Apr 2013.

[4] R. Acciarri, C. Adams, R. An, C. Andreopoulos, A. Ankowski, M. Antonello, J. Asaadi, W. Badgett, L. Bagby, B. Baibussinov, *et al.*, "A proposal for a three detector short-baseline neutrino oscillation program in the fermilab booster neutrino beam," *arXiv preprint arXiv:1503.01520*, 2015.

[5] R. Acciarri, M. Acero, M. Adamowski, C. Adams, P. Adamson, S. Adhikari, Z. Ahmad, C. Albright, T. Alion, E. Amador, *et al.*, "Long-baseline neutrino facility (lbnf) and deep underground neutrino experiment (dune) conceptual design report, volume 4 the dune detectors at lbnf," *arXiv preprint arXiv:1601.02984*, 2016.

[6] C. Andreopoulos *et al.*, "The GENIE Neutrino Monte Carlo Generator," *Nucl. Instrum. Meth.*, vol. A614, pp. 87–104, 2010.

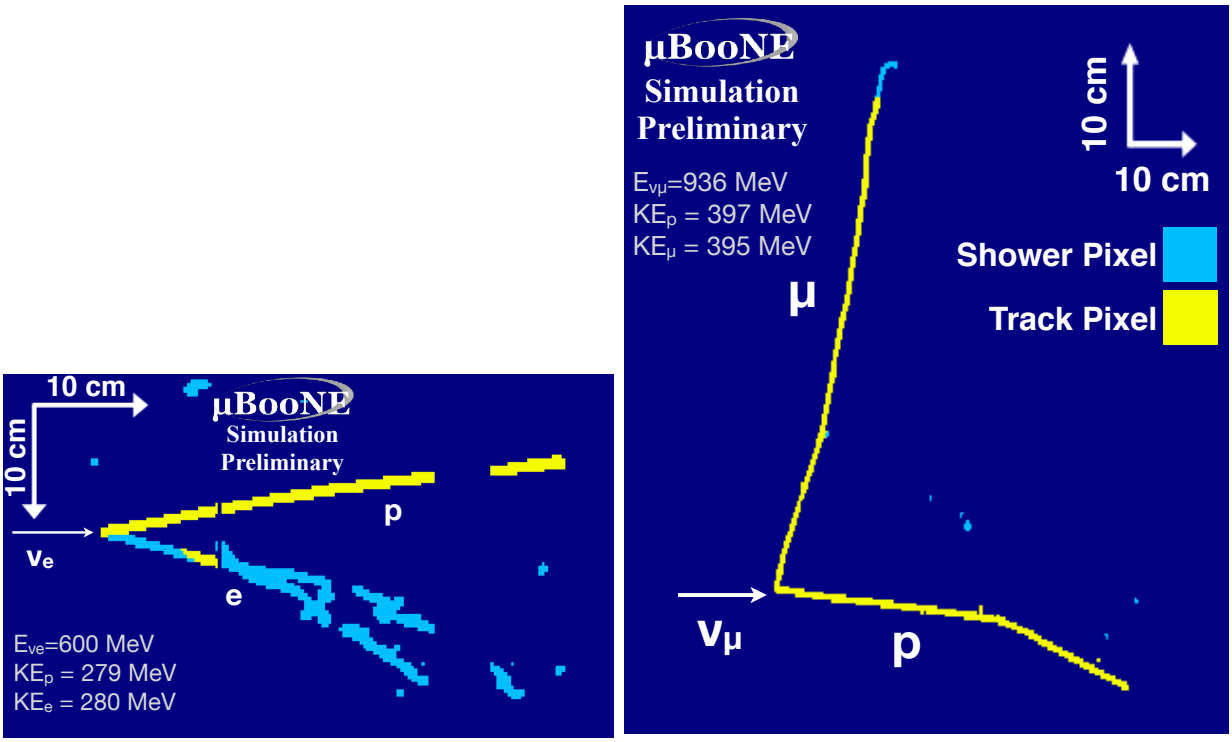
[7] D. Heck, G. Schatz, J. Knapp, T. Thouw, and J. Capdevielle, "Corsika: A monte carlo code to simulate extensive air showers," tech. rep., 1998.

[8] S. Agostinelli, J. Allison, K. a. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. . Barrand, *et al.*, "Geant4—a simulation toolkit," *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.

[9] J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. A. Dubois, M. Asai, G. Barrand, R. Capra, S. Chauvie, R. Chytracek, *et al.*, "Geant4 developments and applications," *IEEE Transactions on nuclear science*, vol. 53,

no. 1, pp. 270–278, 2006.
[10] J. Allison, K. Amako, J. Apostolakis, P. Arce, M. Asai, T. Aso, E. Bagli, A. Bagulya, S. Banerjee, G. Barrand, *et al.*, "Recent developments in geant4," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 835, pp. 186–225, 2016.
[11]
[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.
[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[14]
[15] http://opencv.org/.
[16] R. Acciarri *et al.*, "Convolutional neural networks applied to neutrino events in a liquid argon time projection chamber," *Journal of Instrumentation*, vol. 12, no. 03, p. P03011, 2017.
[17] R. Acciarri *et al.*, "Noise characterization and filtering in the microboone liquid argon tpc," *Journal of Instrumentation*, vol. 12, no. 08, p. P08003, 2017.
[18] https://github.com/LArbys/dllee_unified.
[19] http://pdg.lbl.gov/2017/AtomicNuclearProperties/HTML/liquid_argon.html.
[20] https://physics.nist.gov/PhysRefData/Star/Text/PSTAR.html.
[21] M. Anghinolfi, M. Ripani, R. Cenni, P. Corvisiero, A. Longhi, L. Mazzaschi, V. Mokeev, G. Ricco, M. Taiuti, A. Teglia, A. Zucchiatti, N. Bianchi, A. Fantoni, V. Muccifora, P. LeviSandri, V. Lucherini, E. Polli, A. R. Reolon, P. Rossi, and S. Simula, "Inclusive electron scattering from an oxygen and argon jet target," *Journal of Physics G: Nuclear and Particle Physics*, vol. 21, no. 3, p. L9, 1995.

APPENDIX



(a) (b)

FIG. 23. Two examples illustrating the SSNet-image pixel labeling. The left panel shows a $1e1p$ type event from a $\_e$ interaction. The pixels corresponding to the proton are here here correctly classified as track by the SSNet. The pixels corresponding to the electron are here mostly classified as shower by the SSNet, except for a small portion mistakenly labeled as track. The right panel shows a $1\mu 1p$ type event from a ν_μ interaction. Here both the proton and muon tracks are correctly labeled as track pixels. The muon decays into a Michel electron, classified as shower pixels. Dark blue pixels correspond to empty pixels, without charge deposition.

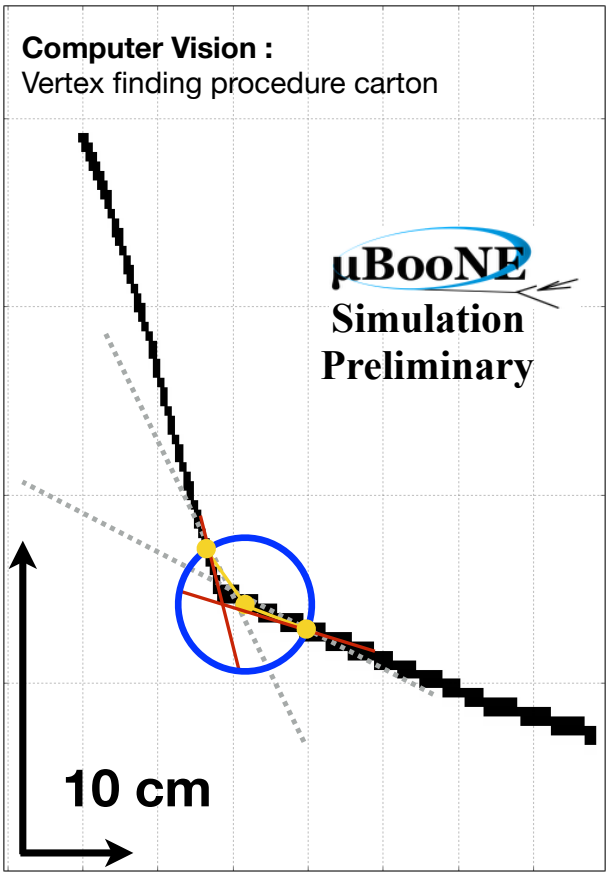


FIG. 24. Cartoon of a vertex finding procedure with the OpenCV computer vision tool. For each candidate vertex position, a circle is drawn centered at the vertex position, the tracks intersects the circle boundary at two points. The yellow lines link the track-circle crossing points to the circle center. The angle between the two yellow lines is θ. The red lines represent the local PCA approximation at the track-circle intersection. The angle between the red lines is ϕ. The gray dashed lines represent the initial PCA lines. The circle is scanned along the initial straight line PCAs and the magnitude difference of θ and ϕ is recorded per time tick. A local minimum in this magnitude difference that coincide across the three planes is a possible vertex.

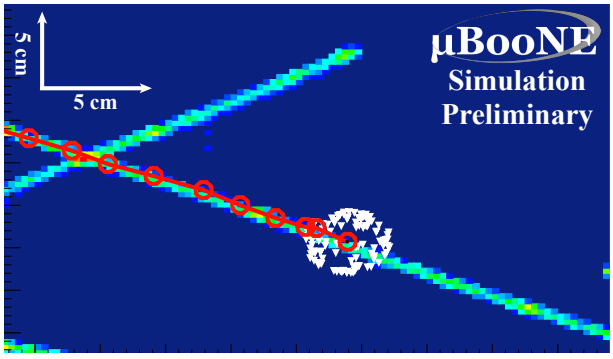


FIG. 25. Cartoon of a step in the track finding algorithm : a new point is searched for in the set of possible new points (white triangles) randomly generated. The point that is 3D consistent with charge depositions on all planes and furthest away from the last track point (red dots) is added to the track.

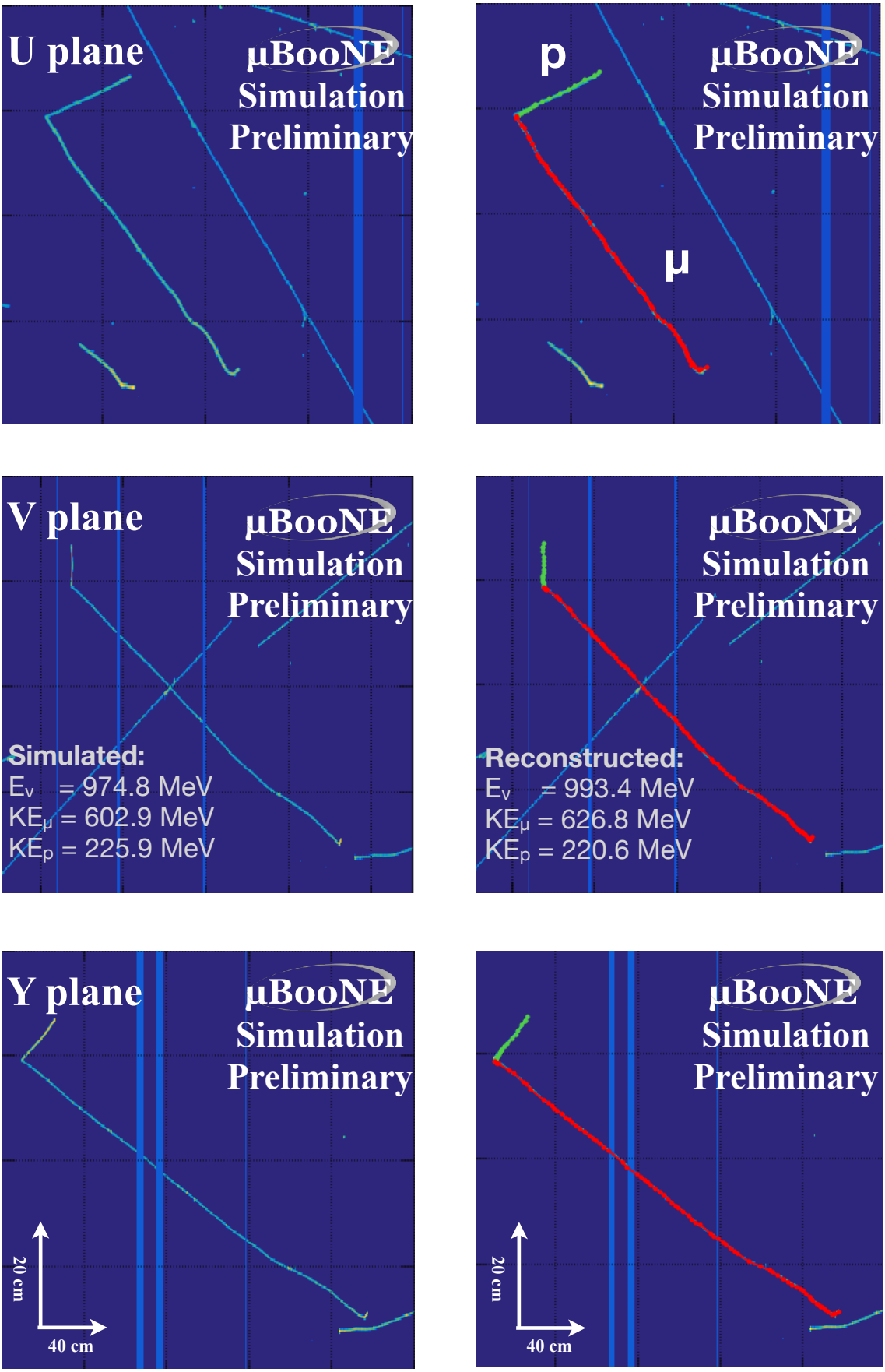


FIG. 26. Example of a reconstructed MC event : Left : ADC image of a 974.8 MeV simulated $1\mu 1p$ neutrino event producing a 602.3 MeV muon and a 225.9 MeV proton. Right : Reconstructed tracks are overlaid on top of the ADC image. The event is reconstructed as a 626.8 MeV muon (red) and a 220.6 MeV proton (black), for a reconstructed E_ν^{vis} of 993.4 MeV.