Spring 2021

Spectator Proton Detection and Reconstruction in Deep Inelastic *D(E,EP$_S$)* Scattering

David Payette
Old Dominion University, dcpayette@gmail.com

SPECTATOR PROTON DETECTION AND RECONSTRUCTION IN DEEP INELASTIC $D(E, EP_S)$ SCATTERING

by

David Payette
M.S. December 2015, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

PHYSICS

OLD DOMINION UNIVERSITY
May 2021

Approved by:

Dr. Sebastian Kuhn (Director)

Dr. Gail Dodge (Member)

Dr. Jay Van Orden (Member)

Dr. Yuan Zhang (Member)

Dr. John Adam (Member)

ABSTRACT

SPECTATOR PROTON DETECTION AND RECONSTRUCTION IN DEEP INELASTIC $D(E, EP_S)$ SCATTERING

David Payette
Old Dominion University, 2021
Director: Dr. Sebastian Kuhn

A Radial Time Projection Chamber (RTPC) was designed and installed in Jefferson Lab's Hall B as part of the BONuS12 (Barely Off-shell Nucleon Structure) experiment. The goal of BONuS12 is to accurately measure the structure function of the neutron by scattering 11 GeV electrons and detecting them with the CLAS12 spectrometer. Deuterium gas was used as an effective neutron target, and the new RTPC was used to detect low-momentum spectator protons. Protons follow a curved path in the 5 Tesla solenoid that is part of CLAS12, ionizing the He-CO2 gas in an annular drift region surrounding the target. These ionization electrons are radially drifted outwards, amplified using cylindrical GEM (Gaseous Electron Multiplication) foils and recorded using readout pads located along the entire outer face of the cylindrical detector. The particle track reconstruction software discussed in detail in this thesis uses the signals from these pads to build tracks, which are reconstructed into the drift region using the arrival times of the signals and the positions of the pads. The proton momentum is measured from the track's curvature and thus used to extract information about the struck neutron. This thesis introduces the theory of spectator tagging as an effective strategy for measuring neutron structure, by minimizing nuclear effects in the absence of a free neutron target. Along with discussing the many detectors that make up the CLAS12 spectrometer, the RTPC will be covered in detail, along with the tracking software designed to interpret the electronic signals to rebuild the low-momentum particle tracks, and fit them to extract the relevant kinematics. The results of the software, and preliminary analysis will be shown in the final chapter, as well as the discussion of possible improvements which could be made to the tracking software.

ACKNOWLEDGEMENTS

Completing this project challenged me in many ways. When I first began working for the BONuS12 group I was tasked with writing the Particle Tracking software for an RTPC. I didn't even know what that meant when I first started, and 5 years later, this paper defines how I accomplished this. I'd like to thank Jennifer for sticking with me through this entire program, a time when I was beyond stressed on a daily basis. She had to listen to me talk about work every day and even though she often didn't know what on earth I was talking about she always listened patiently, and worked hard to create a stress-free environment for me outside of work. I'd also like to thank Torri Roark. I spent the first two years of my Master's program working closely with Torri, getting coffee daily, and just in general kicking each other to make sure we were staying on track. Without her, getting my homework done, and passing the qualifiers would not have even been possible and I'm so fortunate to have made a lifelong friend. Once I began my PhD work, I began working closely with Gabriel Charles who helped me out endlessly when I was first getting started. Being able to go and vent, get together for lunches, dinners and drinks, and just laugh together made the early years of my PhD work so much more fun on a daily basis. Once Gabriel sadly moved back to France, Mohammad Hattawy joined in his place and served as a driving force and motivator for me all the way through the end of the program. His kind words at my thesis defense will never be forgotten.

Of course I have to thank Dr. Sebastian Kuhn, my thesis advisor, for dealing with me this entire time, and for teaching me invaluable lessons that will help me for the rest of my working life going forward. I went through a lot during this program, and he was always patient, kind, and understanding, especially when having to send me 100 drafts of this paper, covered in red ink. Thanks so much to my friends and family for always supporting me throughout this process, especially my closest friend Brad Beyea who must've had to listen to me talk about work hundreds of thousands of times throughout the years, and stuck with me even after moving halfway across the country, and Eric Jameson, who spent hours almost every day hanging out with me digitally and working together during the pandemic when we were all trapped inside.

I'd also like to thank all the wonderful people I got to work with on this project. Nate Dzbenski, Jiwan Poudel, Madhusudhan Pokhrel, Stephen Bueltmann, Gail Dodge, Eric Christy, Narbe Kalantarians, Carlos Ayerbe, Veronique Ziegler, Gagik Gavalian, and many many others were just amazing people to work with and get help from throughout the years,

and I'm glad to have been a part of the amazing work that all of us have accomplished. Finally, I'd like to thank the members of my thesis committee, who provided invaluable feedback to me throughout the years, and dealt swiftly, flexibly, and patiently with the difficult times during which my thesis defense was scheduled.

TABLE OF CONTENTS

Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The field of Particle Physics has been expanding ever since the introduction of atomic theory over 200 years ago. Since then, the scale at which scientists are capable of studying particles has continued to decrease by many orders of magnitude, starting from a "plum-pudding" model of how an atom is structured, then to the electron orbitals around the nucleus, continuing into the nucleus to study how nucleons (protons and neutrons) distribute themselves and interact. Continuing deeper into protons and neutrons, we learned how they consist of 3 valence quarks, and a sea of quarks and gluons. As we continue probing further and further, experiments continue to reach higher and higher energies to study these structures. Of particular interest to our experiment is the structure of protons and neutrons, or more specifically, the way that quarks tend to distribute themselves inside these nucleons. The distribution of quarks inside of nucleons can be expressed in the form of Parton distribution functions, and the goal of our experiment is to extract these functions, specifically for the neutron.

In order to study nucleons and their structure, a high energy beam of charged particles (in our case, electrons) is used to scatter off the quarks inside the nucleon. In order to perform such an experiment, a target containing a high density of such nucleons is required. For protons, their structure is well understood. This is simply due to the fact that a target of protons is easily achieved using Hydrogen. Hydrogen nuclei only consist of protons, and are highly stable, so Hydrogen makes for a great proton target. Many experiments have been conducted to study the structure of protons because of this, and the so-called structure functions of the proton are well known. However, the same cannot be said for neutrons. The challenge with neutrons is that, unlike for protons, a neutron target is not easily achieved. There are no nuclei which only consist of neutrons, and neutrons which are not bound have a lifetime of about 15 minutes. As a result, it is a challenging feat to study neutron structure. Targets containing both protons and neutrons may be used to study the neutron, but unfortunately, due to their bound nature, the results are often hard to interpret.

This is the goal for our experiment, to study the structure of the neutron. In order to do so, we used an effective neutron target. For this purpose, the Deuterium or ^{2}H nucleus, which consists of only a proton and a neutron, is used as a target. In order to limit our

interaction with the proton, we need to study collisions in which we detect protons travelling backwards after the collision. Under such circumstances, the probability that the neutron was struck is very high. These protons are called "spectators" to the collision, and their interaction in said collision is minimal. In this way, we are the closest to scattering off of a "free" neutron target.

By detecting the proton which was a spectator to the collision, and the electron scattered on the neutron, some straightforward kinematics can be used to understand the effect on the neutron. The challenge lies in detecting the proton travelling backwards. In such conditions, the proton will travel quite slowly, and at such a backward angle, that in typical particle collision detector arrangements, which often detect particles travelling forward, or radially outward at high speed, the proton would be completely missed. For this reason, a custom detector designed specifically for detecting extremely low-energy protons was used. In the rest of this thesis, I will describe the experiment conducted to extract the structure functions. I will explain how this custom detector works, and how we can understand the signals we receive from the detector. The underlying physics to explain the complexity of scattering off a bound neutron, the BONuS12 experiment, the software I wrote to reconstruct the path charged particles take in our detector, the way we monitored our experiment and the way we analyzed data will be discussed in detail in this thesis. In the end, I will present some preliminary results.

CHAPTER 2

SCIENTIFIC BACKGROUND

The standard model of Particle Physics [1] consists of elementary particles divided into 2 fundamental groups: fermions and bosons. fermions, also known as the building blocks of all matter, consist of leptons and quarks. All atoms are made up of fermions. Bosons are also referred to as force carriers. These are the particles which represent the exchange of the four fundamental forces in the standard model: strong, weak, electromagnetic, and gravity. fermions and bosons are separated by their spin, namely fermions have a half-integer spin, and bosons have integer spin. Spin does not refer to particles actually spinning, but rather the total angular momentum of the particle. Leptons and quarks are differentiated by their charge, with quarks having charges of 2/3 or -1/3, and leptons having charge -1, or, in the case of neutrinos, charge 0. Non-elementary or "composite" particles which are made entirely of quarks are called hadrons [2]. Hadrons are separated into two groups, baryons and mesons. Baryons consist of 3 valence quarks, and mesons consist of a valence quark-antiquark pair. Valence quarks are the quarks used to determine the net charge of a composite particle. A nucleus for instance consists entirely of baryons called nucleons (protons and neutrons). The proton consists of two "up" quarks (u) and a "down" quark (d). These names have nothing to do with direction but are rather the identifying "flavors" of quarks, which divide the quarks by mass and charge. U quarks each contribute a charge of 2/3 and d quarks contribute a charge of -1/3, which results in the net charge of a proton of +1. The neutron however consists of two d quarks and a u quark, which results in a net charge of 0. This is the underlying reason that protons have charge and neutrons do not. Hadrons are not entirely made up of valence quarks however. In fact, aside from the valence quarks, there is a large distribution or "sea" of quarks and gluons, gluons being the strong force carrying boson. These "sea" quarks are made up of equal numbers of quarks and antiquarks and are constantly annihilating, resulting in a gluon which then splits into a quark-antiquark pair, and the cycle continues. For this reason the sea quarks never contribute to the overall charge of the hadron. It is precisely the distribution of valence quarks in nucleons that is of great interest to our experiment. The goal of our experiment is to extract so-called structure functions, which are related to the quark distribution or quark momentum contribution to the overall nucleon momentum. In order to extract such functions, we need to perform

a scattering experiment. The goal of a scattering experiment is to scatter particles off of one another, and based on the resulting kinematics, it is possible to determine the internal structure of the particle scattered from.

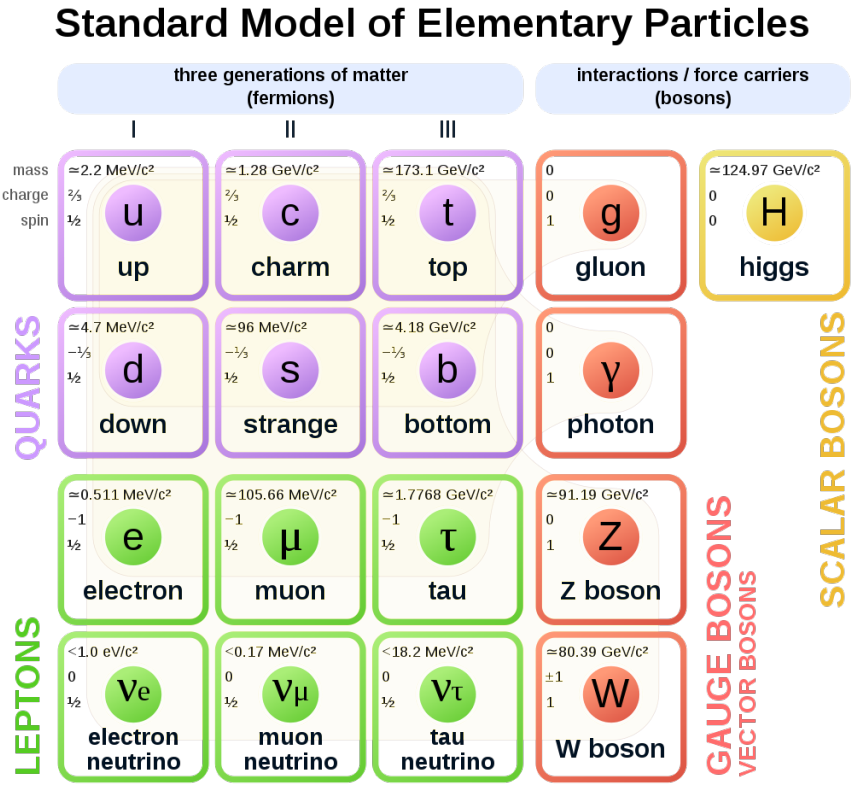

FIG. 1: The standard model of Particle Physics. [3]

2.1 TYPES OF PARTICLE SCATTERING

Particle scattering, or collisions in general, can be classified as either elastic or inelastic. Elastic collisions are collisions in which the structures involved do not undergo any changes, or in other words, the kinetic energy of the collision is conserved. In an inelastic collision however, kinetic energy is no longer conserved. This could be the result of one or more bodies in the collision fragmenting into smaller pieces, or an excitation of one or more of the particles. It is possible for such particles to fragment into many daughter particles, in which case the collision is classified as deeply inelastic. For our purposes, we are most interested in scattering on quarks inside nucleons, which would classify as deeply inelastic scattering. [4]

In particle scattering experiments, the momentum before and after the collision is used

to describe the state of the particle. Traditionally, when relativity is not considered, momentum is defined in 3-dimensions as $\mathbf{p} = (p_x, p_y, p_z)$. However, in most particle scattering experiments, the particles travel at such high energies that relativity becomes increasingly important, and for this reason, and for all future calculations, momentum will be defined instead in a 4-dimensional space-time way, namely $\mathrm{p} = (p_0, \mathbf{p})$. The p_0 term is defined as E/c where E is the energy and c is the speed of light. In all further calculations, we will use so-called "natural" units, where $c = 1$, as this greatly simplifies the results.

2.1.1 DEEP INELASTIC SCATTERING

For the case of studying nucleon structure, instead of scattering off the nucleon as a whole, we instead wish to scatter off individual quarks inside the nucleon. This process of scattering off quarks is known as Deep Inelastic Scattering. The process is considered inelastic because it results in excitations or "resonances" of the struck nucleon, as well as to fragmentation of the nucleon. In either case, the final (unobserved) state of the struck nucleon has a center-of-mass energy W which is greater than the initial nucleon mass, M. The differential cross section for inelastic scattering, with a final state electron moving with energy E' in the direction of the solid angle $\mathrm{d}\Omega$, can be written as follows

$$\frac{\mathrm{d}\sigma}{\mathrm{d}E'\mathrm{d}\Omega} = \frac{4\alpha^2 E'^2}{Q^4}\left[W_2(\nu, Q^2)\cos^2\frac{\theta}{2} + 2W_1(\nu, Q^2)\sin^2\frac{\theta}{2}\right]. \tag{1}$$

The differential cross section can be characterized as the scattering probability. Here, $\alpha = 1/137$ is the so-called "fine structure constant". For an incoming electron with momentum $p = (E, \vec{p})$ and the scattered electron with momentum $p' = (E', \vec{p'})$, E' is the energy of the scattered electron, $Q^2 = (p - p')^2$ is the squared four-momentum of the virtual photon exchanged in the collision, $\nu = E' - E$ is the energy transferred by the virtual photon, and θ is the scattering angle. The functions W_1 and W_2 are referred to as "structure functions". As mentioned before, in Deep Inelastic Scattering, we are effectively elastically scattering on quarks. When $W \gg 2$ GeV and $Q^2 \gg 1$ GeV^2, we can be sure that we are no longer scattering on the nucleon as a whole, but rather the quarks inside the nucleon. In such cases we can rewrite the differential cross section in terms of structure functions which only depend on a scaling variable known as Bjorken-x which is defined as

$$x_B = \frac{Q^2}{2M\nu}. \tag{2}$$

x_B is effectively the representation of the fraction of the nucleon's total momentum carried

by a quark. In the limit of large Q^2 (and thus large x), $W_1 \to F_1(x)/M$ and $W_2 \to F_2(x)/\nu$. These structure functions are fairly well known for the proton, [5] due to the readily available and long lasting proton target, where many experiments have been performed to scatter off quarks in the proton and study the structure functions $F_1^p(x)$ and $F_2^p(x)$. However, due to the lack of a reliable and long lasting neutron target, these functions are less well known for the neutron. This is particularly unfortunate since we could use our knowledge of the neutron's structure functions to learn more about the proton's structure functions, as the valence quarks are simply swapped (see below), and we can form relationships between the proton and neutron structure functions in an effort to more effectively extract them from data. The two structure functions, at large enough Q^2, can be related by the Callan-Gross relation as

$$2xF_1(x) \approx F_2(x), \tag{3}$$

where $F_1(x)$ and $F_2(x)$ are the structure functions and I have dropped the B in x_B for simplicity. Because of the relationship between the two, it is simpler to study F_2. F_2 can be written in terms of the parton distribution functions for quarks as follows

$$\frac{1}{x}F_2^N = \left(\frac{2}{3}\right)^2 [u^N(x) + \bar{u}^N(x)] + \left(\frac{1}{3}\right)^2 [d^N(x) + \bar{d}^N(x)] + \left(\frac{1}{3}\right)^2 [s^N(x) + \bar{s}^N(x)]. \tag{4}$$

Here N refers to the nucleon in question (p or n). One can use the fact that

$$\begin{aligned} u(x) \equiv u^p(x) = d^n(x), \\ d(x) \equiv d^p(x) = u^n(x), \\ s(x) \equiv s^p(x) = s^n(x), \end{aligned} \tag{5}$$

to find some simple relationships which will be used to extract both $u(x)$ and $d(x)$ from measurements on the proton and neutron. For comparison, it is useful to split these nucleon structure functions into valence and sea contribution terms as follows

$$\frac{1}{x}F_2^p = \frac{1}{9}[4u + d] + S, \tag{6}$$

where $u = u^p = d^n$ and $d = d^p = u^n$ are the PDFs only for the valence u and d quarks in the proton. S refers to the sea quarks. As $x \to 1$, we are probing a single quark which carries a large fraction of the proton's momentum, which is likely to be a valence quark. In these circumstances, S becomes negligible and we can rewrite the ratio of structure functions as

follows

$$\frac{F_2^n}{F_2^p} \approx \frac{u+4d}{4u+d}. \tag{7}$$

This only works at large enough x, thus the approximation. For $x \to 0$ this ratio approaches 1. The particular interest in the limit $x \to 1$ comes from the fact that several approximations to the fundamental theory of strong interactions, QCD, make specific predictions for that limit that we want to test.

2.1.2 STUDYING NEUTRONS

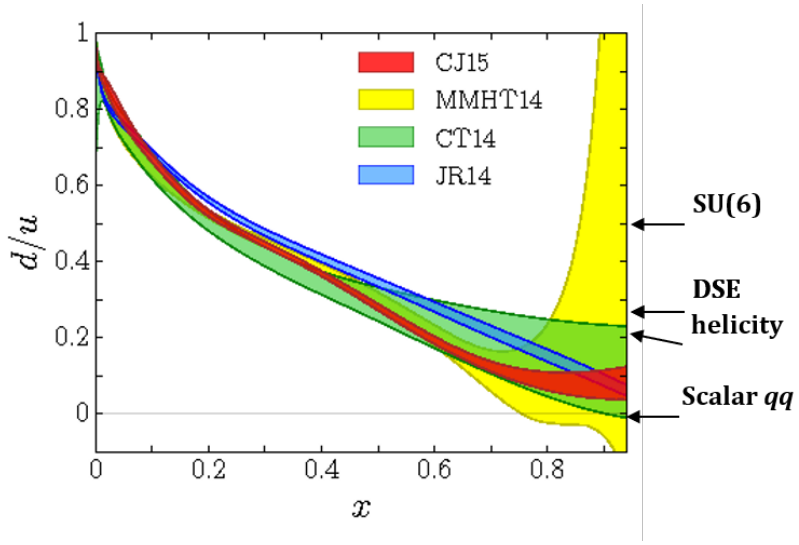

FIG. 2: The ratio of d/u as a function of Bjorken-x, with several different parametrizations of the world data (shaded bands), and showing the uncertainties of each. Notice how for large x, the uncertainty and disagreement grow due to the lack of high energy scattering data on neutron targets. On the right hand side, several theoretical predictions for the limit of d/u as $x \to 1$ are indicated. [6]

Before diving deep into the techniques used in BONuS12 to study neutrons, it is useful to clarify the importance of studying the neutron, and how measurements in BONuS12 would contribute to the field moving forward. [7] The main reason for studying neutron structure is simply that, due to the challenges that come with neutron scattering experiments, there is much uncertainty and competing theoretical predictions on exactly what the neutron structure looks like, in stark comparison to the proton. Taking a look at FIG. 2, the measurement and prediction of d/u becomes uncertain especially in the region as $x \to 1$. More specifically,

this ratio is affected by the lack of good data on d. However, due to the isospin symmetry of protons and neutrons, it is possible to measure both u and d quark distributions by measuring both proton and neutron structure functions.

By measuring the u quark distribution in the neutron, we can use such a measurement as a representation for the d quark distribution in the proton, and by measuring each we know the complete picture for the proton and the neutron. Due to the high-energy upgrade at Jefferson Lab's CEBAF, it is possible to study this region of high x, using electron energies on the order of 10 GeV, in order to cleanly probe the quarks of the neutron, in an effort to more accurately measure this ratio. Predictions for the expected data on this ratio using BONuS12 can be seen in FIG. 3. Clearly, we expect to rule out or confirm at least some of the existing theoretical predictions for $x \to 1$.

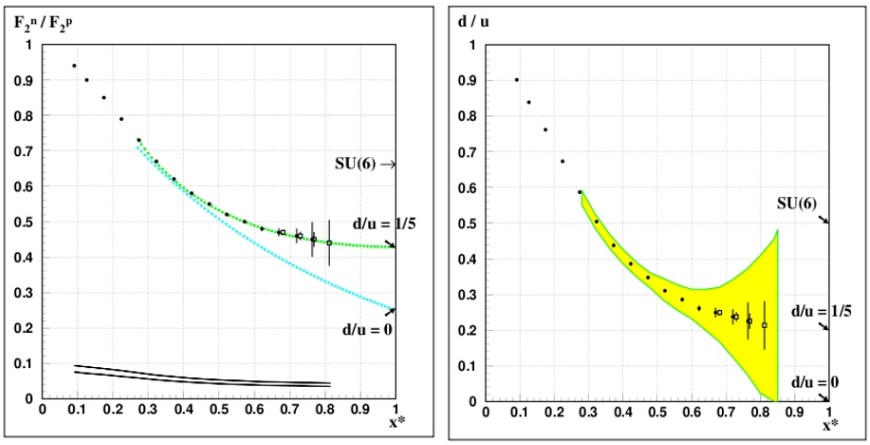

FIG. 3: The ratio of F_2^n/F_2^p as a function of Bjorken-x, with overlaid predictions including expected measurement uncertainties of the ratio up to large x. [7]

Additional questions to answer by studying neutrons more accurately lie in the nucleon resonance region, at large-x and Q^2 on the order of 1 GeV$^2/c^2$. By studying the ratio of σ_n/σ_p, it is possible to constrain the isospin structure of the total cross section. The neutron structure functions can also be used to test Bloom-Gilman duality (parton-hadron duality) for the neutron. [8] It is also possible to use these data to calculate the contribution of nuclear corrections in the deuteron, which would determine the magnitude of the EMC (European Muon Collaboration) effect [9], or in other words, the effects of nuclear binding on the quark momentum distribution of nucleons. In order to accurately extract the F_2^n structure function, we need a method of scattering high-energy electrons on effectively free neutrons. However,

due to the bound nature of the neutron in the deuteron, a careful approach should be taken to minimize the nuclear binding effects. [10]

2.1.3 DIS ON THE DEUTERON

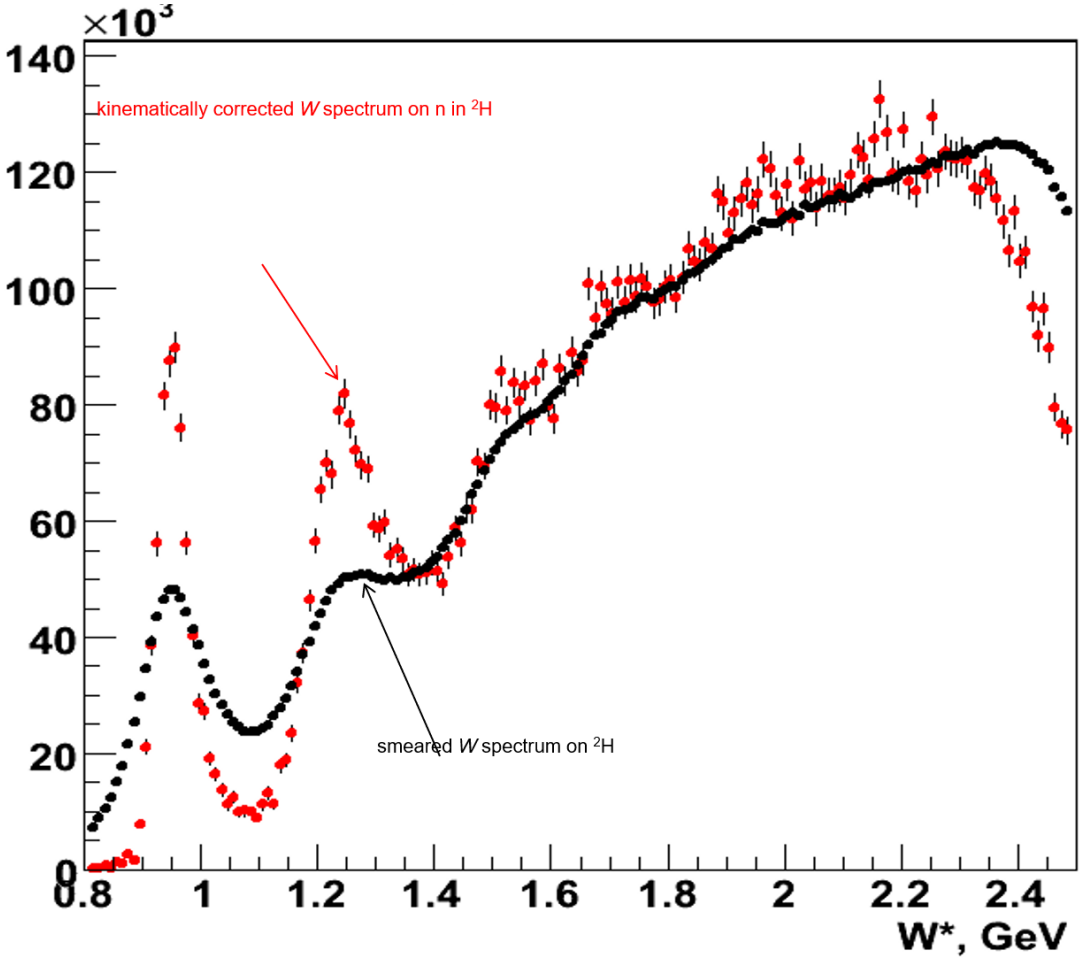

FIG. 4: A histogram showing the smeared (black) and unsmeared (red) spectrum in the deuteron. Notice how the proton and neutron peaks are far less pronounced when bound together in the nucleus. This is the effect known as Fermi smearing. [11]

Recall that the deuteron nucleus contains both a proton and a neutron bound together. As a result, both of these nucleons are moving relative to each other with some momentum distribution, with their combined momenta being equal to the total momentum of the deuteron, in other words, $p_n + p_p = p_D$. With this motion, the kinematics of each scattering event are smeared out leading to a disappearance of the sharp resonance features one

observes on free nucleons (known as "Fermi Smearing", see FIG. 4). Because of the fact that in scattering we are not able to distinguish whether we scatter off the neutron or the proton, we instead look at the sum of both, and have to subtract the proton contribution in an effort to study the neutron's contribution to the total momentum. However, this is non-trivial because of both the Fermi smearing and the binding effects which will modify the nucleon's structure when compared to a free nucleon. Even if we were to be sure that we could scatter on the neutron, these binding effects would still distort the picture. If we consider the case when the deuteron is at rest in the lab frame, then we can write,

$$\begin{aligned} p_D &= (M_D, \vec{0}), \\ p_p &= (E_p, \vec{p_p}), \\ p_n &= (E_n, \vec{p_n}). \end{aligned} \tag{8}$$

Momentum conservation tells us that $\vec{p_p} + \vec{p_n} = 0$ or in other words that $\vec{p_p} = -\vec{p_n}$. However, the energy of the bound neutron E_n must be less than the energy of a free neutron with the same momentum to conserve total energy, $E_p + E_n = M_D$. The difference is referred to as "off-shell-ness" of the bound neutron and is expected to affect its structure. In order to get as close to a free neutron as possible, we need to minimize E_p. This will result in a neutron which is closer to mass shell. For this reason, we are highly interested in events with low momentum protons, and the effort to select such protons is known as spectator tagging. [10, 12] In doing so we can minimize the binding effects on the neutron.

2.1.4 SPECTATOR TAGGING

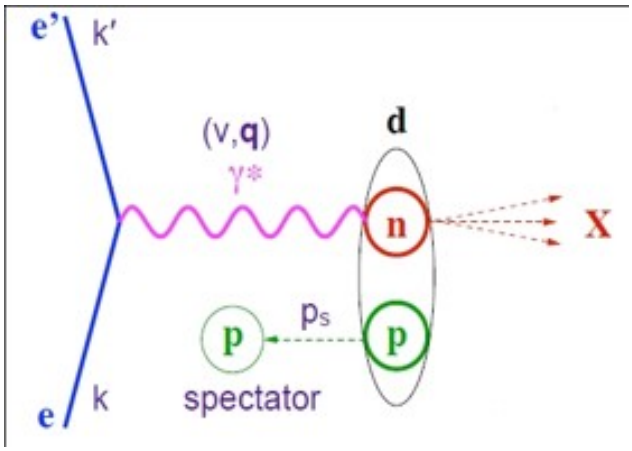

FIG. 5: A Feynman diagram of spectator tagging, showing the electron scattering off the neutron bound to the proton in the deuteron nucleus.

As mentioned in the introduction, the main issue with studying F_2^n is the lack of a free neutron target. In a universe in which neutrons didn't decay in about 15 minutes, free neutron targets could be used to perform high-energy scattering experiments, not limited by the effects of nuclear binding. The best we can possibly do for now is to form a target in which the neutron is bound to the proton in the deuteron, and look specifically for experimental conditions in which we know the effects which may distort the picture of neutron DIS are minimized. This is the reason for Spectator Tagging. The goal is to scatter on quarks in the bound neutron such that the outgoing proton has a low momentum and backward angle. In such circumstances the neutron is nearly on mass shell, or effectively a free neutron. By focusing on low-momentum protons, we can ensure that the neutron energy is as close as possible to the on-shell value. Selecting backward protons eliminates events where the proton is the struck nucleon, in which case it would be moving in the direction of the momentum transfer. Measuring the kinematics (p_p) of the spectator proton allows us to deduce the initial motion of the struck neutron, and hence correct for that motion, recovering the sharp peaks seen in FIG. 4 that were otherwise washed out. Also, even if we are scattering off the neutron, it is possible for the proton to interact with the final state product following the collision, also known as final state interactions (FSI). By studying these FSI it is possible to deduce the necessary proton kinematics to minimize these effects (approaching the so-called Plane Wave Impulse Approximation where the spectator proton escapes undisturbed).[13]

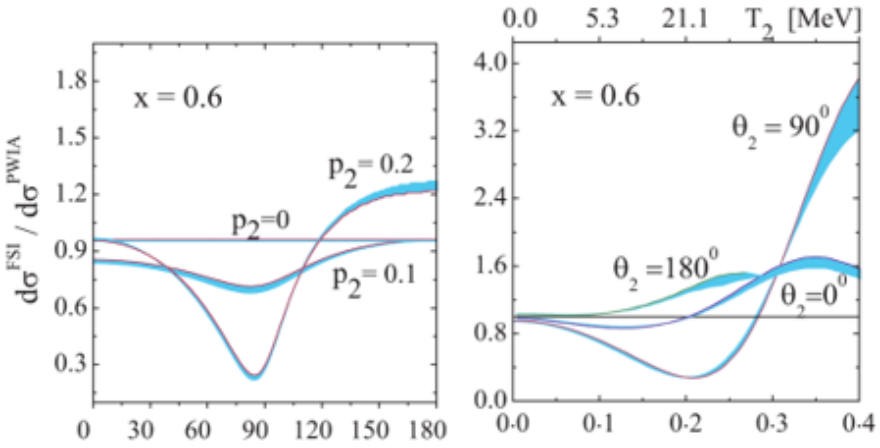

FIG. 6: The ratio of the differential cross section for final state interactions and plane wave impulse approximations, as a function of the angle between the outgoing proton and the momentum transferred from the scattered electron, θ_{pq} (Degrees) and spectator proton momentum (GeV), respectively. Notice how in the regime of large scattering angle θ and low momenta < 0.15 GeV the ratio approaches 1. [13]

One such study is shown in FIG. 6, which shows the ratio of differential cross sections including Final State Interactions (FSI) versus the Plane Wave Impulse Approximation (PWIA). This ratio approaches 1 in regions where the spectator proton momentum is low (< 0.15 GeV) and opposite to the incoming virtual photon. By detecting such protons as spectators to neutron scattering we can limit the binding effects, and final state interaction effects on the neutron. As a result, the deuteron can be treated as a nearly-free neutron target, with minimal binding effects. In order to study neutrons in this way, a detector should be constructed which is capable of detecting not only low-momenta protons, but also such low-momenta protons which are travelling at backward angles. For this, the detector must have a large scattering angle acceptance, and use a gas mixture and electronics which are highly sensitive to lower momentum particles which would ionize such a gas. For this the Radial Time Projection Chamber was constructed, and is explained in more detail in the following sections.

CHAPTER 3

THE BONUS12 EXPERIMENT

The Barely Offshell Nucleon Structure experiment at 12 GeV (BONUS12) is a higher energy version of the original BONuS6 experiment [12, 14–17] which was executed in 2006 prior to the 12 GeV energy upgrade at Jefferson Lab. For the new experiment at a much higher electron beam energy, a new Radial Time Projection Chamber (RTPC) was constructed, which has a larger detection region and a more complete angular acceptance than the previous detector. The goal of the experiment is to detect low momentum backward angle protons which are spectators to the semi-inclusive DIS interaction $D(e, e', p_s)X$ and extract the structure function ratio F_2^n/F_2^p. The experiment was conducted in experimental Hall B of the Thomas Jefferon National Laboratory (JLab). JLab accelerates electrons up to energies of 12 GeV using two linear accelerators (LINACs) which accelerate electrons in up to 5 passes before traveling into 4 experimental halls, A, B, C, and D. The BONuS12 experiment (and BONuS6 experiment) were conducted in Hall B. Electrons traveling into Hall B will scatter off a suitable target and then are detected in the CEBAF Large Acceptance Spectrometer (CLAS12) which consists of several subsystems at a large range of forward and central angles. The RTPC detector constructed for BONuS12 is placed at the center of the Spectrometer. First, we will discuss the subsystems that make up the CLAS12 spectrometer, and then the RTPC detector built for BONuS12.

3.1 CEBAF LARGE ACCEPTANCE SPECTROMETER

The CLAS12 spectrometer consists of 14 subsystems divided into two groups, the Forward Detector, and the Central Detector. The Forward Detector is responsible for covering a range of scattering angles of 5 to 40 degrees, with the Central Detector covering a range of 40 to 125 degrees. The Forward Detector is divided into 6 subregions or "sectors". The RTPC was installed inside the Central Detector and replaces the detectors usually comprising the central vertex tracker. In addition to the RTPC, the Central Detector also houses a Central Time of Flight (CTOF) detector, the Central Neutron Detector (CND), the Forward MicroMEGAS Tracker (FMT), and a Solenoid magnet responsible for generating the magnetic field which curves the charged particles in the RTPC. The Forward Detector consists of the Torus magnet responsible for bending forward-angle charged particles inside each of 6

sectors towards or away from the beam line. The Forward Detector also houses the Forward Time of Flight (FTOF) detector, the Low and High threshold Cherenkov Counters (LTCC and HTCC), the Ring Imaging Cherenkov Counter (RICH), the Drift wire chambers, and the Electromagnetic Calorimeter (ECal). Each detector is discussed in more detail below, followed by the RTPC built specifically for the BONuS12 experiment. [18]

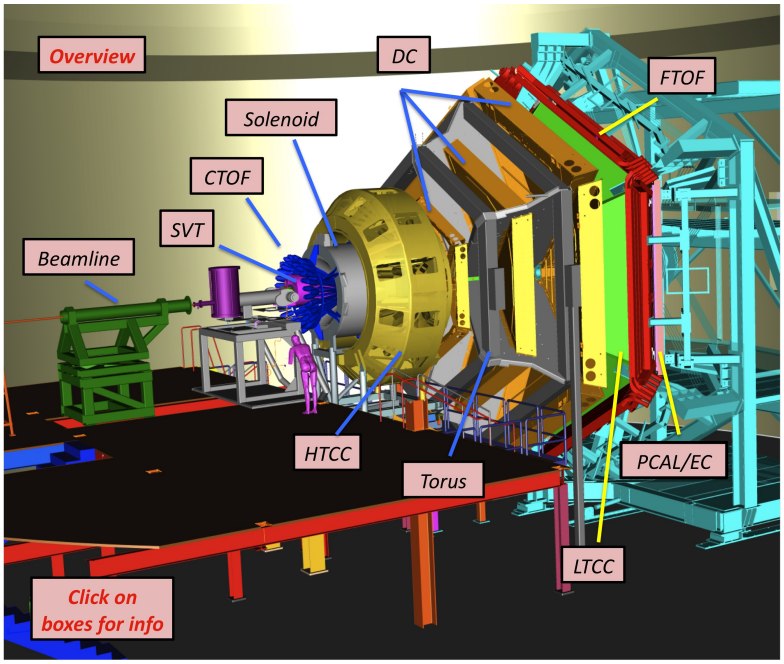

FIG. 7: The CEBAF Large Acceptance Spectrometer after the 12 GeV upgrade (CLAS12), located in experimental Hall B of Jefferson Lab features 14 subsystems divided into the forward detector and the central detector. Each subsystem is discussed in more detail in the text.[18]

3.1.1 SOLENOID AND TORUS MAGNETS

The Torus and Solenoid magnets in combination are responsible for guiding particles travelling at different angles within the appropriate subsystems of CLAS12. The solenoid magnet operates at a field of up to 5T along the beam direction, and curves the paths of charged particles in the Central Detector region, where the RTPC is housed. The solenoid magnet also protects the Forward and Central Detector from background electrons, also known as Moller electrons, which are a result of electron-electron collisions, by directing

them into the center near the beamline towards the a conical shield (the so-called Moller cone). During BONuS12, the solenoid was run with fields between 3.5 and 4 T. The torus magnet consists of 6 coils located between the six sectors of the FD and operates at a field up to 3.5T, curving charged particles towards or away from the beam line. [19]

3.1.2 DRIFT CHAMBERS

The Drift Chambers are divided into six sectors, with each sector having 3 layers, each of which is divided into 2 superlayers with each having stereo angles of $+6$ and -6 degrees. The Drift Chambers are used for charged particle reconstruction. The chambers consist of a complex and dense arrangement of field and sense wires, totaling 24192 sense wires. The sense wires are each surrounded by 6 field wires in a hexagonal pattern. The chambers are filled with a gas combination of 90% Argon and 10% CO_2. When a particle travels through the gas, it ionizes electrons which are drawn to the nearest sense wires. The distance between the ionization and the nearest sense wire can be deduced from the measured drift time, and allows for a path to be drawn which is tangential to the surfaces of constant distance from each field wire. The combined information from all superlayers allows for a accurate reconstruction of a charged particle's path as it travels through the layers of the Drift chambers, with sub-millimeter spatial resolution. [20]

3.1.3 TIME OF FLIGHT DETECTORS

The Central and Forward detectors both house time of flight detectors. The purpose of time of flight detectors is to identify particles based on their arrival time from the vertex of the scattering. Due to the fact that like-momenta particles travel at different speeds due to their mass, the particles will arrive at the TOF detectors at different times, and these arrival times can be used to distinguish charged particles from one another. These detectors both consist of several scintillation counters. The resolution of the FTOF improves at lower scattering angles down to a resolution of 80 ps. The CTOF resolution is as low as 60 ps. [21] [22]

3.1.4 LOW AND HIGH THRESHOLD CHERENKOV COUNTERS

Cherenkov counters take advantage of a phenomenon called Cherenkov radiation, which is light emitted when a particle travels faster than the velocity of light in a medium. This phenomenon is a function of the mass of the particle, which allows the Cherenkov Counters

to have mass thresholds to isolate certain particle mass ranges. The two Cherenkov counters are filled with different gases which limit which particles will emit cherenkov radiation as they travel through the detector. The high threshold counter is most sensitive to electrons. The low threshold counter is also sensitive to pions. The HTCC covers all 6 sectors, while only two sectors were equipped with LTCCs. One more sector houses a Ring Imaging Cherenkov Counter that can separate all particle species based on the opening angle of the Cherenkov light cone. [23]

3.1.5 ELECTROMAGNETIC CALORIMETER

The Electromagnetic Calorimeter consists of three layers in each of the 6 sectors, the preshower calorimeter (PCAL) and the inner and outer calorimeters (EC_{in} and EC_{out}). These calorimeters detect the amount of energy lost by particles as they travel through alternative layers of lead and scintillators. Electrons and photons initiate electromagnetic showers inside the lead leading to large signals in the scintillators. The scintillator layers in between each layer of lead are rotated 120 degrees respective to each other, forming three angles of scintillators (called U, V, and W), which provides some information about the path the particle travelled through the sector. The combination of HTCC and EC provides a highly selective identification for scattered electrons. [24] [25]

3.2 RADIAL TIME PROJECTION CHAMBER

The CLAS12 Spectrometer previously described is used as part of the BONuS12 experiment. However, as explained in the section on Spectator Tagging, we are also interested in the backward-angle low-momentum protons which cannot be detected due to the acceptance of the CLAS12 spectrometer's central detection region. For this purpose, BONuS12 uses a state-of-the-art Radial Time Projection Chamber, which is capable of detecting such spectator protons travelling with low momentum and backward angles relative to the beam direction. [7]

3.2.1 CONCEPT AND DESIGN

The Radial Time Projection Chamber (RTPC) used in the BONuS12 experiment is a evolution of the RTPC used in the previous BONuS6 experiment. The detector is cylindrical in design, and is placed along the beam axis in the center of the solenoid, such that beam electrons will collide with the deuterium gas target in the center of the detector, and the

forward scattered particles will be detected by the Forward Detector in the CLAS12 Spectrometer. The target consists of a 6 mm diameter tube with 63 μm thick Kapton walls, filled with 7 atm of deuterium gas. Protons or other particles that are also ejected at larger angles in the collision will enter the drift region of the RTPC beginning at a radius of 3 cm from the beamline.

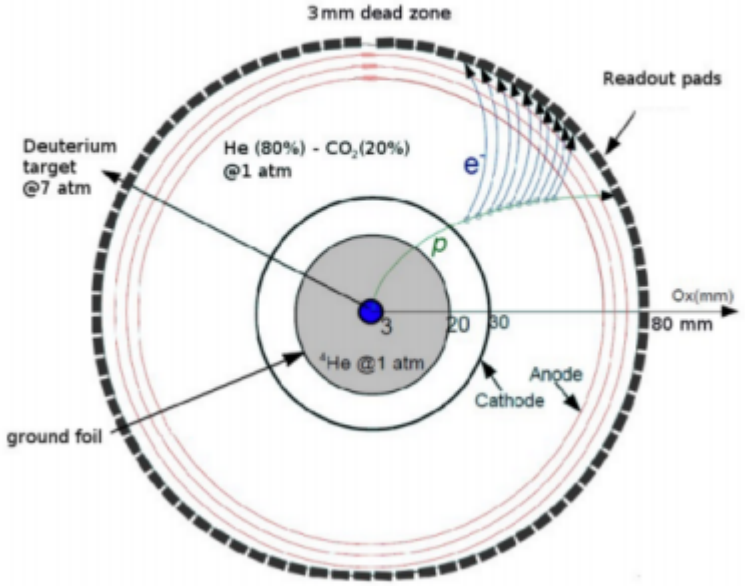

FIG. 8: This is a cross-sectional view of how the custom-built RTPC detector works. Here, a proton labelled p travels through the drift gas and ionizes electrons which travel out to the readout pads. The purpose of the reconstruction is to sort these ionized electrons and reconstruct these ionization points.

These particles will travel through the drift gas, a mixture of 80% Helium and 20% CO_2, following a curved path due to the influence of the solenoid magnetic field. As charged particles travel through the gas, they will ionize electrons. The potential difference (3000-4000V) between the negative potential applied to the Cathode, and the Anode located at 7 cm causes these ionization electrons to drift radially outward, curving due to the magnetic field. At the outer edge of the drift region, the electrons avalanche at the Gaseous Electron Multiplier (GEM) foils. There are 3 GEM foils located at 7 cm, 7.3 cm, and 7.6 cm. These GEMs consist of 50 μm Kapton foils with copper cladding on both sides, at a potential difference of about 300 V. 70 μm holes at a 150 μm pitch form regions of extremely intense

electric fields. Electrons approaching these holes trigger avalanches which result in about 100 times as many electrons being emitted on the downstream side of each. 3 GEM foils in series bring this number all the way up to approximately 100,000 electrons per primary ionization electron.

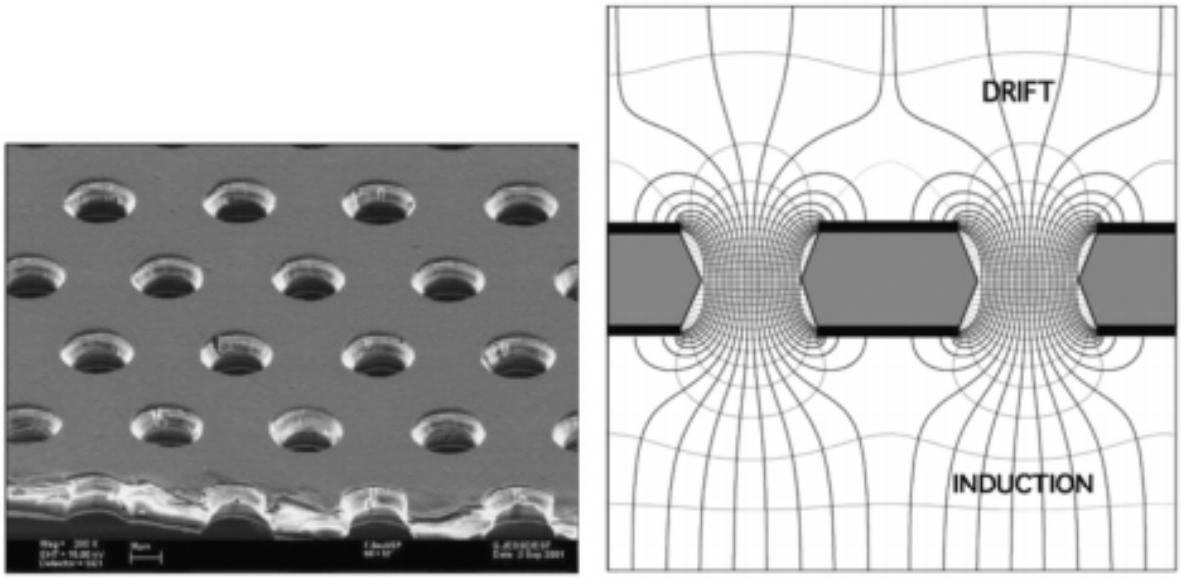

FIG. 9: The left image shows a microscopic photo of the GEM foils. The foils are 50 μm thick and include holes which are 70 μm in diameter. The right image shows the electric field lines in each of the holes of the GEM foils. [14]

These electrons are detected by the readout pads on the outer surface (mantle) of the detector, at a radius of 7.9 cm. The readout pads cover the entire inner face of that surface. There is a single 3 mm dead zone where the pad board is glued together. The pads are 4 mm long in the z-direction, and cover 2 degrees each of the surface of the detector. There are 17,280 pads divided into 180 rows in ϕ, and 96 columns in z. This amounts to 384 mm in z, and 360 degrees in ϕ. Each row of pads is shifted by 1 mm in z. The layout of this pad board is of utmost importance to the tracking. The geometry of the pad board and the signal detected on each pad is used for the tracking software in order to rebuild particle tracks. This is because the ionization electrons drifting away from the point of ionization form a sort of "roadmap" that can be seen on the pad board, showing the relative path

a particle traveled through the drift gas almost as a projection onto the outer face of the RTPC. The projection has a phi-shift due to the magnetic field, which must be corrected. The arrival time of these ionization electrons is then used to trace back to the original point of ionization, effectively rebuilding the track by fitting these ionization points in 3D space.

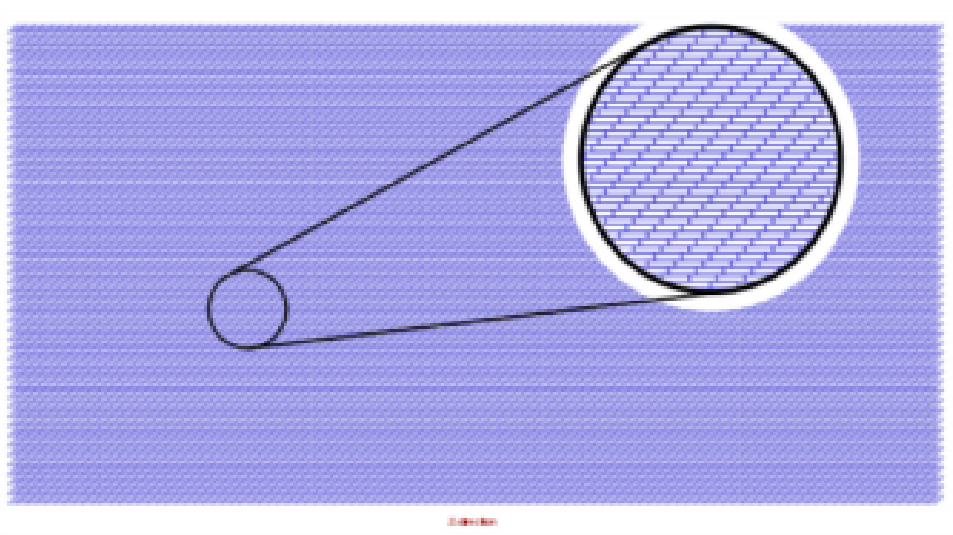

FIG. 10: This is a rendering of what the readout pad board looks like. Notice the stagger in each row of 1 mm. A section of the pad board is zoomed in to make it easier to see.

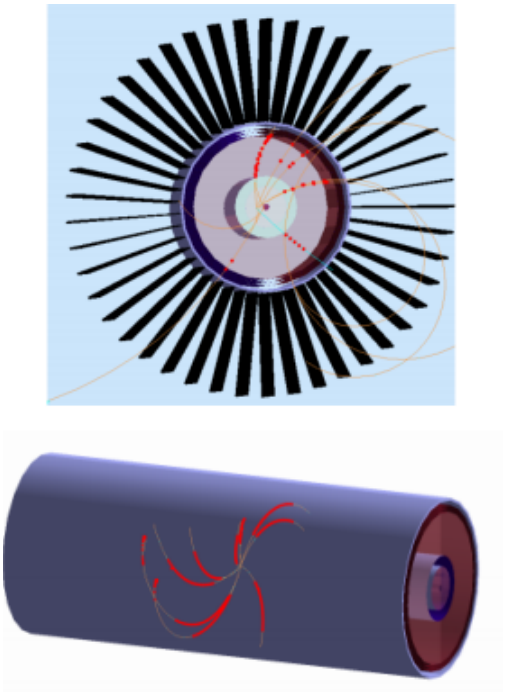

FIG. 11: These images show a simulated rendering of the RTPC with particles traveling through the detector. The small red points are ionizations. The top image also features the translation boards. (Courtesy of Nate Dzbenski)

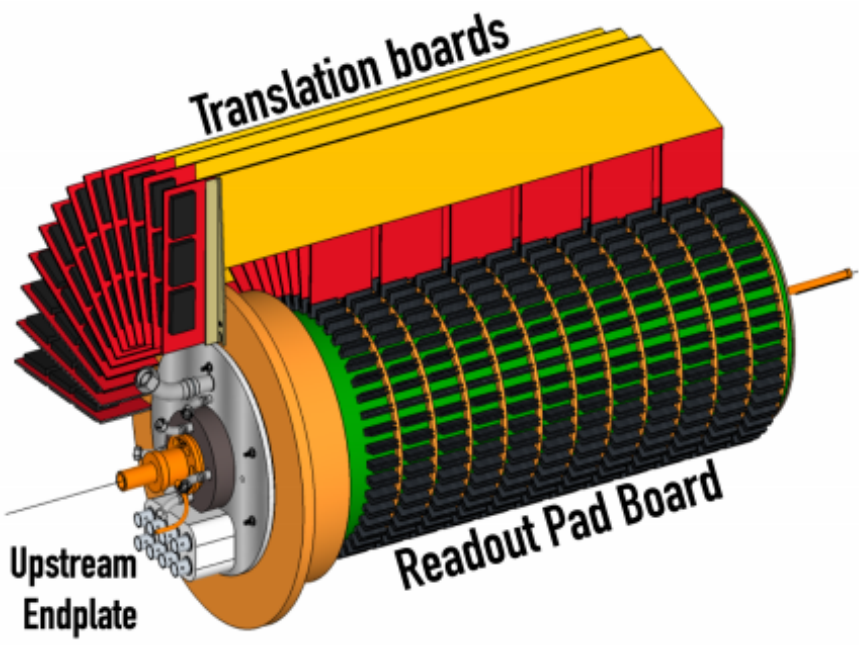

FIG. 12: A 3D engineering rendering of the RTPC showing the translation boards, the upstream endplate, and the readout pad board.

The readout pads are connected to translation boards which communicate the signals on the pads to the data acquisition system (DAQ). Each pad is connected through an isolation capacitor and a overvoltage protection circuit to a mini-coaxial cable that carries the charge from the collected electrons to the Front-End electronics Units (FEUs). The mini-coaxial cables are bundled into 64 for each of 270 MEC-8 connectors which directly interface with the Dead-timeless Readout Electronics ASIC for MicroMegas (DREAM) chips. The DREAM chips (see FIG. 13) convert the signal from each pad into a Gaussian pulse with a width of about 400-800 ns, using a filtering circuit. This signal is integrated over 40 ns bins and every third bin is kept in a buffer. Once a trigger arrives from CLAS12, all buffers across a 7 μs time period are read out and digitized, giving a representative sample of the entire signal. The DREAM chips will handle noise subtraction and zero suppression as well. Each FEU contains 8 DREAM chips consisting of 64 channels each for a total of 512 channels per FEU. The FEUs are grouped into 6 each, for each of 6 FEU crates. Two more FEUs per crate are used to read out the FMT. The FEU is responsible for communicating with the BackEnd Unit (BEU), which then passes the digitized signal information in time-amplitude pairs to the CLAS12 Data Acquisition system (DAQ). There, a decoding software will produce a file which includes a list of readout Pad IDs which have a signal above threshold, and for each readout pad ID, a list of time-amplitude pairs across the time window. These three pieces of information are all the RTPC reconstruction software needs to operate.

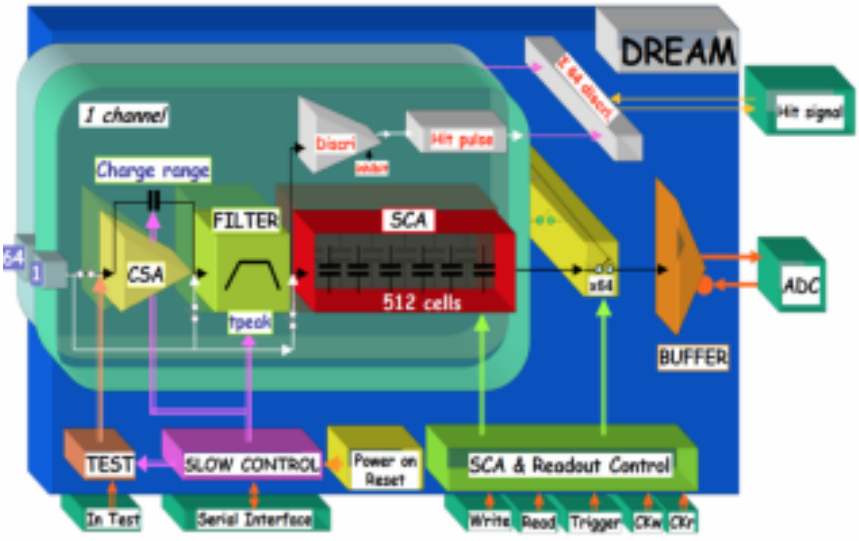

FIG. 13: A technical drawing of the components included in the DREAM chips. [26]

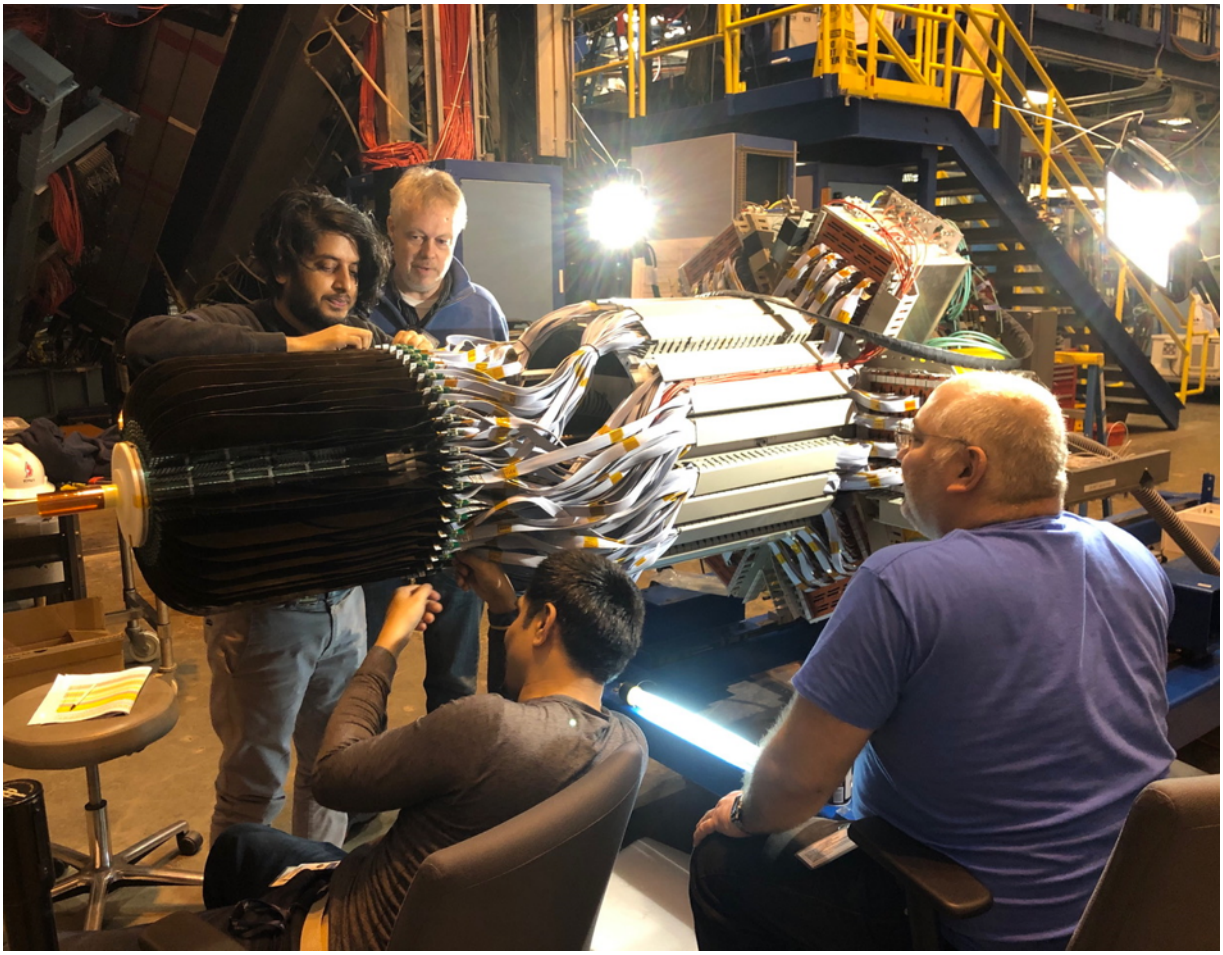

FIG. 14: A photo of the completed RTPC. In this photo the beam would travel from right to left. On the left side of the photo, the so-called buffer volume is visible (the Kapton tube). This is a cylinder from 3 mm to 2 cm filled with ^{4}He to minimize material in the path of scatter protons and the outgoing beam. The green board visible on the surface of the detector is the pad board, and the black signal translation boards are attached all around the outside of the detector. The translation boards are connected to the FEUs which contains the DREAM chips shown on the far right.

3.3 CONSTRUCTION AND OPERATION

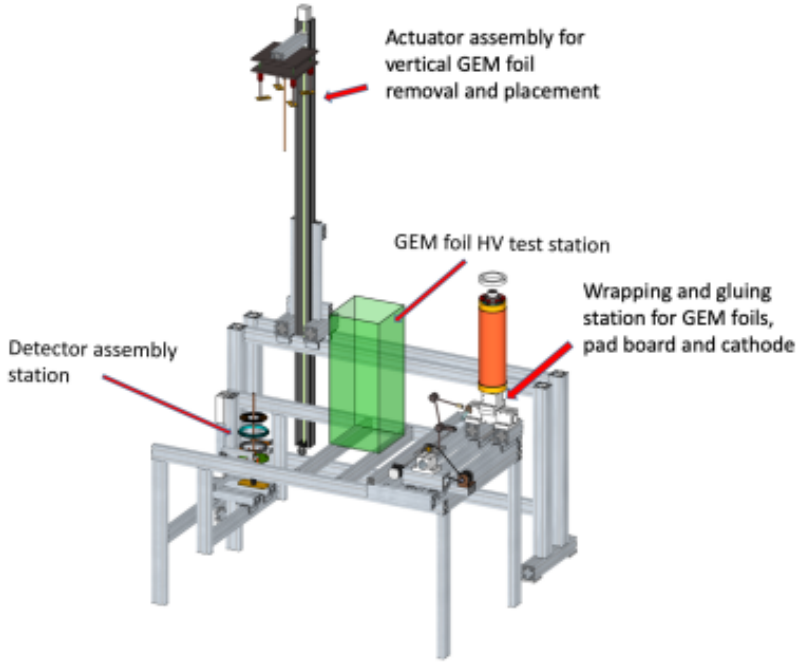

FIG. 15: A digital rendering of the assembly station for the RTPC.

The construction and assembly of the RTPC detectors used in BONuS12 was completed at Hampton University. The assembly station uses mandrels to shape the components of the detector into cylinders. This includes the ground foil, the cathode, the GEM foils, and the readout pad board. Upon assembly of the detectors, they are then delivered to JLab, and first stored and tested in the Experimental Equipment Lab (EEL). Here, cosmic ray tests and high voltage tests are performed, and the electronics are completely assembled and connected. From here, the detector is delivered to experimental Hall B, where the detector is installed into the central region of CLAS12. In addition to the detector, there is a gas distribution and monitoring system which is responsible for filling the target, buffer, and drift regions of the detector with the appropriate gases, and a Drift Monitoring system which is used to calculate the drift speed of electrons in the gas and electric field similar to the drift region of the RTPC [27]. Further ancillary systems include the high voltage power supplies for all GEMs and the cathode.

CHAPTER 4

RTPC RECONSTRUCTION SOFTWARE

4.1 RTPC RECONSTRUCTION SOFTWARE LAYOUT AND UTILITIES

The purpose of the RTPC Reconstruction software is to interpret the time-dependent signals on each of the readout pads of the RTPC and sort them into tracks, and to reconstruct these tracks back into 3-dimensional space inside the drift region, and fit them in order to calculate the kinematic properties of the particles that produced those tracks. In order to do so, the software consists of many individual classes which divide the process into 7 major steps. These steps are reading the input files, track finding, time averaging of the signals, disentangling crossing tracks, reconstructing final tracks into the drift region, fitting them to extract the momenta of the particles, and writing the information to an output file. Each of these individual classes is executed in order by the RTPC engine, which is an extension of the Reconstruction Engine from COATJAVA. The RTPC reconstruction software is a single part of the larger COATJAVA architecture which contains individual reconstruction engines for each of the detectors which make up the CLAS12 spectrometer. The RTPC engine is responsible for reading the information relevant to the RTPC in the constants database CCDB, and then executing each of the major classes in order. Behind each of these classes is a HitParameters object which contains all the relevant "getters" and "setters". The HitParameters object hands the information from one class to the next, so that each step of the engine sees the work of the previous step. Each event is passed through the RTPC engine, and the engine is executed on multiple threads which allows for many events to be read and written at the same time. The RTPC engine expects to find a RTPC::adc bank in the input file or else the entire event is skipped. The engine also reads configuration variables from the yaml file that is used for the reconstruction. These configuration variables define which of the reconstruction modes the user is running. These modes are cosmic, simulation, or experiment. Running in cosmic or simulation modes changes many of the operations performed by the software, and the experiment mode is the default mode. The configuration variables are also used to turn the disentangler on or off, and to determine whether to force the helix fitter to use the beamline as part of the fit of a track. Once the CCDB constants are

initialized and read, the HitParameters object is created, and all the configuration variables are read, the engine executes all the major classes and then continues to the next event.

In addition to the engine and the HitParameters classes, there are many objects and utilities used in the software. These objects are referenced in the different major sections of the software as they are used. Arguably the most important object in the software is the Track object. The track object contains a Java HashMap which uses time as the "key" for the map and returns a List of all the readout pad IDs assigned to that time. The Track object lets you add new time slices, either empty, or already containing an existing list of pads. There is a method for adding a specific pad to a specific time slice as well. There are many methods which return information about the track, or the data stored inside the track. It is possible to return a list of the unique readout pad IDs, or to simply determine whether a track contains a specific readout pad. It is also possible to set specific flags for the track, such as whether this track should be sent to the Track Disentangler. This is discussed more in the sections about the Track Finder, the Time Average, and the Track Disentangler.

Tracks undergo a few transformations throughout the entirety of reconstructing a single event, and as such there are a few variations of the Track object which appear in the software. These are modified to support new data structures that are required to contain the relevant information about the tracks. For instance, a Track in the Time Average class is converted into a ReducedTrack, which has a different time structure than the tracks used in the TrackFinder class. Many of the same methods appear in these alternative objects, with minor changes to support the new ways hits are stored in the tracks.

In order to maintain tracks, no matter which way the data structure changes, custom objects called "TrackMaps" are used to store, manipulate, and return tracks. Each track has its own unique key in a TrackMap which is the Track ID (TID). By passing the TID to a TrackMap, it returns the Track object which is assigned to that TID. TrackMaps contain methods to allow you to get or store a track, to access a list of all the TIDs in the map, and most importantly to merge two tracks together. This is extremely important in the Track Finder, and the Track Disentangler. TrackMaps are extensions of Java HashMaps. HashMaps are structures in Java, similar to arrays which allow storing objects with specific keys attached. So, ordinarily, when storing something in an array it is automatically assigned the next available index. However, for a HashMap, everything is assigned as Key-Value pairs. When an object is stored in the map, it must be assigned a specific key, and thus when retrieving something from the map, the key is used. This structure is perfect for storing and maintaining tracks which have specific IDs. The ID is used as the key, and the value

returned is the Track object.

4.2 INPUT AND OUTPUT

During the development of the RTPC reconstruction software, the file type used for the input and output of the reconstruction changed from EVIO to HIPO. The EVIO file type stands for "Event Input Output" or Event IO. EVIO files consist of a "bank" structure, where the bank structure is defined using an XML file. This file type is the native output file type for CODA, thus files produced while the experiment is running are of this type. During the development of the software, a new file type called HIPO became the default filetype of the COATJAVA reconstruction software. HIPO stands for "High Performance Output". It is designed specifically for Data analysis, and still uses a similar bank structure to that of EVIO. These files are extremely efficient for reading and writing, and perform considerably faster than other files of similar type such as ROOT files. The bank structure of HIPO files is defined using JSON files. The RTPC has a JSON file for instance, which contains all the banks expected to be read for input, and written to for output. In order for the data from the experiment to be read and understood by the software, it is first passed through a decoder. The decoder interprets the raw data from the electronics, and converts it into the integrated and discretized signals assigned to specific pads, time slices and ADC values that the software expects. The output of the Decoder is a HIPO file which is then used as the input for the reconstruction. For simulated events, the process is slightly different. The simulation software (GEMC) used to extensively test the reconstruction software outputs files in the EVIO file type, so these files must first be converted to HIPO, using a converter which is included with COATJAVA.

4.3 TRACK FINDER

The purpose of the Track Finder class in the RTPC reconstruction software is to sort hits into tracks, and flag tracks which may be crossed, in order for the Track Disentangler class to correctly disentangle such crossing tracks. The track finder looks at discretized 120 ns time bins for each of the signals from the RTPC, treating each individual combination of readout pad ID and time bin as a "hit". It uses the pad board around the circumference of the cylindrical detector as a effectively 2-dimensional plane, with the progression of the hits in time to determine track trajectories. Hits which occur close enough in time or space will be sorted together as tracks, and later these tracks will be tested for potential crossing candidates.

The Track Finder class is divided into 3 parts, the sorting algorithm, the track combiner, and the test for crossing tracks. The purpose of the sorting algorithm is to assign a TID (Track ID) to each hit in the event. For the purpose of the track finder, an entire signal is divided into many hits, as this matches with the way the data are organized in the input file. After the completion of track finding, when the tracks are passed to the Time Average class, these signals will be averaged into single hits. For more on this, see the chapter about the Time Average class.

The Track Finding algorithm consists of a series of nested loops. The first loop in the algorithm is over time. The time loop checks every time bin, starting from 0, all the way to the maximum time of the window, in 120 ns steps. This matches with the discretization of the signals from the DREAM chips. For each individual time bin in the loop, all the readout pads which have a signal above a user-defined threshold in the current event are checked at this time. In effect, you are looking at the pad board for an individual time "slice" and seeing what the ADC values are for every pad. Each different pad for the current time slice is treated as a "hit".

For each individual hit, first the ADC value (or the amplitude of the signal at this time) is compared against a threshold. If the hit passes the threshold, it is considered a valid hit which will be sorted into a track. For each valid hit, a PadVector is formed for the hit. A PadVector is a custom object which contains the RTPC reference frame positions of the pad. The PadVector object returns quantities like the row and column of the readout pad, the (x,y,z) coordinates, and the (r,ϕ) coordinates. All the position coordinates are calculated using the pad ID. For more information about this, see the section about the RTPC geometry.

Now that the PadVector for the hit is formed, there is a loop over all the existing track IDs in the current event. For the first valid hit, there will be no existing TIDs, and in this circumstance, a new Track object is created and added to the TrackMap object. The Track object contains methods which allow you to add new hits to the track, but also many useful utility methods to learn information about the tracks. These include returning a list of all the hits in the track, or all the time slices in the track, or how many unique readout pads exist in the track. The TrackMap object is responsible for organizing Tracks into a HashMap in which it is possible to add or retrieve tracks based on their unique Track ID. It also contains a method which allows you to merge multiple tracks into one, a feature which will be very useful later. From now on, Track objects and the TrackMap will just be referred to as "tracks" and the "map".

For the second valid hit and onwards, there is a loop over the existing tracks. In this

case, the current hit will be compared to the hits which are already assigned track IDs. If the current hit is close enough in space or time to a hit which already has a track ID, then the current hit is also assigned this same track ID. First, there is a loop over time slices, starting from the time of the current hit, and moving backwards in time in 120 ns steps. In each time slice, all the hits from the current track in the loop are listed, and the current hit is compared against each one. In order to compare the position of two hits, a TrackUtils class is used. The TrackUtils class contains the method comparePads which takes two hits, and returns a Boolean based on whether these two hits should be sorted or not. The current method used to compare hits looks at the distance between the two readout pads these hits originate from in (ϕ,z) coordinates, and ensures they are close enough together, by comparing the distance in these coordinates to a user-defined threshold. Recall that all thresholds used by the Track Finder are defined in CCDB. If the current hit finds a hit in the current track which is close enough, then the current hit will be assigned the track ID of the comparison hit. In this way the hit now belongs to the track. However, this is not enough for the Track Finding algorithm. For instance, suppose that there are 3 hits which occur in the same time slice in a straight line, but because of the order these hits are read into the software, the third hit in the list is between two different hits. If the first and second hits are compared to one another, they may actually be too far apart to be sorted into the same track. This would result in these hits having different track IDs. Then the third hit is compared to each of the first two hits, and is close enough to each. If the algorithm simply stops when a hit finds a valid track, then this would result in a broken track. The solution to this issue is to continue comparing the current hit to all the other tracks in the event. If the current hit finds other tracks which would result in the hit being assigned their track ID, then this is allowed, and the current hit may have more than one assigned track ID. Such hits which contain multiple track IDs are then used to merge broken tracks. However, this also leads to a potential issue. Say the hit is close enough to multiple tracks, but actually these tracks should in fact be treated as separate tracks, then this merger will result in crossing tracks merging together. This is the reason why the Track Disentangler exists.

Once a hit has been compared against all the existing tracks in the event, the merger is performed. If the hit has more than a single track ID assigned to it, then all the existing tracks with any of these IDs are merged into a single track in the map. This means that all the existing hits in these tracks are now treated as belonging to a single track. If the hit has only a single track ID assigned to it, then nothing will happen, and if the hit has no track ID assigned to it, then this means the hit was not sorted into any tracks. At this point, a

new track is formed with a new unique track ID, and the current hit is added to the track. Then the new track is added to the map, so that from the next hit onwards, this track will also be used for sorting new hits.

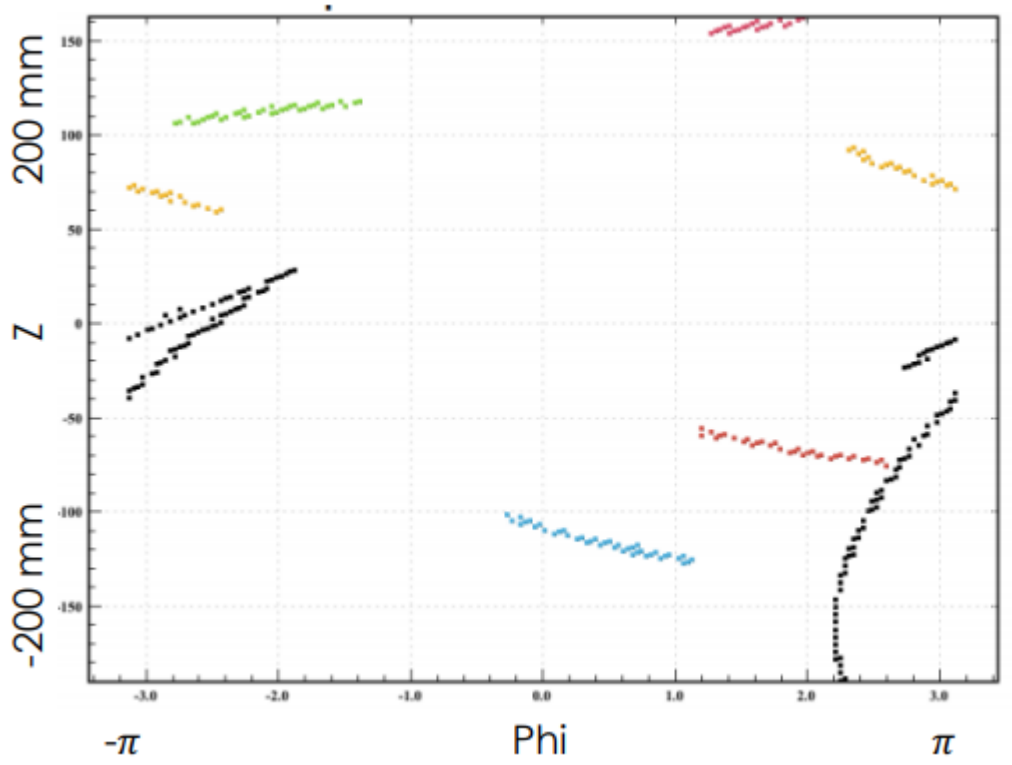

FIG. 16: An example of the output of the Track Finder. The colors are automatically assigned to each track, so that it is easy to see which tracks are identified as separate tracks by the Track Finder. This example perfectly demonstrates how the merging algorithm can cause two tracks to be merged together into one as shown by the black track which wraps around the detector in ϕ.

After every time slice, with every valid hit sorted, the track finding algorithm and merging is complete. There is only the disentangler flag left. First, a simple check of the size of tracks is performed. For every track in the map, if there are not enough hits in the track then the track is removed. The reason for this is that tracks with too few hits would not be reconstructed or fit properly, and would not be useful for analysis. This saves time for the software, as tracks are thrown away early. To be more precise, the size of the tracks is determined not by how many hits are in the track, but by how many unique readout pads are

in the track (recall that a hit is only a single time bin or slice of the full signal on a readout pad). For this reason, it is possible that the same readout pad appears across multiple hits in the track. In this way it is possible for a track that only contains 3 readout pads to have as many as 15 or more hits. For this reason, simply checking the number of hits in a track is not enough to determine whether it is long enough to be useful in the analysis. In the Time Average section these hits will be averaged together for each signal, so that the end result is every signal relates to only one hit.

The last step is to flag potential crossing tracks. This is done by determining the amount of time spanned by a "track" in the map. If a track spans a long enough time range, longer than the typical maximum drift time of a track, then it is probable that the track is actually a combination of multiple tracks which arrived in the detector at different times. A similar test is performed for each readout pad in a track. Since the readout pad has a signal, and this signal is divided into individual hits, it is also possible that the readout pad has multiple signals across the time window. If hits associated with the same pad span a long enough time range, then this is also cause for concern that the track is actually a combination of multiple tracks, since this usually means that the hits belong to different signals on the same readout pad, and different signals should belong to different tracks. Lastly, we perform a check to see if we have hits in the track which are not actually meant to be sorted together. We do this by looping over the hits of the track, in order, and comparing the next neighbors. If we find that there are actually hits which are not close enough to be sorted (which should only occur in the case of backbending tracks, or crossing tracks), or either the time span of a single pad, or the time span of the entire track are too long, then this track is flagged, and will be treated again later by the disentangler.

4.4 TIME AVERAGE

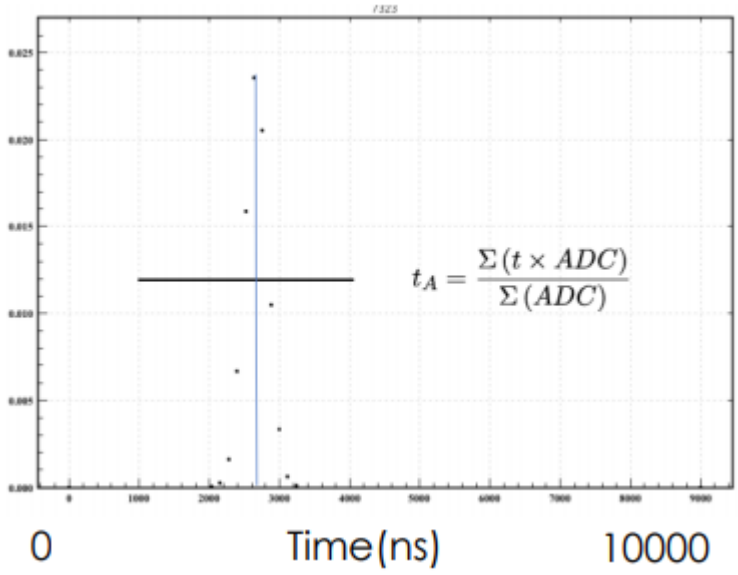

FIG. 17: A single signal, showing how the weighted average is calculated. The horizontal axis is time in ns and the vertical axis is ADC in arbitrary units.

The purpose of the Time Average class in the RTPC reconstruction software is to switch from the time-discretized nature of the signals used for sorting in the Track Finder, to individual hits per signal, which have an average time, and contain the total energy deposition of the signal, used for determining the track's total energy deposited, and the dE/dx of the track. The Time Average class looks at all the tracks from the Track Finder and creates a new map of the same tracks, but with averaged times. This means that after the Time Average method is complete, the number of hits and the number of unique readout pads in a track are the same. More information on this distinction can be found at the end of the Track Finder section.

The Time Average class takes in the TrackMap object from the Track Finder, and also creates a new object called a ReducedTrackMap. All the Track objects from the TrackMap will be converted into ReducedTrack objects for the ReducedTrackMap. The reason for this distinction is that the data structure of a ReducedTrack is different now that it can no longer be divided into time slices, and instead consists of hits with non-discretized times. Otherwise the ReducedTrack and ReducedTrackMap objects work very similarly to their matching non-reduced counterparts, including containing modified versions of the same methods, to support the new structure.

First, a loop over all the tracks in the TrackMap is performed. For each track, a ReducedTrack is formed. If a track was flagged by the Track Finder for the Disentangler, then this new ReducedTrack will carry on the flag. For every pad in the track, all the hits belonging to that pad will be used to determine the max ADC value of the signal, and the weighted average time of the signal. The weighted average time of the signal is calculated as follows

$$\frac{\sum_t ADC_t t}{\sum_t ADC_t}. \tag{9}$$

This average is only calculated for hits which have an ADC value greater than a quarter of the maximum ADC value of the signal. The denominator is also used to maintain the total ADC of the signal. The total ADC of the signal is also divided by the calibrated gain of the readout pad. The readout pad the hits originate from, as well as the weighted average time, and the total ADC of the signal are collected together in a HitVector object. HitVectors are extensions of PadVectors. They contain all the information a PadVector contains, but also the information about the ADC and average time of the hit. ReducedTrack objects store the hits as HitVectors so that all the important information about each hit is stored together. Once this is done for all the pads in the Track, and all the new HitVectors are added to the new ReducedTrack, the ReducedTrack is added to the ReducedTrackMap.

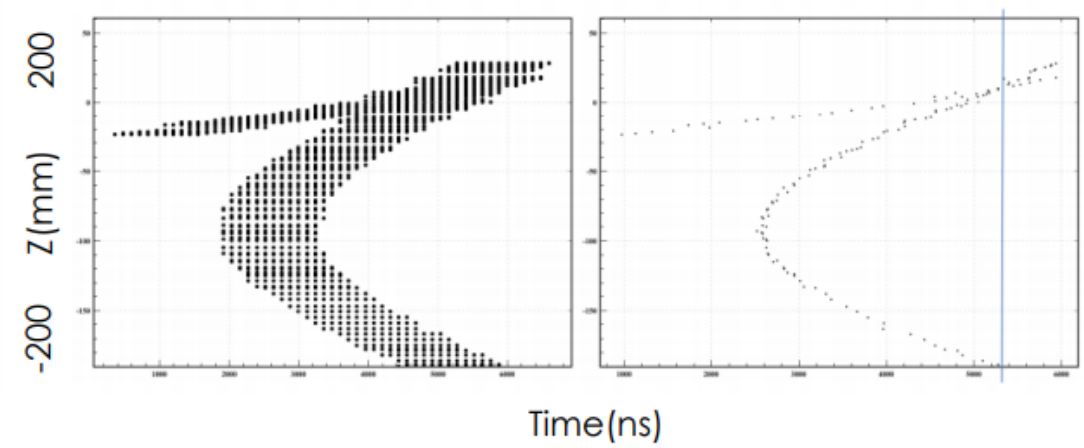

FIG. 18: Here we see the crossing track example from the previous section. Without applying the time average to each signal, the point where the two tracks cross is not well-defined. However, after reducing the signals to their average times, the point of intersection is much clearer.

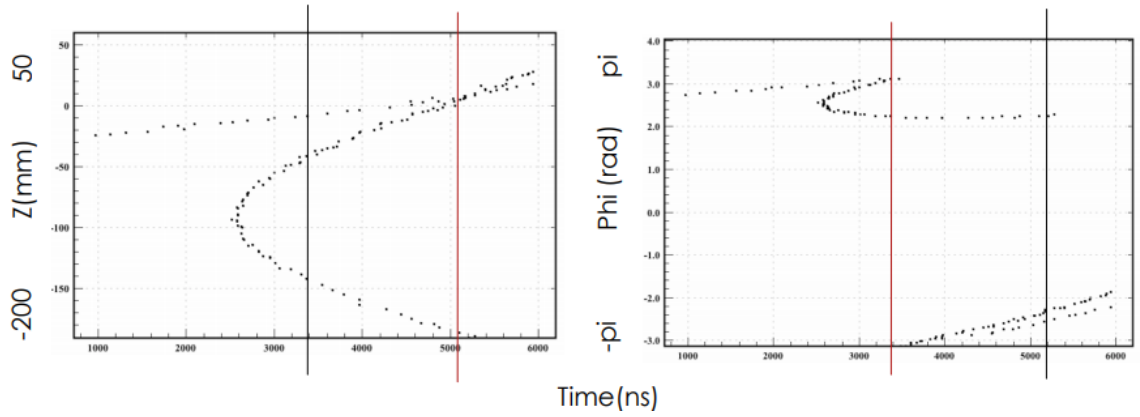

FIG. 19: Here is the same two crossing tracks, after taking the average time. Comparing Z vs t and ϕ vs t, it is very clear to see that these two tracks cross at different times in the two space variables, meaning it will be easy to separate these tracks. The two vertical lines are added to show the intersection points from both plots.

4.5 TRACK DISENTANGLER

The purpose of the Track Disentangler class in the RTPC Coatjava reconstruction software framework is to apply a second sorting algorithm to hits in tracks which were previously identified as potential crossing track candidates, and hopefully sort these hits into individual tracks rather than combined tracks. The reason a class like the Disentangler exists is because without it, crossing tracks would be treated as single particle tracks in the detector, and the resulting 3-dimensional track reconstruction, the helix fit of the track, and the momentum would all be calculated incorrectly. For more details on how tracks are flagged and passed to the Disentangler, refer to the Track Finder section.

The input of the Track Disentangler is an object called RTIDMap which stands for Reduced Track ID Map. This Hashmap is the structure used for the output of the Time Average class, and it contains all the tracks in the event, which are stored as ReducedTrack objects. For each track contained in the map, first a check is performed to determine whether this track has been flagged for the Disentangler. This flag is the very same flag set in the Track Finder where the track is first potentially identified as containing more than one particle. Assuming this flag is set to true, then the sorting algorithm begins.

The sorting algorithm in the Disentangler only acts on a single track at a time. The goal is to re-sort all the hits in the current track into a new track or tracks, depending on if the Disentangler determines that there is indeed more than a single track here. First, all the

hits in the track are time ordered, from largest to smallest time, and then compared to one another in z, ϕ, and time. The parameters used to define the maximum allowed difference in these quantities is stored in the Constants Database (CCDB). At the beginning of the sorting algorithm, these hits will no longer have a track ID. For the first hit, it is assigned a completely new Track ID (TID). After the first hit, when a new hit needs to be sorted it is first compared against the existing track IDs created by the Disentangler, to determine if the hit needs to be sorted into one of the existing tracks. Each hit is compared with the last hit (the hit with the smallest time) in each of the new tracks, one at a time. If the distance in ϕ and z between the current hit and the last hit of a track is within the limits set by CCDB, and the current hit is not too far away in time, then the current hit will be assigned to that respective track.

This algorithm is performed once on every single hit in the original Track. Once the algorithm is complete the result will be either a new track which is identical to the track which was used as the input, or multiple new tracks which were found using the more strict sorting algorithm. This is however not enough to completely rebuild all the existing particle tracks in the event. For instance, if a track bends back towards the cathode, then it will form a parabola, such that the hits with the largest time in the track can correspond to completely different ϕ values or z values. Since the sorting algorithm is performed in a time-ordered fashion, namely from largest time to smallest time, the result for a back-bending track will be two separate tracks which approach very near one another at small time, at the peak of the parabola. This is of course a problem considering the goal of the Disentangler is not to break existing tracks. For this reason a second more specific sort is performed.

All the new tracks are compared once again, only this time using a pool of hits which consists of the first two and last two hits of each track. If any of these hits is close enough in z, ϕ and time to be sorted, then these tracks will be combined into a single track. This will allow back-bending tracks to be combined at the smallest time, but it will also allow straight tracks which may have been broken by a time separation that is too large, to be recombined at the points of closest approach.

All the resulting tracks from this process are then passed on to the original RTIDMap, as new tracks, and any tracks which were re-sorted are removed from the map. This is to prevent crossing tracks from continuing to be contained in the map. This map is then passed back to HitParameters and overwrites the existing map, to then be used by the TrackHitReco class. Because the input and output data structure is the same, this entire class can be skipped. The file which defines the parameters and detectors for reconstruction contains a flag which

when set to false will turn the Disentangler off completely. The reason for this feature is because it is sometimes possible for the Disentangler to cause unexpected results, depending on the track geometry, and hit density. For instance, a single straight track with gaps small enough to be sorted by the Track Finder, but too large to be sorted by the Disentangler, may result in the track being broken into pieces. This track of course shouldn't be flagged for the Disentangler in the first place, but it's always possible for unexpected situations such as this to arrive. The Disentangler has a possibility of breaking or recombining tracks incorrectly, and was certainly the most challenging part of the software to test and debug. It is for this reason that it is possible to turn the Disentangler off, especially if strange tracking results are seen in the output of the reconstruction.

4.6 TRACK RECONSTRUCTION

The purpose of the Track Reconstruction class in the RTPC reconstruction software is to reconstruct all the hits of a track into the drift region of the detector, close to where the original ionization occurred. In order to reconstruct the true position of the hit in 3D space, the position of the pad, and the time of the hit are used. The reconstruction formulas which calculate the (r,ϕ,z) position of the hits only need these two quantities. The end result of the Track Reconstruction software is a map of tracks which will contain all the previous information about the tracks, as well as the new reconstructed positions of the hits in each track. This map will then be sent to the Helix Fitter to fit the tracks and extract the momentum.

First, a loop over all the tracks is performed, and again each is checked to see if it is too short to be usefully reconstructed. For tracks which are a reasonable length, a time shift is calculated for the track. This shift is based on the expected maximum drift time of a track, and the current largest time value of the track. This time difference is applied to all the hits of the track. This has to be done in order for the track to be reconstructed correctly. The equations which relate the hit's time and readout pad ID to a spatial point are expecting the time of a hit to be based on an "in-time" track relative to the trigger. However, since our time window is so large, we have many more tracks than just the in-time track, and for these tracks to be properly reconstructed and studied, the time shift has to be applied. This means that for an in-time track, the time shift should be extremely small.

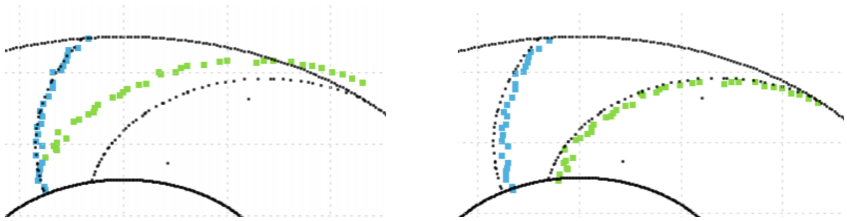

FIG. 20: The above figures demonstrate how using this technique to shift the time of all the hits allows for a more accurate reconstruction. The black points are simulated tracks, and the colored hits are the reconstructed hit positions. Note how in the left image, the blue track is already well reconstructed, implying this track was "in-time", however the right track was out of time, and the shift corrected for this, while still applying a small shift to the in-time track.

Once the time shift has been applied to all the hits, the radial distance from the target center of each hit is calculated. The formula for calculating this distance is as follows

$$r\left(t,z\right) = \sqrt{r_{max}^2\left(1-x\right) + \ r_{min}^2 x}, \tag{10}$$

$$x = \frac{t - a_t}{b_t}. \tag{11}$$

The ϕ shift due to the curved ionization electron paths in the solenoid field of each hit is then calculated using the reconstructed radius

$$\Delta\phi\left(r,t,z\right) = a_\phi + b_\phi \ln\frac{r_{max}}{r}, \tag{12}$$

$$\phi_{true} = \phi_{meas} + \Delta\phi. \tag{13}$$

r_{max} and r_{min} are 70 and 30 mm respectively and a_t, b_t, c_t, a_ϕ, and b_ϕ are z-dependent coefficients whose parameters are defined in CCDB, and were determined by simulating the magnetic field of the detector in Garfield++. Garfield++ is a simulation software designed specifically for the purpose of simulating ionizations in gases inside of particle detectors. For the purpose of the RTPC, Garfield++ was used to simulate the drift in ϕ and z of ionized electrons as they travel towards the GEM foils. The magnetic field map was simulated in Garfield++ to get an accurate drift path of these ionized electrons, and determine the relationship between the radial point of ionization and the time of the signal on the readout pads.

The z of each hit remains unchanged and is based on the z of the readout pad. The r, ϕ, z, and time shift, as well as the original times of the hits are passed as RecoHitVector

objects to the final map. RecoHitVector objects are further extensions of HitVectors and contain much of the final information of the track.

4.7 HELIX FITTER

The purpose of the Helix Fitter in the RTPC reconstruction software is to fit tracks which have been reconstructed into the drift region with a helix. Since the particles in the detector are travelling through a magnetic field of up to 5T, charged particles will curve along a helical path, and the momentum of charged particles in this field can be calculated using the relationship between the centripetal force of the particle and the magnetic force

$$p_{\perp} = qBr \tag{14}$$

. Here, r is the radius of the circular path the particle follows, or in our case the radius of the helix. $p_{\perp}$ is the momentum perpendicular to the magnetic field, q is the charge, and B is the magnetic field. Because of the fact that the helix has a third z-dimension due to the forward or backward motion of the particles, there is an additional factor of $\sin(\theta)$ to account for the z-component of the helix. The final form used to calculate the momentum is

$$p = \frac{qBr}{\sin(\theta)}. \tag{15}$$

The helix fitter is a modified form of the fitter used for BONuS6, which has a few associated authors and credits, all listed in the references. It is based on the subroutine CIRCLE from the book "Computer Physics Communications". The result is a helix fit to the hits in 3D space, and the fit returns the radius of the helix, as well as the center of the helix in the X,Y plane, the angles ϕ and θ of the first step of the fit, the position of the vertex which could be forced to the beamline axis, the DCA or distance of closest approach to the beamline axis, and the χ^2 of the fit which is not used. The actual code for the helix fitter is in the HelixFitJava class. The parameters of the fit are passed to a HelixFitObject, and additional calculations are performed in the HelixFitTest class, where we calculate a custom χ^2 of the helix fit by finding the χ^2 of each individual hit relative to the fit, and include extra terms based on the radius and center of the helix. The formula used for χ^2 is the following

$$\chi^2 = \frac{\sum_i^n \left[\left(\frac{\phi_i - \phi_0 + \arcsin\left(\frac{r_i}{2R}\right)}{0.01} \right)^2 + \left(\frac{z_i - z_0 - 2R\arcsin\left(\frac{r_i}{2R}\right)\arctan\theta_0}{1.2} \right)^2 \right] + \left(R - \sqrt{A^2+B^2}\right)^2}{2n-4}, \tag{16}$$

where the summation is over all the hits in the track, and $\phi_0, \theta_0, R, A, B$, and z_0 are all the parameters of the helix fit, with (A, B) being the center of the helix. The first term in the

sum calculates the error in ϕ relative to the helix, the second is the same for z, and the last term is a correction term for the radius relative to the center of the helix. Due to the fact that several tracks have some excess hits which appear as noise on the nearby pads in the detector, a chi2 limiter is applied to all the tracks. If any hit exceeds this limiter then such hits are removed, and the entire track is fit again. This is to prevent noise from clouding the fit of an otherwise good track. If too many hits are thus removed from a track, the whole track is discarded. Once the fit is performed, and the chi2 of the track is calculated, the track length and dEdx are calculated for the track. The track length is calculated using helical geometry by determining the span of the helix which is occupied by the track and calculating the arc length and delta z of this portion of the helix and adding them in quadrature to calculate the total track length as follows

$$\text{Track Length} = \sqrt{R^2\theta^2 + \Delta z^2}, \tag{17}$$

where θ is the angle between vectors pointing to the start and endpoints of the track, R is the radius of the helix and Δz is the span of the track in z. Then the ADC or energy of all the hits in the track is summed to determine the total energy deposited by the proton and this is divided by track length to determine dEdx. The final parameters of the track are then passed to the output. The final track parameters are momentum, vertex, θ, ϕ, the number of hits, the total energy, dEdx, the Radius and center of the helix, and the chi2 of the fit.

4.8 NONSTANDARD OPERATION

In addition to the usual operating method of the reconstruction software, analyzing real experimental data, there are also a few additional "modes" in which to operate the software. For instance, when reconstructing simulated data or studying cosmic ray data. The following sections explain elements of the software which are considered to be "non-standard" and have to be activated through the use of "flags" which must be set in the configuration file of the software.

4.8.1 SIGNAL SIMULATION

Before the BONuS12 experiment began running in spring of 2020, the main way of testing the reconstruction software was through simulation. The software used for simulation is called GEMC or Geant4 Monte Carlo, which is designed as an extension to Geant4, the simulation toolkit originated at CERN. The purpose of GEMC is to simulate particles

travelling through matter, and more specifically, how the detector subsystems will respond to the particles. The version of GEMC used for our experiment contains all the detector subsystems that make up the CLAS12 spectrometer, and also the RTPC which is added to CLAS12. As a result, all the electronics outputs are simulated for each detector, and the output is meant to look as similar to real data as possible so that testing on the reconstruction software can be performed.

However, in the case of the RTPC, the structure of the output of GEMC limited our ability to realistically simulate the structure of real data. This is because the output we expect is accumulated signals on each readout pad, but GEMC is only providing each individual electron's arrival time, and deposited energy. Since the reconstruction software expects signals which are accumulations of many electrons which have avalanched in the GEM foils, the Signal Simulation class in the RTPC reconstruction software is used to recreate these signals.

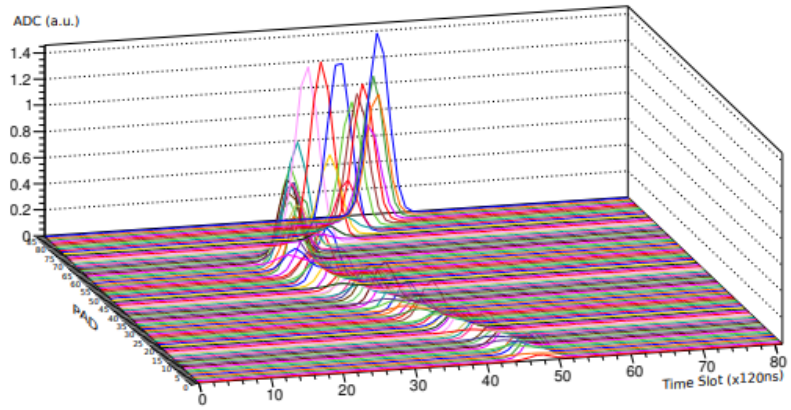

FIG. 21: A 3 dimensional plot of the signals simulated on each pad, forming a track.

In order to simulate the signals, each individual electron in the output of GEMC is assigned a signal, based on the realistic signal shape we expect. Then, for all the electrons which arrive on the same readout pad, all these signals are summed together and integrated into 120 ns time bins. This is to match the output structure of the DREAM electronics, and the input structure of the Track Finder. For each readout pad, the signal is binned over the entire time window which is defined by how long data are read after the trigger. In most cases this was on the order of 10 μs. In each time bin, the integrated signal for this bin is defined as the ADC value for that bin, and is used to determine the total energy of the track

later in the software.

The signal simulation is used as a pass-through to organize the real data into the structure expected by the Track Finder, since there is no need to simulate signals when working with non-simulated data. However, simulating signals is still possible for future simulations, and simply requires enabling the simulation flag in the yaml file for the reconstruction.

4.9 COSMICS

In an effort to test and calibrate the RTPC, the detector could be used to detect cosmic ray muons. Cosmic studies were also used to test the monitoring software, which will be explained in further detail in the following chapter. In this case, no beam or CLAS12 detectors are used, and the detector simply needs to be turned on and sensitive to these cosmic rays. The trigger for data acquisition can be a set of scintillator paddles in coincidence to detect a cosmic ray which hits both paddles, or the CTOF in CLAS12. In order to operate the software to reconstruct cosmic rays, there is a rtpcCosmic flag which, when set to true, converts the reconstruction software into cosmic mode. In cosmic mode, the reconstruction expects a single track in the event, and thus the track finder is not required to separate hits into multiple tracks, thus all hits are just assigned to one track. Furthermore, no time shift is applied since all cosmic rays are expected to be in time with the trigger, and their tracks don't necessarily go through the cathode. An example of a cosmic ray track in the RTPC can be seen in FIG. 22.

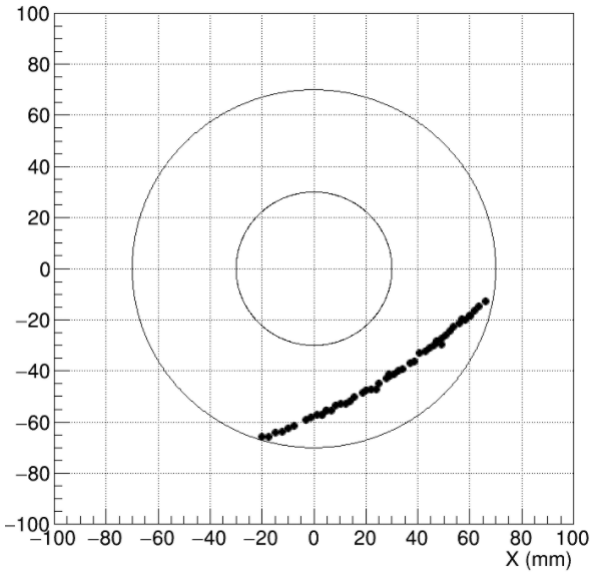

FIG. 22: A plot of Y vs X showing an example of a cosmic ray track in the RTPC.

CHAPTER 5

DATA ANALYSIS, RESULTS, AND FUTURE WORK

In the following sections, I discuss the history of the software development and how generated and real data were used to validate the software. In addition, the monitoring software used during data collection will be covered in detail, and some of the preliminary results from both simulated and real data will be shown. The final section covers the future of data analysis for BONuS12, as well as planned improvements and additions to the software. The reconstruction software explained in chapter 4 is being used daily to reconstruct BONuS12 data, as the data analysis group prepares for the first pass analysis of the entire data set. A first publication of the BONuS12 results is expected in 1-2 years.

5.1 SOFTWARE DEVELOPMENT BEFORE THE EXPERIMENT

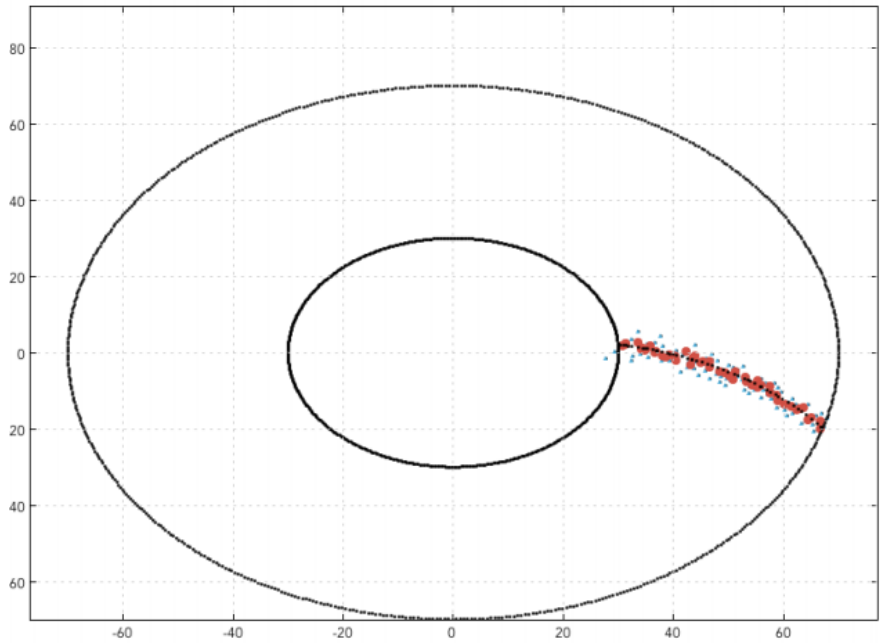

FIG. 23: X-Y view of an early version of the RTPC reconstruction software. The graph shows 3 sets of data. Black dots are the true x-y locations of the generated hits in the simulation. Blue dots are a test of the reconstruction formulas without any track finding or signal averaging, just using the time of the hits and reconstructing the position. Red dots are the actual results of the reconstruction software including sorting these hits into a track, averaging their signals, and reconstructing the final time into the drift region.

Long before the experiment began in February 2020, much work was being done in preparation for not only the experiment, but also for being ready to analyze data both during and after the experiment. Once a preliminary version of the reconstruction software was completed, which could reconstruct tracks but not fit them, a lot of studies with simulated data ensured that the reconstruction worked as expected. First, we worked with tracks which were all in-time, and overlaid reconstructed tracks on top of the reconstructed hits. An early example is shown in FIG. 23. Then, we worked on dealing with out-of-time tracks and figuring out how we could time-shift these tracks inside the reconstruction, without knowing what the time-shifts actually were. In the RTPC simulation software GEMC, in order to test this, the simulation would throw one electron and the scattered proton from a random vertex position, and this would be considered the "in-time" proton, which would have a time shift of zero. In addition to the in-time proton and electron, there would be many more protons that pass through the drift region with different vertex positions, and at different times, spread out across $\pm 8\mu$s. Due to the fact that the reconstruction uses the time of each signal in order to reconstruct the position of each ionization in the drift region, these tracks would be reconstructed incorrectly, unless a time-shift is applied to each track to correct for this. Applying this time-shift forces the reconstructed track to start with the first ionization at the cathode. There is some error introduced in this way due to the fact that it is not always true that the first ionization occurs at the cathode. This will cause a slight shift of the in-time track in each event. You can see an early version of that in FIG. 24. Then we moved on to the addition of a helix fitter, early results of which are shown in FIG. 25. Analysis scripts were developed upon the completion of the earliest versions of the RTPC reconstruction software. These were tested using simulated data, and modified to support the analysis of real data as soon as they became available.

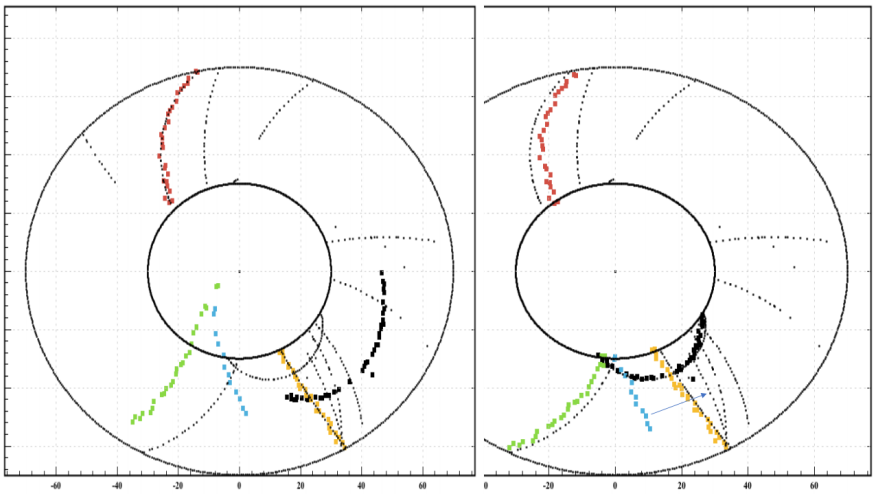

FIG. 24: X-Y view of an early version of the RTPC reconstruction software. These pictures were used to demonstrate the effects of applying a time-shift correction to reconstructed tracks. The left image shows an event with no time-shift correction, and the right image is the same event with time-shift corrections applied. Black dots are true x-y points from the simulation, and the colored dots are the tracks found by the track finder. Notice how without a time-shift correction there are a couple "in-time" tracks, and that these tracks are shifted slightly off by the time-shift correction. The others are out-of-time, and can be corrected, unless the hits start too far away from the cathode, which is the case for the blue track for example. Notice how in addition to out-of-time hits having poorly reconstructed r values, they also get shifted in ϕ. This is due to the fact that both the r and ϕ reconstruction depend on the time of the signal, so by applying a time shift we can correct r and ϕ. The black track here demonstrates this well. The curvature of the track tightens up, and shifts in ϕ when applying a time shift to the hits in the track.

The analysis of BONuS12 has been underway since the beginning of the experiment in February 2020. The RTPC reconstruction software performance exceeded the requirements in terms of computation time after many optimizations and thread-safety related updates. Several pieces of the software, especially the Track Finder, were reduced in size by nearly an order of magnitude. The original version of the software was far too slow, taking upwards of a whole second per event, but more importantly the software was originally not "thread-safe". Thread safety in software means that the software can be executed on multiple different cores of the CPU simultaneously. In order for this to work, it is important to be careful about the types of objects used, and how they are initialized into memory, and how they are accessed. For example, the Track Finder is adding and removing tracks and hits to and from HashMaps. If the Track Finder is running in a multi-threaded environment, and information

is removed from the map, while another thread is reading from the map, this would cause a thread-related crash in the software. In our case, the HitParameters object was the biggest problem. As a reminder, the HitParameters class is where all the variables used for all of the software are stored. Also, when proceeding from one class to the next, information is stored and accessed from HitParameters. For this reason, when running the software with multiple threads, it's very important that the HitParameters object is initialized once per thread, so that each thread is not manipulating the same instance of HitParameters. The advantage to multi-threading is that the software can process multiple events simultaneously, resulting in much faster computation times. Once the experiment began, the reconstruction software was used in nearly real-time to provide diagnostics and optimize the RTPC performance. Due to unforeseen properties of the real data, updates were also made during data-taking as new problems arose. The version of the software presented in this thesis is based solely on the most updated version at the time of writing, but there are many more planned updates to the software.

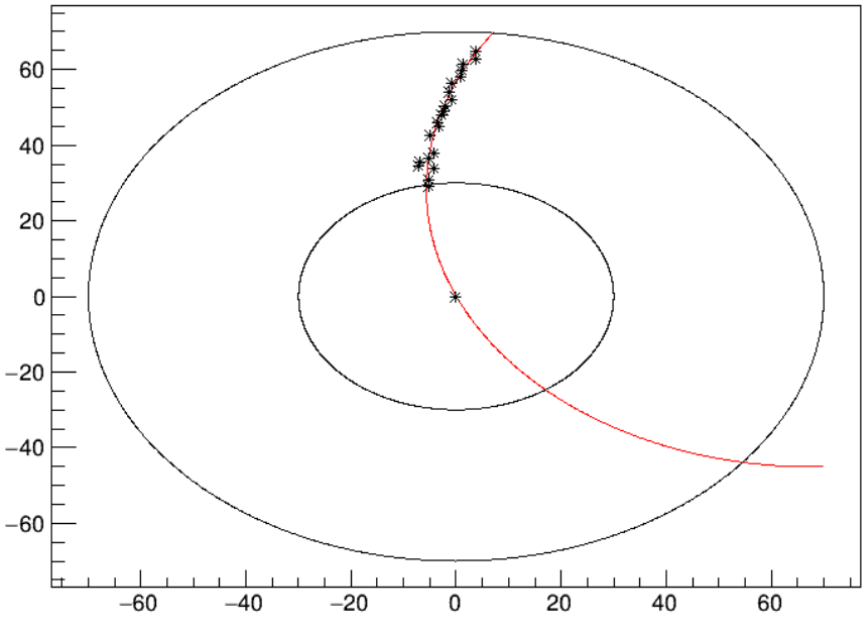

FIG. 25: X-Y view of an early version of the C++ script used to test the helix fitter. The black stars are the resulting hits of the RTPC reconstruction software. The red line is drawn using the parameters of the helix fitter, which includes the center of the helix, and the radius of the helix. There is a point in the center which is added by the helix fitter when the beamline fit option is turned on.

5.2 EXPERIMENT STATISTICS

BONuS12 ran from February 2020 until September 2020. During that time, data were taken using three different targets. A hydrogen target was used to study events where the electron strikes a free proton. This is useful both for calibration and for background studies. Deuterium was used as our main target to study neutron structure as explained in chapter 2. Helium was used to study background from target contamination. Empty target runs were also used to study background. The experiment ran with two different beam energies, dubbed as 1-pass and 5-pass. The pass number is based on the number of passes the electrons take around the accelerator racetrack before entering Hall B. 1-pass runs reach an energy of 2.1 GeV, with 5-pass runs reaching an energy of 10.4 GeV. 1-pass runs are used for normalization and calibration of the detectors, and the 5-pass runs are used for collecting physics data. Approximately 434 million events were recorded at 1-pass, and 5.241 billion events were recorded at 5-pass. Two different RTPCs were used during the experiment. RTPC1 was used in February and replaced in March by RTPC3, due to performance issues. In addition, the experiment was interrupted from March 24 through the end of July by the so-called MEDCON6 condition at Jefferson Lab due to the COVID-19 pandemic.

5.3 EXPERIMENT MONITORING

While the experiment is underway, and data are being collected, there are many ways to monitor the status of the detectors. In the case of the RTPC, I contributed to and developed two of the monitoring systems used to ensure the RTPC is working properly while data are being collected. First, every single operating detector has a set of histograms which make up the software called “CLAS12MON”, or in other words, CLAS12 monitoring. This software collects information about each detector in real time, and can be used to locate potential issues in each detector, or to prompt changes which may need to be made to each detector, such as, in our case, changing the high voltage. When installing the RTPC into CLAS12, we had to create our own set of histograms in CLAS12MON, and an example of the RTPC monitoring is shown in FIG. 26.

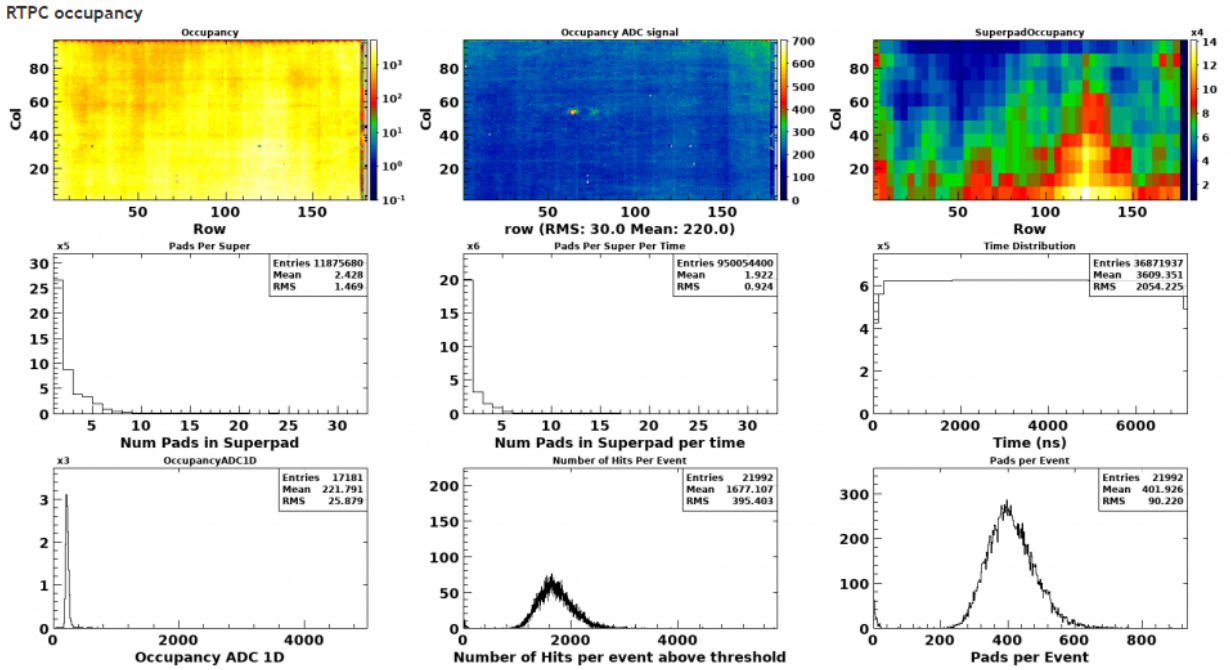

FIG. 26: Example of the RTPC monitoring histograms in the CLAS12 monitoring software. For explanations of each of the histograms see the text.

The RTPC monitoring histograms show several quantities which we are interested in monitoring during the experiment. The first row of histograms is related to the occupancy of the readout pad board. This is monitored in a few different ways. The first histogram monitors simply how many times a readout pad shows up over a specified time period (e.g., a run). This can happen several times for each signal, due to the fact that the signal is binned into 120 ns time bins. Each bin above the ADC threshold will contribute to this histogram. The second histogram shows the average ADC value for each pad averaged over all the hits on that pad. This histogram also tracks the RMS and mean of the ADC values of each signal, and is used to calibrate the gain of each pad. The third histogram shows "superpad occupancy". This divides the readout pad board into larger superpads which consist of 32 pads in 8x4 rectangles. This allows us to study interesting things such as what fraction of a superpad is active in an event (histogram four) and in a single time bin (histogram five). This is important because the DREAM hardware assumes that the lowest 16 signals from each superpad is background noise which is subtracted from each signal in the superpad. The time distribution shows the range of times of signals being collected by the RTPC. The 1D

occupancy ADC distribution is showing the distribution of ADC values for all pads, which is essentially a 1D representation of the 2nd histogram. We also studied how many hits and pads were in each event, to be sure the RTPC is collecting a reasonable amount of data in each event.

In addition to the monitoring histograms, we use CED (CLAS12 Event Display), software developed by Dave Heddle from CNU, which shows several different views of the CLAS12 detectors as they are collecting data. For the RTPC, there is a view which shows the readout padboard and highlights pads which saw a signal above the ADC threshold, which can be adjusted. An example of what the CED display looks like for the RTPC is shown in FIG. 27. The red squares in this image are the pads which had a signal above the threshold with time information as the third dimension. There are a few areas of this display which show tracks, and this is essentially what the track finder sees as well. This is a great way to see how the track finder is able to sort hits into tracks. Another example with crossing tracks can be seen in FIG. 28. It is also possible to monitor all the operating information about the RTPC, such as the High Voltages, the Gas information, the electronics overview and the interlocks.

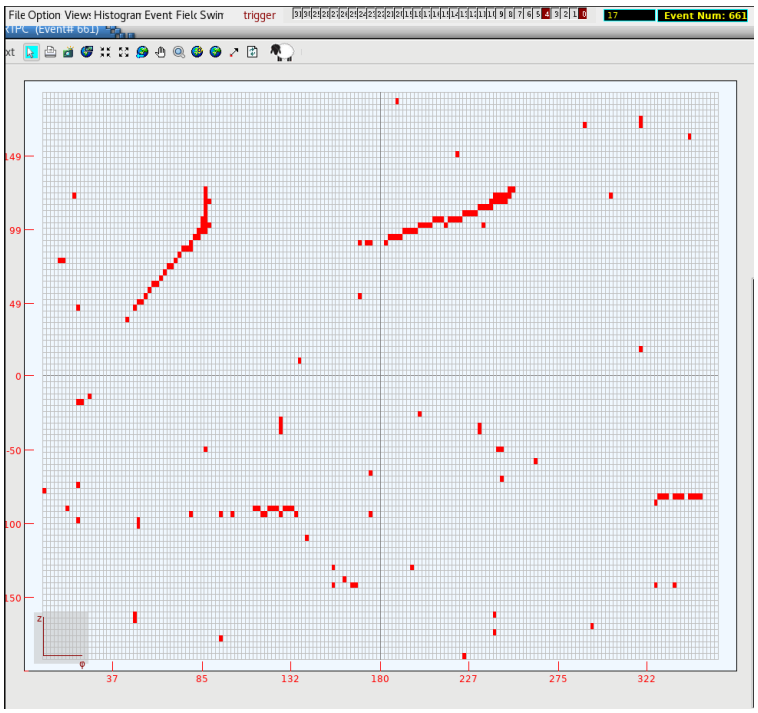

FIG. 27: Example of the RTPC view of the CLAS12 Event Display. This is event 661 of run 12943, a Hydrogen run with 10.4 GeV beam energy.

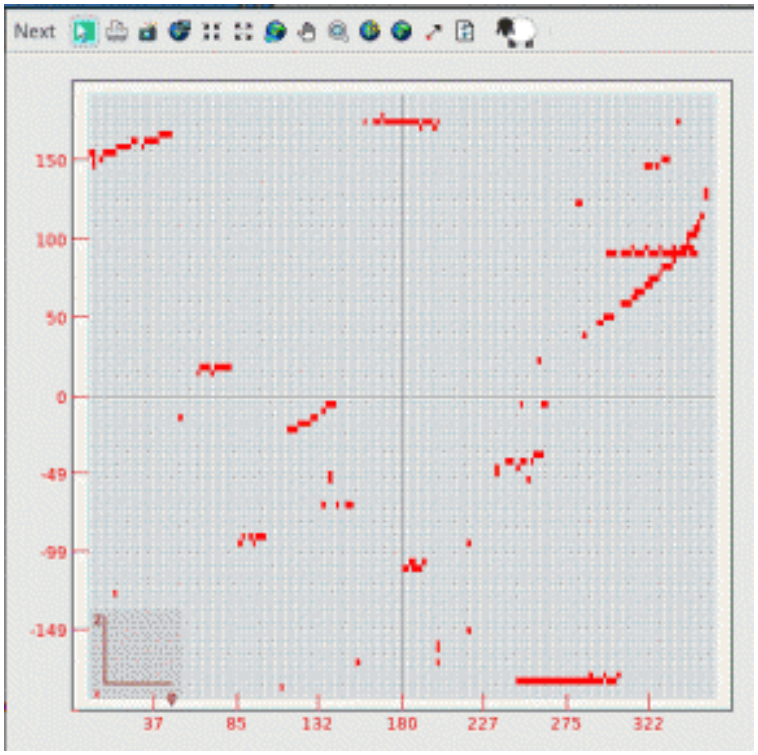

FIG. 28: Example of the RTPC view of the CLAS12 Event Display. This is event 2049 of run 12937, a Deuterium run with 10.4 GeV beam energy.

5.4 ANALYSIS

In order to analyze data from the experiment several initial steps must be taken. The software described in this thesis determines most of the variables needed to describe the particles in the RTPC, but in order for the RTPC reconstruction software to be able to use the information from the detector it must first be decoded. The decoder was provided by the COATJAVA software team at Jefferson Lab, and essentially interprets the raw data files from the experiment, and converts them into the HIPO bank structure we need to be able to run the reconstruction software. The decoder requires that we have a definition of every readout pad in our detector located in a database. The decoder will then map the signals from the RTPC electronics to the readout pads they came from using a translation table which provides the relationship between the electronic modules (FEMs) and the readout pads themselves. Once the information from the RTPC is properly decoded, the HIPO file is passed to the reconstruction. There is a decoder for all the detectors of CLAS12, and the resulting data files are in HIPO format and contain the relevant banks for all CLAS12 detectors and the RTPC. The output of the reconstruction will be a HIPO file which contains both the banks produced by the decoders for each detector, and the banks which are produced by the respective reconstruction software for each detector. For analysis, we are not only interested in the RTPC banks but also the banks for the rest of CLAS12. These banks must be used to extract information about all the other particles which were not detected in the RTPC. For example, the scattered electron can be tracked and reconstructed in the Drift Chambers of CLAS12.

Prior to studying the real experimental data, the software can be tested on simulated data. In order to study simulated data, the process is slightly different from real data. The output of GEMC is not decoded, but rather converted from EVIO to HIPO (see section 4.2). The data from the simulation also requires simulating signals internally in the reconstruction (see section 4.8.1). An example of simulated data can be seen in FIG. 29.

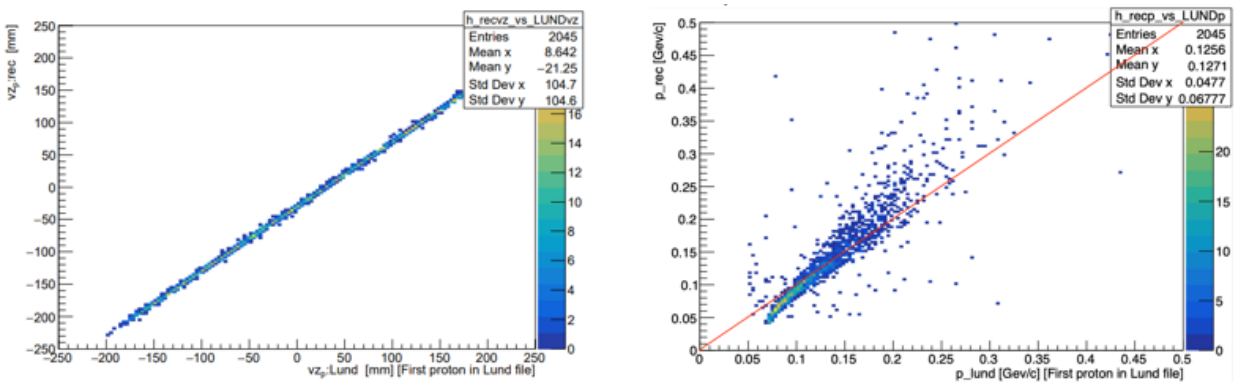

FIG. 29: Examples of simulated data analysis, courtesy of Madhusudhan Pokhrel. The left image shows a comparison of the reconstructed vertex of simulated particles in the RTPC, to the original simulated vertex position. There is a 3 cm shift due to the reference frame in the simulation compared to the reference frame of the reconstruction. The right image shows a comparison of the reconstructed momentum to the "thrown" momentum in the simulation. Notice how for low momenta there is a disagreement due to the fact that particles with such low momenta lose enough energy before entering the drift region that they are reconstructed with lower momenta.

In order to study the output of the RTPC reconstruction software using real data coming from the experiment, a basic analysis script was written to produce plots of the kinematic variables mentioned previously (see FIG. 30-41), and a script which displays pictures of tracks (like the ones shown in FIG. 42-44 for example). The analysis script is broken down into a few parts. First, all the data banks are read in from the files produced by COATJAVA. In order to read these data banks, one needs to use a reader which can open HIPO files. I chose to use C++/ROOT for this, but it's also possible to read HIPO files using JAVA. Both come with their own advantages and disadvantages. The analysis script reads in a few banks, including the banks produced by the RTPC reconstruction software, the bank for the Calorimeter detectors, the bank for the Cerenkov detectors, and the bank which contains basic kinematic information about all the particles detected by CLAS12. Then we loop over all the particles in CLAS12, looking specifically for an electron. The reason we need to find the electron in the event is so that we know which electron provided the trigger. In the case of H(e,$e'p$) events where the proton is struck at low beam energies, we can directly study the predicted and reconstructed proton track to calibrate our detector. When we move on to 5-pass physics runs on Deuterium, we can study D(e,e'p)X where, as previously mentioned, the kinematics of the proton are selected so that we know we are scattering on the neutron. We

use the calorimeter information to make fiducial cuts on the three layers of the calorimeter. These fiducial cuts ensure that we are simultaneously avoiding the edges of the detector, and also avoiding areas of the detector where particles we are not interested in will interact. With the HTCC, we are selecting events which include more than 2 photoelectrons to select electrons which produce Cerenkov light in the HTCC.

Assuming we find an event with a good electron, and the event passes our fiducial cuts, then we use some of the electron kinematics to make additional cuts. E.G., we require selected electrons in the forward detector with energies at greater than 20% of the beam energy. Events which pass all cuts are used to create a large number of histograms, some of which are shown in FIG. 30. In the following, we will go through some of the histograms included, and look closer at what each one represents. These histograms were created by studying run 12422 in which a Hydrogen target was used with a beam energy of 2.2 GeV.

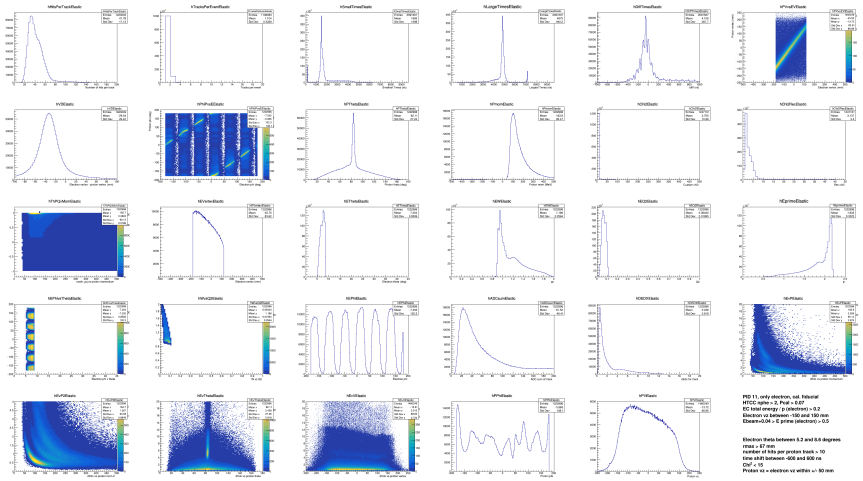

FIG. 30: Example histograms from run 12422 using the analysis software.

Several of the following quantities are calculated by the CLAS12 reconstruction software, based entirely on the CLAS12 forward detector signals for the electron. FIG. 31 shows the W vs Q^2 distribution, where for elastic scattering in the case of a proton target, we expect an elastic W peak at the mass of the proton, near 1 GeV. For elastic scattering at small (forward) electron scattering angles, we expect a fairly small Q^2. Larger Q^2 values correspond to a larger transfer of momentum to the proton. The events selected for further analysis are those below $W = 1$ in FIG. 31, and at low Q^2.

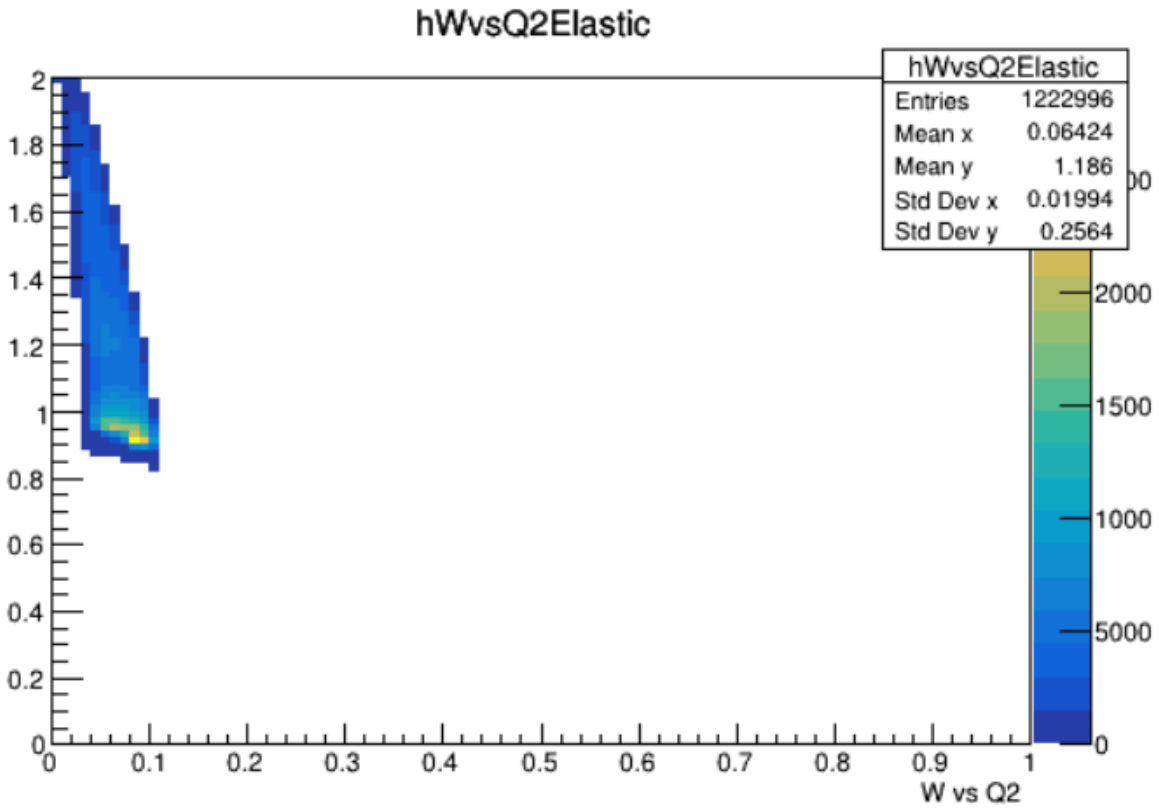

FIG. 31: 2D histogram comparing W to Q2.

FIG. 32 shows the reconstructed vertex of the electron. This is calculated using the known magnetic fields, and "swimming" the reconstructed track back to the vertex. The RTPC is sensitive only to the region between -200 mm and 200 mm; however, to avoid the region close to the endplates and target entrance/exit windows, further cuts are made to study electrons with a vertex within a range of -150 mm to 150 mm (the actual selected range is shifted by the known offset of 30 mm).

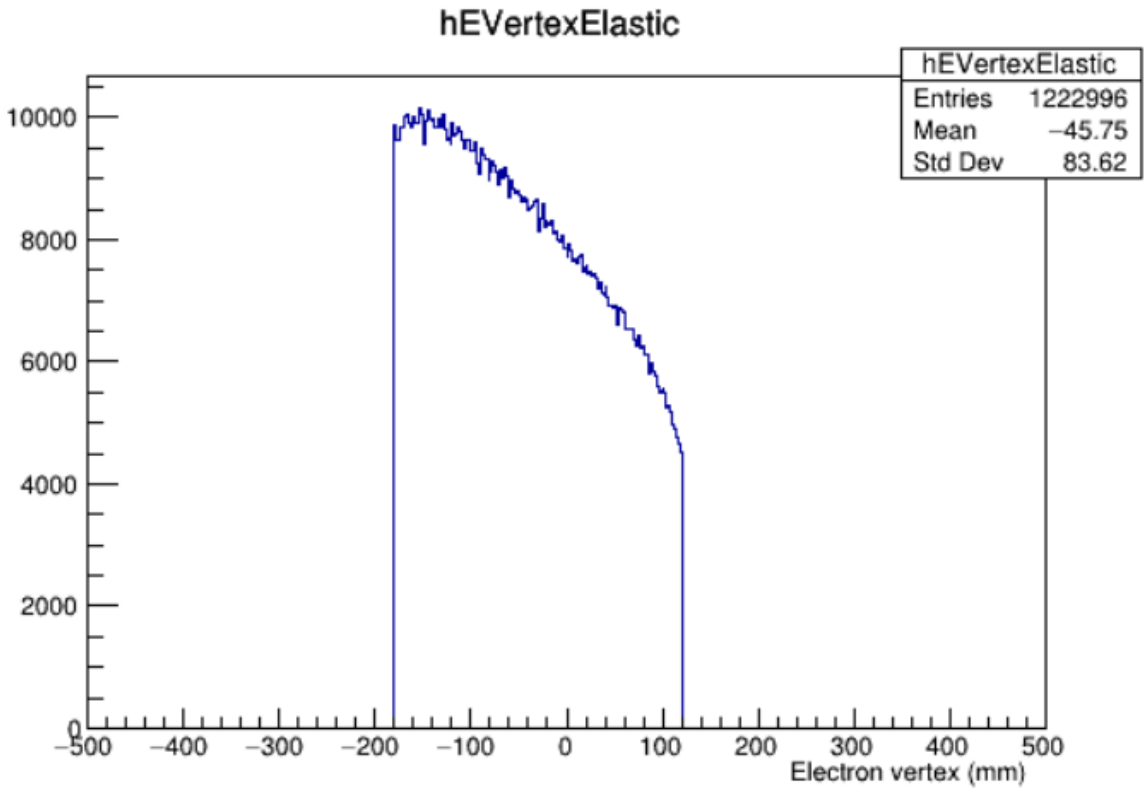

FIG. 32: 1D histogram showing the distribution of the scattered electron vertex in mm.

FIG. 33 shows the reconstructed azimuthal angle ϕ for the electron, as a function of the polar (scattering) angle θ. Recall that the forward detector angular acceptance is between 5 and 40 degrees. As explained above, we cut on low forward angles no greater than 8.6 degrees to select events with small momentum transfer to the proton. The 6 sector nature of the forward detector of CLAS12 can be seen here. We expect a mostly even distribution across all 6 sectors, as elastic scattering should have no phi-dependence.

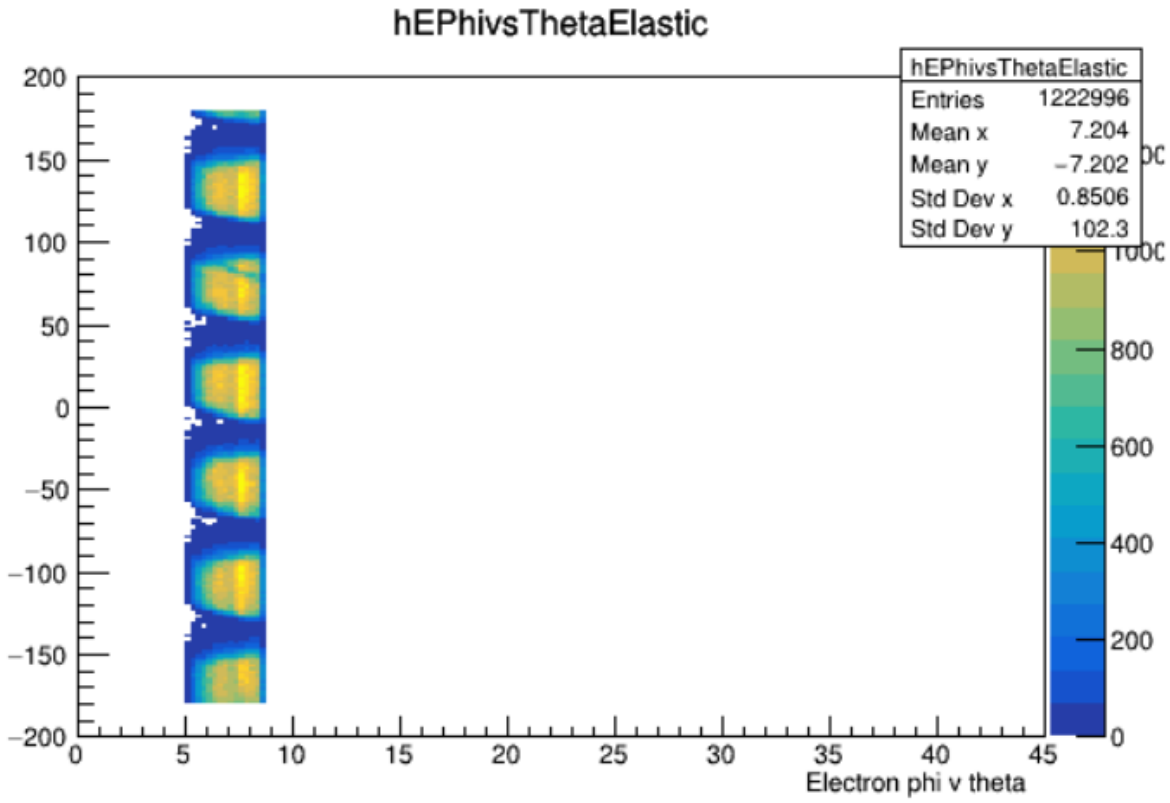

FIG. 33: 2D histogram comparing the ϕ and θ angles of the reconstructed electron.

FIG. 34 and FIG. 35 are used to study, and also calibrate, the timing parameters a_t and b_t in Eq. (11), for in-time proton tracks. These two histograms show that in the majority of cases tracks start at around 1500-1600 ns, which is the sum of all the time offsets between the trigger and the RTPC electronics, and stop around 4700 ns, which represents the maximum drift time from the cathode to the first GEM in addition to this offset. The time shift for each track is calculated by taking the difference between the largest time of the track and the maximum drift time. The resulting distribution of time shifts applied to all tracks is plotted in FIG. 36. In the case of Hydrogen, where we mostly expect only the proton struck by the electron to be reconstructed, the reconstruction is often only dealing with a single in-time particle and thus the average time shift is centered near zero.

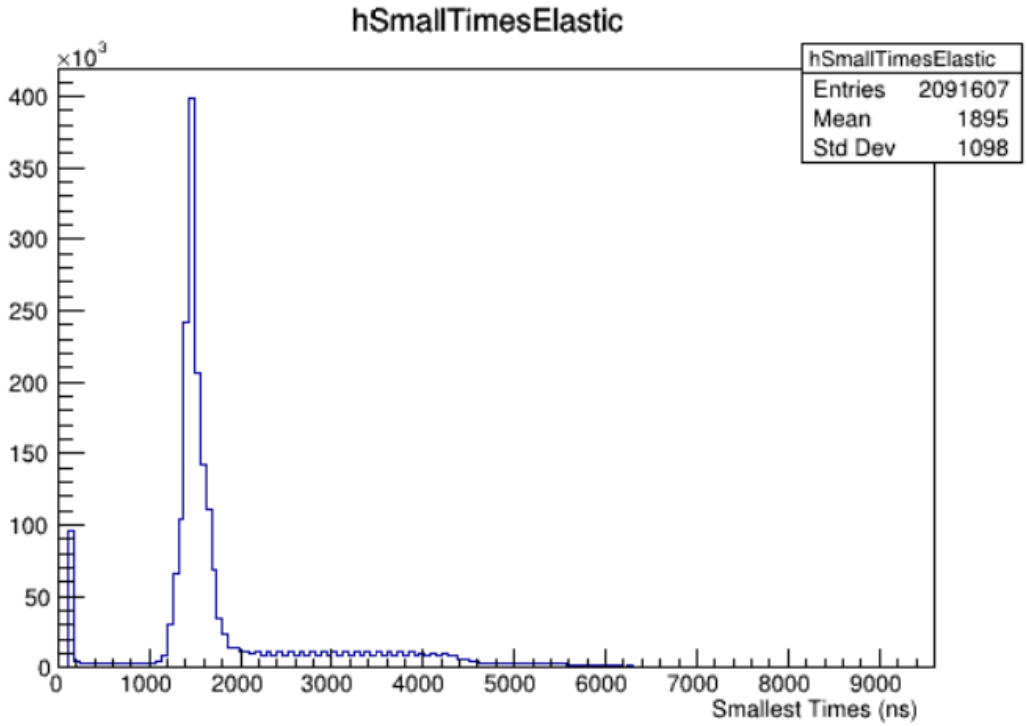

FIG. 34: 1D histogram showing a distribution of the smallest times for all tracks in the run.

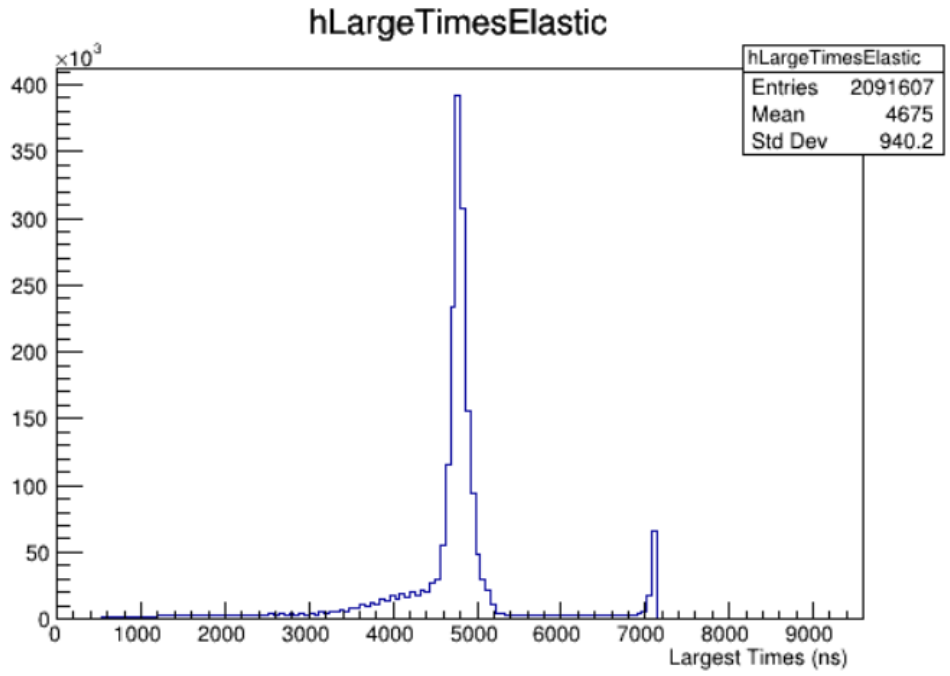

FIG. 35: 1D histogram showing a distribution of the largest times for all tracks in the run.

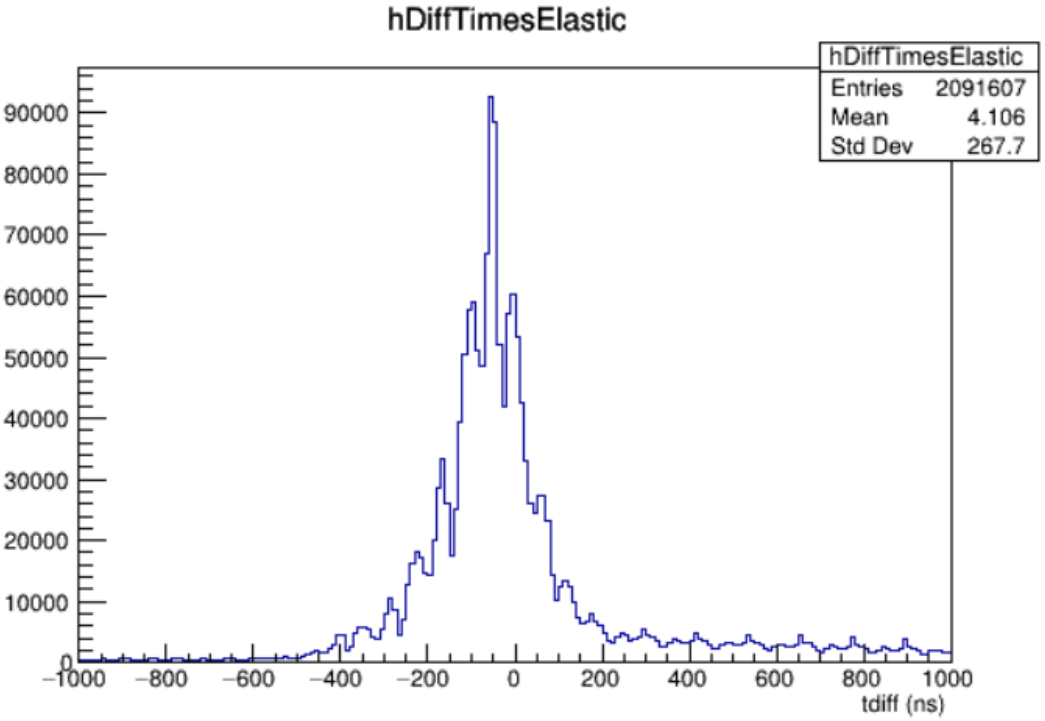

FIG. 36: 1D histogram showing a distribution of the time shift of each track in the run.

In FIG. 37, the distribution shows how many hits we have per track in the RTPC. The distribution peaks at around 40 hits per track, which is what is expected for a good track which extends from the cathode out to the first GEM, and doesn't curve too much, where on the other hand, the tracks which contain upwards of 70 hits or more would be tracks which don't travel to the GEM, and instead curve back towards the cathode. Cases like this, or cases in which multiple tracks are improperly merged, would have many more hits than the average. Similarly in FIG. 38, the number of tracks per event are shown. Run 12422 was on a Hydrogen target, and we would not expect many more than 2 particles in an event, and in most cases would only expect a single particle, the proton, as the electron should not be detected by the RTPC.

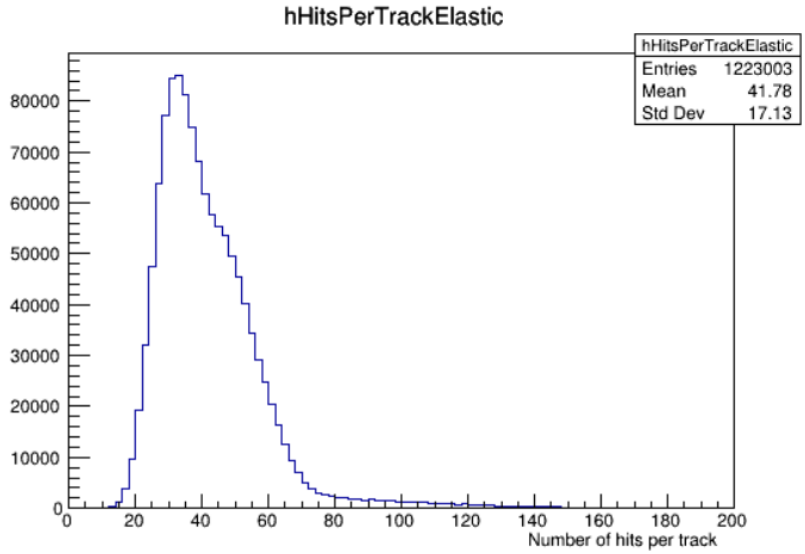

FIG. 37: 1D histogram showing a distribution of how many hits each track found by the reconstruction has.

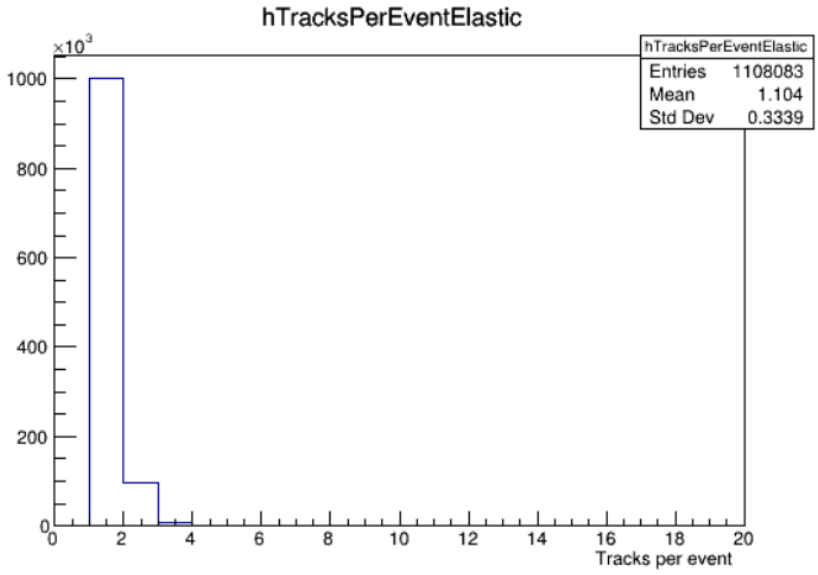

FIG. 38: 1D histogram showing a distribution of, for each event in the run, how many tracks were found in the event.

In FIG. 39, the reconstructed vertex of the electron and proton are compared. The offset of the vertex here can be used to calibrate our detector, as we would expect a vertex difference of zero. We can use the difference between proton and electron vertex to select protons

which originate from the same point along the beam as the electron. The coordinate system of the RTPC and CLAS12 are not the same, and include a 30 mm offset, so here we see a difference between the vertex of each particle of approximately 30 mm.

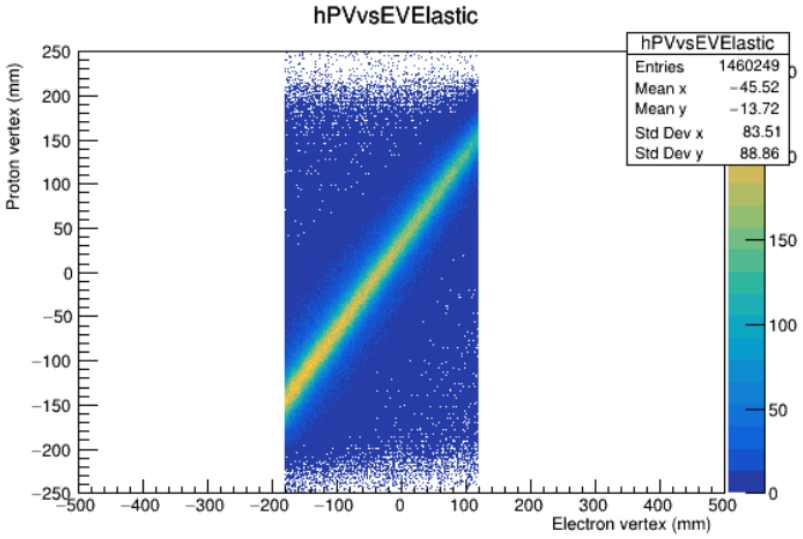

FIG. 39: 2D histogram plotting the proton vertex vs the electron vertex in each event.

The proton momentum histogram, FIG. 40, demonstrates that our detector is capable of reconstructing protons with momenta lower than 100 MeV, which is important for spectator tagging when studying deep inelastic scattering. There is a sharp cutoff at approximately 40 MeV which would be the reconstructed momentum of a particle that barely enters the drift region of the RTPC and then bends back out. As FIG. 29 shows, these correspond to protons starting out around 70 MeV/c in the target and then losing energy until they reach the drift region. Also, there are some bins which seem to show particles with negative momenta. This is due to the fact that the Helix Fitter changes the sign of the radius of curvature for particles with negative charge which bend in the opposite direction, and thus the calculated momenta are negative as shown here.

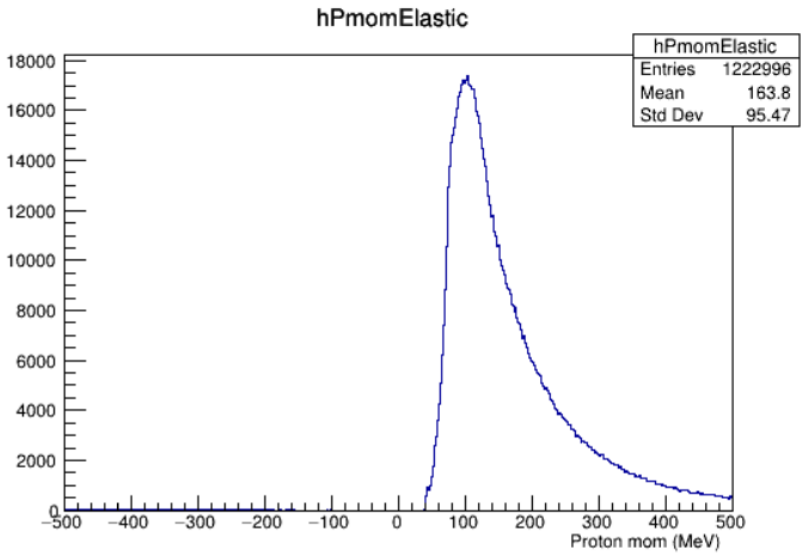

FIG. 40: 1D histogram showing the distribution of momentum in MeV for protons reconstructed in the RTPC.

FIG. 41 shows the measured signal size along the entire track, divided by the track length, versus the reconstructed proton momentum. The signal size is proportional to the number of liberated ionization electrons, and since each ionization requires an average energy of about 40 eV, the quantity show on the vertical axis is proportional to the energy loss per unit length, dE/dx, of the proton in the ionization region. This energy loss is a well-known function of the velocity of a particle (Bethe-Bloch formula) [28], and by comparing momentum with energy loss, the mass of the particle can be inferred. Hence, FIG. 41 allows us to select different particles, using the dependence of dE/dx on the particle momentum. Notice how for 2 GeV, there is a significant band at lower dE/dx in the momentum range of the protons we are interested in. Recall that the target here is Hydrogen, and we expect to see only protons in our detector, however due to potential scattering on the target walls and endcaps of our detector, there are occasionally higher mass particles detected which explains the higher energy bands shown here. We can easily select on protons by making dE/dx cuts here.

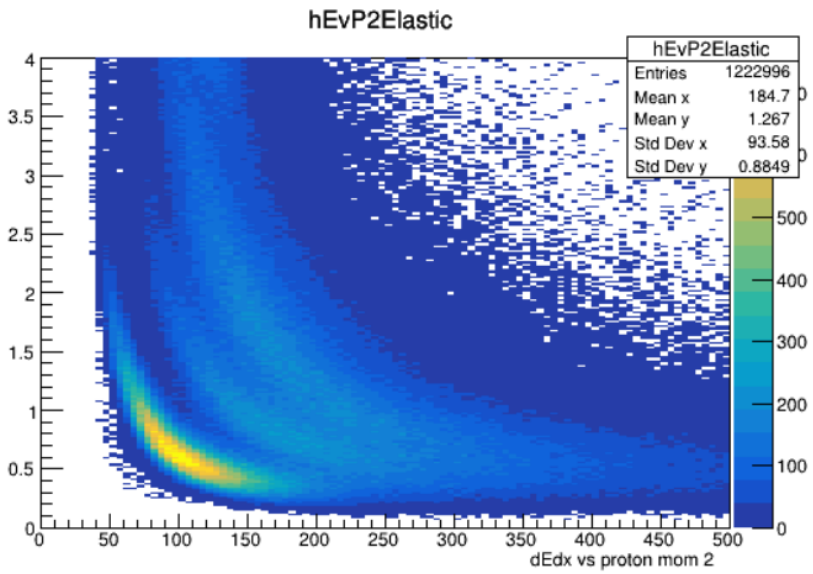

FIG. 41: 2D histogram showing the dE/dx as a function of proton momentum.

In addition to the analysis software I wrote a script which is responsible for displaying track pictures. This script performs two steps. First, it reads the two RTPC banks, and looks specifically for the helix fit information about each track as well as the list of hits for the track. Then the position of each hit is used to display the track in all 3 views, and the parameters of the helix fit are used to draw lines on top of the same graph. FIG. 42 is an example output of this script. Here you can see two particles in the RTPC. Each is colored automatically based on the track ID of each individual ionization. Then the helix fit parameters are used to draw a line on top of this drawing, to show that the helix fitter accurately fits these tracks. The output file of the event contains the radius and center of each helix, and this information is used to draw the lines. The individual hits are plotted using the given x and y coordinates for each hit in the output file. All this information is stored in the output banks of the reconstruction software. Both tracks start at the cathode and end at the first GEM layer. Additional views of such an event can be seen in FIG. 43 and FIG. 44.

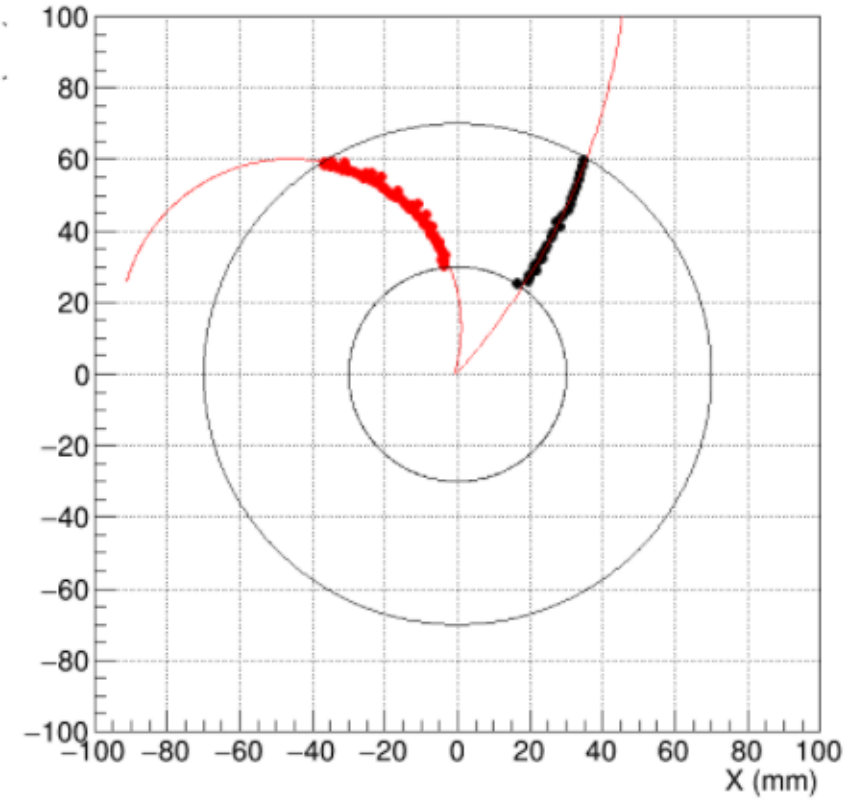

FIG. 42: Track view in X-Y coordinates. This plot includes the individual ionization points which are automatically colored and shaped based on the track they belong to, and the overlaid helix fits. Image from run 12411, beam energy 2.18 GeV, H2 target.

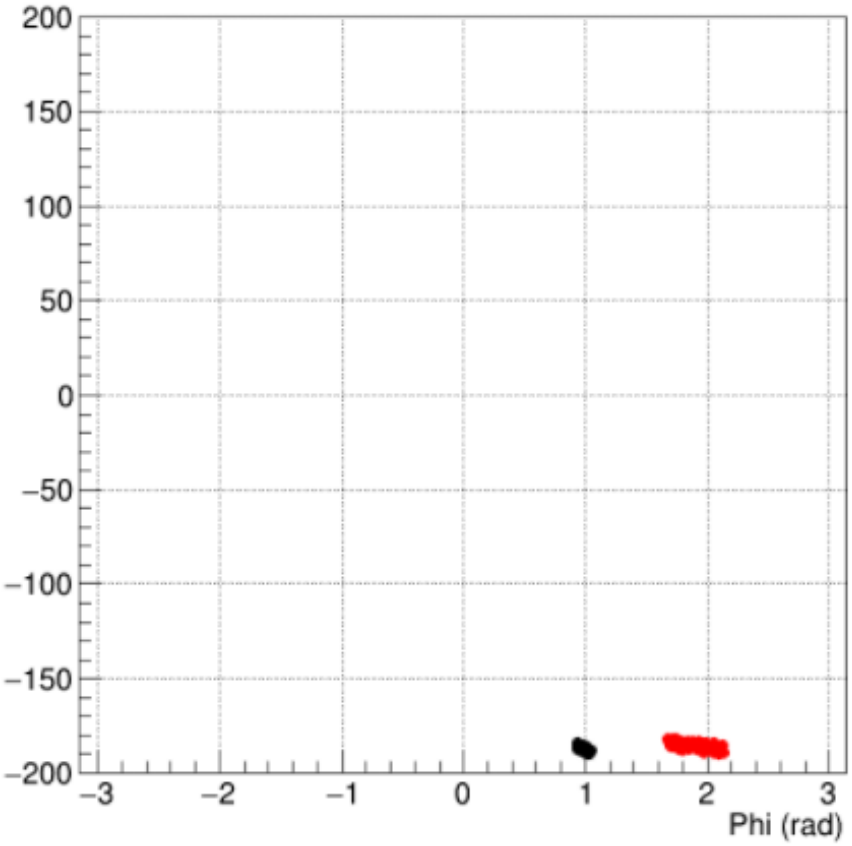

FIG. 43: Track view in ϕ-Z coordinates showing the individual ionization points which are automatically colored and shaped based on the track they belong to. Image from run 12411, beam energy 2.18 GeV, H2 target. In this view, one can see the tracks how they would look projected radially outward to the readout padboard (ranging from z = -200 mm to z = +200 mm and ϕ = -180 degrees to ϕ = +180 degrees), note that this is NOT a rendering of the actual pads that saw these particular tracks, as the ionization electrons drift sideways in the magnetic field of the CLAS12 solenoid.

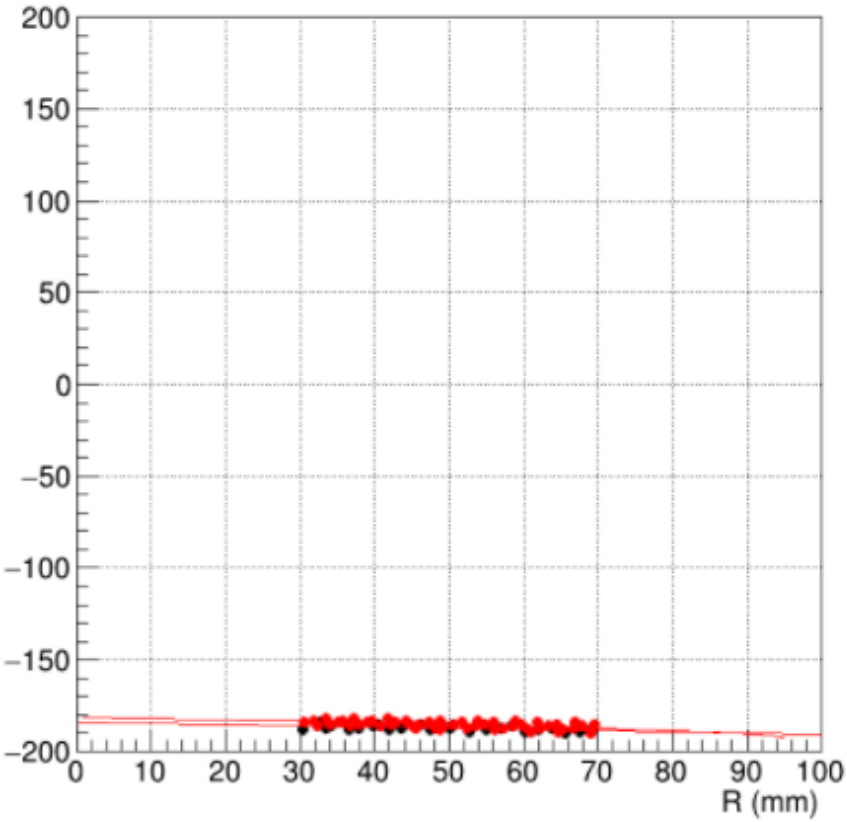

FIG. 44: Track view in Z-R coordinates. Again the plot includes the individual ionization points which are automatically colored and shaped based on the track they belong to, and the overlaid helix fits. Image from run 12411, beam energy 2.18 GeV, H2 target. Both tracks start at the cathode (R = 30 mm) and extend to the first GEM (anode) at 70 mm. Both are extrapolated to a very similar vertex at R = 0 and z ≈ 180 mm; however, FIG. 42 shows that they are two distinct tracks.

Moving on to higher energies, and more complex targets, FIG. 45 shows examples of 10.4 GeV data taken with a deuterium target. The top left histogram shows that, while there is much more background, there is still a correlation between the incident electron and some tracks which must correspond to the spectator protons. The top right image shows how we've selected specifically the proton band using the energy loss, dE/dx. The bottom left image shows the reconstructed proton momentum on the horizontal axis and the cosine of the angle between the momentum transferred by the scattered electron and the direction of the outgoing proton. Since we are interested in low-momentum, backward going spectator protons, the box indicates the selection of such events. The bottom right image shows a comparison of Q^2 vs x for a small subset of data, showing we have bins out to $x \approx 0.7$.

Again this is only for a single run, and later in 2021 we expect to see more statistics after analyzing the full dataset.

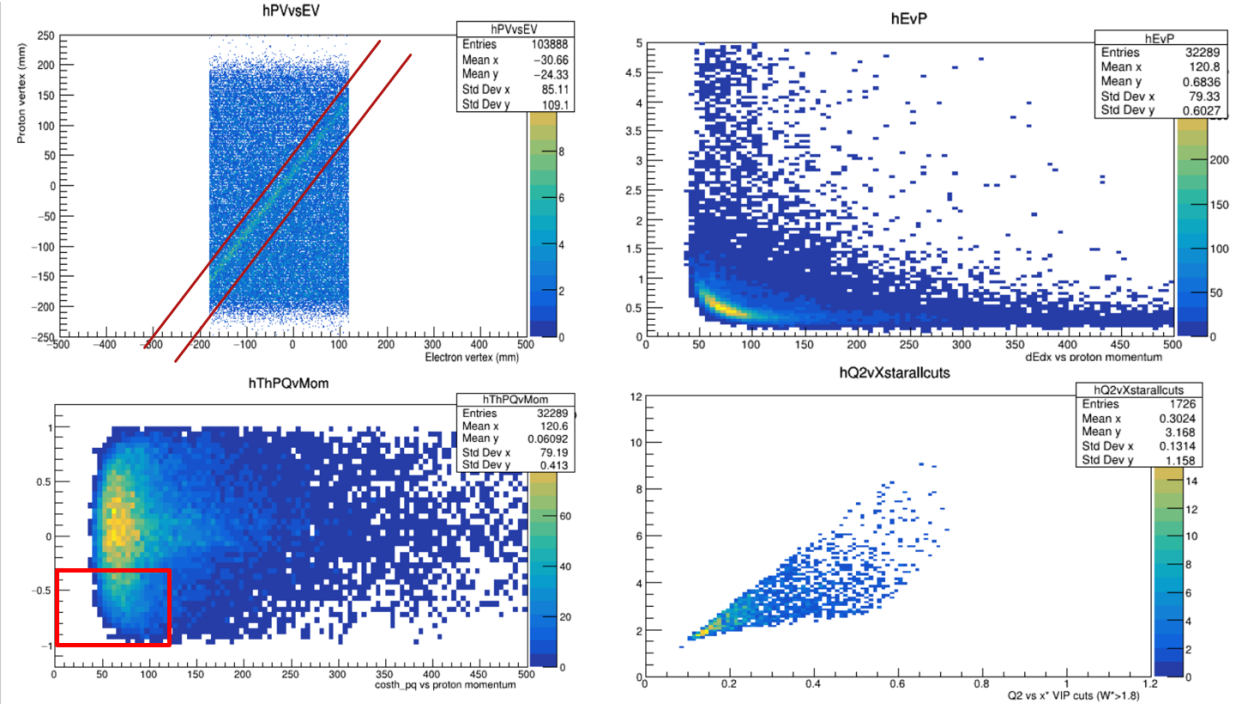

FIG. 45: Four histograms of 10.4 GeV data with a deuterium target. The top-left image is a histogram comparing the proton z-vertex to the electron z-vertex. The top-right image shows the dE/dx as a function of proton momentum. The bottom left image shows the cosine of the scattering angle as a function of the proton momentum. The bottom right image shows Q^2 as a function of x. These images are courtesy of Sebastian Kuhn.

Finally FIG. 46 shows an example of an event from the experiment which contains several tracks. This event in particular shows many of the special cases that might show up across all the data in the experiment. First, notice the green track at the top of the view, and the red track at the bottom right of the view. These are two perfect cases, in which the track extends from the cathode (inner radius) all the way to the first GEM layer (outer radius). The fits for these tracks are quite good, and the momenta of these particles would be well reconstructed in these cases. This event also features some tracks which were not so well reconstructed. The green track at the bottom of the figure for instance extends past the outer radius of the drift region, which is likely due to extra hits being sorted with this track.

Or consider the blue track, in the bottom left corner, which is likely two tracks that were improperly merged. These are of course side effects of the way that track merging works. It's much easier to see the issues with tracks such as the blue one when looking at FIG. 47, where it's obvious that one of these tracks was nearly constant in z, and the other track crossed at similar ϕ and z, and they were merged together. The disentangler would take care of a case such as this, because it would be easy to see that these two tracks are different. This was an event in which the disentangler was disabled while we made improvements to its design.

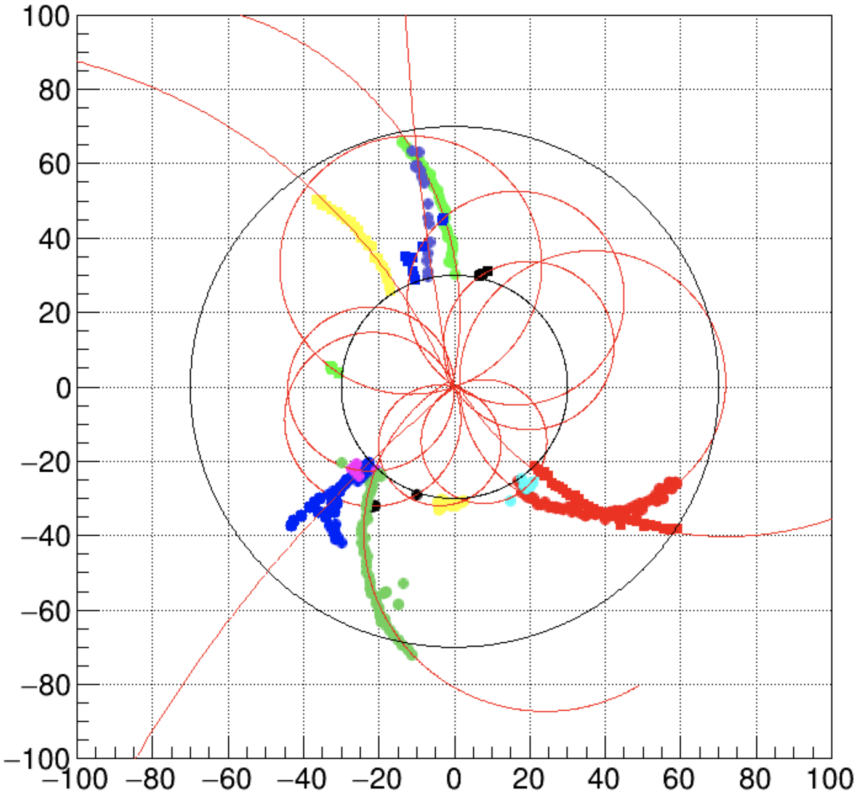

FIG. 46: Track view in X-Y coordinates. Includes the individual ionization points which are automatically colored and shaped based on the track they belong to, and the overlaid helix fits.

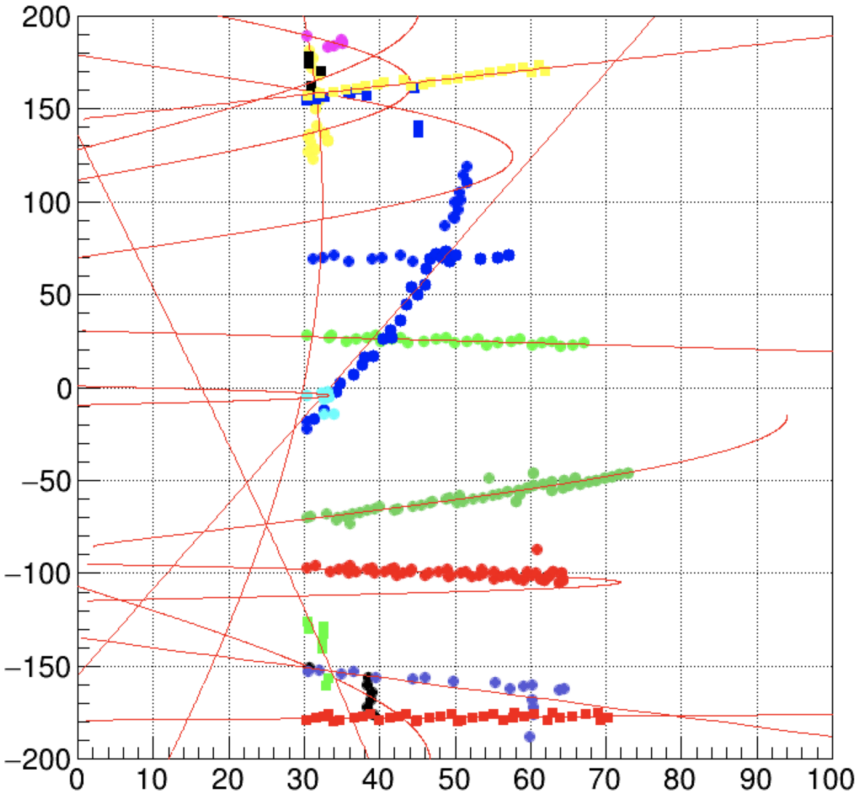

FIG. 47: Same track view as FIG. 46 this time in R-Z coordinates.

5.5 SUMMARY AND FUTURE WORK

Currently the software is being used to process all the data from BONuS12. The experiment successfully used the reconstruction software described in this thesis, and it performed efficiently, running at a similar processing time to that of the other CLAS12 detectors. The software worked to both monitor the detector while running the experiment, and also to help prepare large sets of data for analysis. However, there are more changes one could make to the reconstruction software in the future, to improve it, or perhaps modify it to support other RTPC designs.

One planned update to the reconstruction software is the addition of a Kalman filter. The Kalman filter would use the results of the helix fit to determine a much more accurate track fit. This is because a helix fitter tries to fit a circle to the x-y projection of the track, but as we know the particles lose energy as they travel through the drift region. This results in a non-constant radius of curvature, which could be more accurately fit with the Kalman Filter's "swim" from hit to hit rather than just forcing a helix with constant radius onto the track.

More work can also be done to improve the disentangler. Disentangling crossing tracks continues to be a real challenge. Writing an algorithm which works for every single case in which multiple tracks cross, and does not confuse single tracks with multiple crossed tracks is a difficult problem to solve. This is an area where updates could be made to the way we disentangle tracks. Equally important to disentangling the tracks is rebuilding tracks which are broken in this process. There are certainly cases shown in the analysis where single tracks appear in the final result as two separate tracks due to not being successfully rebuilt after being disentangled from other tracks.

Particles with lower energy that don't make it all the way to the GEMs and instead curve back towards the cathode are the cases which need the most additional treatment. Whichever way these tracks are fit, the energy loss must be handled correctly. It is possible that treating two halves of a back-bending track separately may improve the results.

Culling hits which clearly don't belong to tracks, and tracks which contain too many such outliers is a relatively new concept added to the software fairly recently. This is another area where future improvements could be made. Currently, the results of the reconstruction software are improved by the existence of such a culling, but due to the fact that some tracks are inevitably lost in the process, there is more work to be done.

The last area which could use more enhancement is the way tracks are time shifted to correct for the time relative to the trigger. Currently, these tracks are compared using the largest times and shifting the entire track relative to the expected "in-time" largest time of a track. There are downsides to this method, however. For instance, the in-time track doesn't necessarily have an ionization which occurs right at the cathode, and so it will be shifted a small amount, while the out of time tracks will be shifted much more heavily. This results in a marginal error where the timing resolution will be affected. On the other hand, in the case of back-bending tracks, shifting relative to the smallest time would force a back-bender to be shifted all the way to the first GEM, resulting in a complete distortion of such a track. In general back-benders need additional treatment as stated above.

While some of these improvements will be implemented for a future "pass2" analysis of the entire data set, the software in its present state is sufficiently complete and reliable for a first "pass1" analysis of the data taken in the summer, which will lead to a first glimpse of the neutron structure function $F_2^n(x)$ at large x, the goal of the BONuS12 experiment.

BIBLIOGRAPHY

[1] F. Halzen and A. Martin, *Quarks and Leptons: An Introductory Course in Modern Particle Physics* (John Wiley and Sons, New York, 1984).

[2] C. Amsler et al., "Review of Particle Physics (Particle Data Group)", Physics Letters B **667**, 1 (2008).

[3] *Elementary particle*, `https://en.wikipedia.org/wiki/Elementary_particle`, Sept. 2019.

[4] L. W. Whitlow et al., "Precise Measurements of the Proton and Deuteron Structure Functions from a Global Analysis of the SLAC Deep Inelastic Electron Scattering Cross-Sections", Phys. Lett. **B282**, 475 (1992).

[5] T. Hou et al., "New CTEQ global analysis of quantum chromodynamics with high-precision data from the LHC", Physical Review D **103** (2021).

[6] A. Accardi et al., "Uncertainties in Determining Parton Distributions at Large x", Phys. Rev. D **84**, 014008 (2011).

[7] M. Amarian et al. (CLAS Collaboration), "The Structure of the Free Neutron at Large x-Bjorken", CLAS-PROPOSAL **PR12-06-113** (2006).

[8] G. Ricco et al., "Bloom-Gilman Duality of Inelastic Structure Functions in Nucleon and Nuclei", Physical Review C **57**, 356 (1998).

[9] D. F. Geesaman, K. Saito, and A. W. Thomas, "The Nuclear EMC Effect", Ann. Rev. Nucl. Part. Sci. **45**, 337 (1995).

[10] S. Tkachenko et al. (CLAS Collaboration), "Measurement of the Structure Function of the Nearly Free Neutron Using Spectator Tagging in Inelastic ^{2}H$(e,e'p_s)X$ Scattering with CLAS", Phys. Rev. C **89**, 045206 (2014).

[11] O. Hen, M. Sargsian, L. B. Weinstein, et al., "Nuclear Physics. Momentum Sharing in Imbalanced Fermi Systems.", Science (New York, N.Y.) **346**, 614 (2014).

[12] N. Baillie et al. (CLAS Collaboration), "Measurement of the Neutron F_2 Structure Function via Spectator Tagging with CLAS", Phys. Rev. Lett. **108**, 199902 (2012).

[13] V. Palli et al., "Slow-Proton Production in Semi-Inclusive Deep Inelastic Scattering Off the Deuteron and Complex Nuclei: Hadronization and Final-State Interaction Effects", Phys. Rev. C **80**, 054610 (2009).

[14] H. C. Fenker et al., "BoNuS: Development and Use of a Radial TPC using Cylindrical GEMs", Nucl. Instrum. Meth. **A592**, 273 (2008).

[15] S. Tkachenko, N. Baillie, S. Kuhn, et al., "Measurement of the Structure Function of the Nearly Free Neutron using Spectator Tagging in Inelastic ^{2}H(e, e ' p(s)) X Scattering with CLAS", Phys. Rev. C **89**, 045206 (2014).

[16] K. A. Griffioen et al., "Measurement of the EMC Effect in the Deuteron", Phys. Rev. **C92**, 015211 (2015).

[17] I. Niculescu et al., "Direct Observation of Quark-Hadron Duality in the Free Neutron F_2 Structure Function", Phys. Rev. **C91**, 055206 (2015).

[18] V. Burkert and others [CLAS12 Collaboration], "The CLAS12 Spectrometer at Jefferson Lab", Nucl. Inst. and Meth. A **959**, 163419 (2020).

[19] R. Fair et al., "The CLAS12 Superconducting Magnets", Nucl. Inst and Meth. A **962**, 163578 (2020).

[20] M. Mestayer et al., "The CLAS12 Drift Chamber System", Nucl. Inst and Meth. A **959**, 163518 (2020).

[21] D. Carman et al., "The CLAS12 Forward Time-of-Flight System", Nucl. Inst and Meth. A **960**, 163629 (2020).

[22] D. Carman et al., "The CLAS12 Central Time-of-Flight System", Nucl. Inst and Meth. A **960**, 163626 (2020).

[23] M. Ungaro et al., "The CLAS12 Low Threshold Cherenkov Detector", Nucl. Inst and Meth. A **957**, 163420 (2020).

[24] G. Gilfoyle and K. Sherman, "Geometry Update for the Electromagnetic Calorimeter in CLAS12", CLAS12, 2014 (2014).

[25] G. Asryan et al., "The CLAS12 Forward Electromagnetic Calorimeter", Nucl. Inst and Meth. A **959**, 163425 (2020).

[26] P. Baron et al., *DREAM User Manual, version 3.0.*

[27] N. Dzbenski, "Simulation and Development of the Radial Time Projection Chamber for the BONuS12 Experiment in CLAS12", PhD thesis (Physics, Old Dominion University, 2020).

[28] E. Segre et al., *Experimental Nuclear Physics*, Vol. 1 (John Wiley & sons, 1953).

[29] R. Dupré et al., "Measurement of Deeply Virtual Scattering off Helium-4 with CLAS at Jefferson Lab", (2021).

[30] M. Hattawy et al. (CLAS12 Collaboration), "Neutron DVCS Measurements with BONuS12 in CLAS12", (2019).

APPENDIX A

GLOSSARY OF RECONSTRUCTION TERMS

1. Signal – A readout electronics signal which represents the accumulated charge on a readout pad. These signals are discretized by the DREAM electronics into several integrated 120 ns time bins.

2. Hit – A hit is defined as a 3-dimensional time-dependent element in tracking. The way a hit is defined relative to time and space coordinates changes depending on the software context. A hit contains all the relevant info at the current stage of tracking which will always have a time and position component, as well as a corresponding readout pad ID and the ADC or energy value associated. A hit starts out as a single combination of readout pad ID, time bin, and ADC sample. Hits are first sorted in the 2-dimensional row-column plane of the readout pad board while also taking into account the 120 ns discretized time bins as a "third" dimension. In other words, two hits have their distances compared in row/column/time dimensions. Once all such hits are organized into tracks, these hits are then averaged together when they come from the same readout pad signal, into new composite time-averaged hits. Once these hits are reconstructed into the drift region of the RTPC they have r/ϕ/z/time dimensions. They are still always called hits, no matter which stage of tracking and which spatial dimensions they correspond to. This maintains a consistent track/hit association.

3. Track – A track is defined as a collection of hits. Similar to the hit definition, a track's time and space coordinates change throughout tracking, but a track always consists of a collection of hits, which associates the readout pad board to the track. Tracks start out by containing 120 ns time-discretized hits, but by the end of the TimeAverage class, the track will no longer be discretized in time, and rather is a collection of time-averaged hits (sometimes referred to as a "reduced track"). Eventually the track will be reconstructed into the drift region and fit to extract the kinematics of the particle. Once a track is reconstructed, the hits that make up the track form a representation of the locations of ionization electrons as the charged particle travels through the gas.

4. Tracklet – An incomplete track. Oftentimes a tracklet can be identified by identifying tracks which are not reconstructed into the drift region properly. Usually these tracklets will have incorrectly calculated kinematics. A tracklet could also be formed when a track leaves

the detector early by exiting through the endplate, or when the arrival time of the particle results in the particle either entering or leaving the time window of data collection before it makes a complete trip from cathode to GEM. Note that tracks are not required to travel from cathode to GEM as it is possible for low-momentum particles to curve back towards the cathode if they have low enough energies. Tracklets are most often broken tracks and are difficult to reconstruct and fit correctly.

5. TID – Track ID. Every track is assigned a unique ID which is based on the order in which the tracks are formed. Once a track has a unique ID, this ID is not used again in the same event. If a track is discarded because it is too short or not well reconstructed, the ID will be skipped in the output. If tracks are merged together then a new ID will be generated for the new track, and the old IDs will not be used. For this reason, it is possible that the highest track ID in the event can be a larger number than the number of particles detected in the event. This is also true for cases in which a particle is not tracked properly and may be interpreted as multiple smaller tracklets.

6. Map – The map name comes from the name of the Java standard object called a HashMap. The objects in the software which store tracks are extensions of HashMaps. Often times these objects will be simply referred to as a map, which is a collection of key-value pairs, or in the case of the RTPC reconstruction, TID-track pairs.

APPENDIX B

CCDB DICTIONARY

The CCDB constants database is used extensively to update constants in the reconstruction software without requiring the software to be rebuilt. Constants in the database can even be run-number specific. The RTPC has 4 separate tables in CCDB which are mapped to values in the software. These tables are "gain_balance", "time_offsets", "time_parms", and "recon_parms".

The gain balance table applies a specific gain value to each individual readout pad. This is used to calibrate the dE/dx of particle tracks. The table consists of columns labeled "sector", "layer", "component", and "gain". The RTPC only consists of a single sector, and the pads are divided into rows and columns where the rows 1-180 are mapped to layer, and the columns 1-96 are mapped to component. This naming convention is used only so that it matches with the other tables for the other detectors. Then the gain column consists of the gain value on each pad. In this way, each row of the table maps a specific gain calibration factor to a specific pad on the RTPC. This gain is then used in the software to calibrate the dE/dx.

The time offsets table is used in the reconstruction to apply time offsets to the constant a_t. The table is defined using the same sector, layer, component structure of the gain table. However, these sector, layer, and component values are now simply used as a key to access the specific values of the table, and have nothing to do with the readout pads for the detector. This is again a result of the tables needing to remain consistent with the tables of other detectors. The other 3 columns of this table are labelled "tl", "tp", and "tr" which are the three types of time offsets that can be applied to the reconstruction.

The time parms table is used to define the remaining track reconstruction parameters. Many of these parameters can be z-dependent up to z^4 so for this reason, the table has columns labeled $z0 - z4$. The first six rows of the table define the r and ϕ z-dependent coefficients of the reconstruction formulas. These rows in order are $a_t, b_t, c_t, a_\phi, b_\phi, c_\phi$. The last row defines the value of the time shift for the reconstruction. The value stored in z0 for this row is the time value compared to the largest time value of a track. The difference between this two values is the shift applied to all the hits in the track. For the last row only, the z1, and z2 columns are used to determine whether to compare to the shortest, or longest

time of the track respectively. These two values are binary. One of these should be 1 and the other 0, to define which part of the track you wish to compare to. By default the largest time is used, which is represented by a 1 in column z2.

The final table recon parms is used the most extensively. It contains parameters for the track finder, the track disentangler, the track reconstruction, and the helix fitter. The first row is used for the Track Finder. Dtm defines the number of adjacent time slices to search through. Dzm and Dphim define the largest allowed distance in z and ϕ for two hits to be sorted. ADCmin is the ADC threshold for hits to be considered and sorted. Hitmin is the minimum number of hits a track should have to not be discarded. The second row is used for the Disentangler. The values here are all used in the same way, except Dtm defines the max time between hits in the disentangler, since the disentangler no longer uses time slices. The third row contains a few different values. Dtm is the trigger window size for each event in ns. Dzm is the value which determines the maximum Chi2 a hit can contribute to a track before it is discarded. Dphim is the percentage of a track that, if removed by the Chi2 cut, causes the entire track to be discarded. The fourth and fifth rows define alternate parameters for the Track Finder and Track Disentangler in the case that two hits lie on opposite sides of the gap in the detector's ϕ coverage. The remaining two rows are used for flagging tracks to the Disentangler. Dtm in row 6 defines the maximum time a track can span before it is flagged. Dtm in row 7 is the maximum time a single pad in the track can span before the track is flagged. Examples of all these tables can be found at `https://clasweb.jlab.org/cgi-bin/ccdb/objects`.

VITA

David Payette
Department of Physics
Old Dominion University
Norfolk, VA 23529

David Payette received two Bachelor's degrees in Physics and Mathematics from the University of North Georgia in 2014. He began graduate studies at Old Dominion University in 2014 and received his Master's degree in Physics in January 2016, at which point he joined the Experimental Nuclear Physics group BONuS12, working for Professor Sebastian Kuhn, pursuing a Ph.D. in Nuclear Physics. The projected graduation is May 2021. David Payette is a member of the CLAS Collaboration at the Thomas Jefferson National Accelerator Facility (Jlab), and a member of the American Physical Society (APS).

Typeset using LaTeX.