UNIVERSITY OF NEW MEXICO

Multivariate Moments to Characterize Nuclear Recoil Track Directionality in Low-Pressure TPCs

by

Joshua D. Martin

A thesis submitted in partial fulfillment for the degree of Bachelor of Science in Physics

Research Advisor:
Dinesh Loomba
Department of Physics and Astronomy

May 2015

Abstract

The pursuit of understanding the mysterious dark matter is one of the challenges facing the modern physics community. There are compelling reasons to believe that the majority of our universe is of a form which does not interact electromagnetically or strongly, but does interact through the weak force and gravitationally. There have been, and are currently, many experiments designed to look for a sign of this ghostly matter. There has been a wealth of literature published on the possible nature of dark matter, and we have come a long way in placing limits on what its qualities must be.

The physical evidence in the cosmos suggests that each galaxy is permeated by a spherical dark matter halo which, on average, contains about five times more matter than the stars and gas. While this concentration does vary from galaxy to galaxy, observations at the largest scales indicate that dark matter makes up 84% of the matter density in the universe, with the remaining 16% consisting of ordinary matter made of protons and neutrons. Furthermore, a combination of a body of theories, simulations, and physical measurements suggest that the dark matter is mostly cold, and interacts only very rarely with normal matter. Simulations of galaxy formation indicate that the dark matter halo is not co-rotating with the stars and gas in the galactic disk. This suggest that as the solar system moves through the galaxy, it passes through the cloud of dark matter, creating a "wind"-like effect observable from Earth.

If observed, this wind would be very strong evidence of the dark matter's existence. To look for it, we need to find a way to spot the very rare interactions between the dark matter and normal matter. To this end, the DRIFT (Directional Recoil Information From Tracks) collaboration has developed a low pressure gas detector capable of tracking the recoils of gas nuclei which are struck by the dark matter particles in the theorized wind.

However, because of the very low predicted interaction rate of the dark matter, and the number of more physically familiar ways that gas particles may be caused to recoil, it is of utmost importance that we are able to determine in which direction the gas particles are recoiling with very high confidence. In this thesis we present a new algorithm for determining the direction of nuclear recoil tracks taken in CCD images from a prototype dark matter detector. We compare the new algorithm

to the current method of determining the directionality of nuclear recoils, and investigate methods for improving our measures of confidence in recoil directions. Finally, we present results from applying both algorithms to both simulations and data taken from the detector. These results indicate an improvement of the new algorithm over the old algorithm, and the new algorithm presents a new parameter which we found was correlated with the likelihood of determining the correct nuclear recoil direction.

Acknowledgements

Before we begin, I would like to extend my gratitude to several people. Firstly I would like to thank Professors Sally Seidel and Paul Schwoebel, my academic instructors this semester. They have been understanding and accommodating during these last few months, and have worked with me to help balance my research and academic pursuits.

Secondly, I would like to thank Nguyen Phan. He has always been available for questions and discussion as this research progressed. He helped me feel welcome when I joined the research group, and provided both support and helpful feedback as this research progressed. I would also like to thank the newly conferred Dr. Eric Miller. Over the last year I took over a third of his office, and interrupted his thought process with both obvious and non-obvious questions. He was willing to act as a sounding board for ideas both good and bad, and provided suggestions and criticism for the work presented here.

I would also like to thank my family and friends for their patience over the last year. My family has been an endless supply of support and encouragement, and I would not be where I am without them. Thank you for everything you have done for me, Mom and Dad. Every Saturday my friends and I meet for drinks, games, and the enjoyment of each other's company. Each of them has a perspective and skill set that has brought new ways of thinking into my life. My Saturdays have been the highlight of my weeks, and without them I don't think I could have made it through this work.

I also thank Evelynn Moore, my wonderful fiancée. She has been a strong support throughout my education. She has been understanding and encouraging, and without her help, I would not have completed this work.

Finally, I would like to thank Professor Dinesh Loomba. I consider myself truly lucky to have built a relationship with such a dedicated educator. Prof. Loomba has provided encouragement and support during difficulties of the research, at times he displayed more confidence in my skills than I did. He pushed me when I needed to be pushed, and managed to find time for my education when he was already balancing a research group, a family, and a teaching position. I am deeply thankful for his time, empathy, and mentorship.

Contents

Abstract i

Acknowledgements iii

List of Figures vi

1 Introduction, Motivation, and N. Phan's Detector 1
- 1.1 A History of Dark Matter . . . 1
- 1.2 Directionality of Nuclear Recoils - A Signature for Dark Matter . . 3
- 1.3 Nguyen Phan's Nuclear Recoil Detector . . . 5

2 Univariate and Multivariate Statistical Moments 8
- 2.1 Univariate Statistical Moments . . . 8
  - 2.1.1 Definitions . . . 8
  - 2.1.2 Univariate Moment Properties . . . 12
- 2.2 Multivariate Statistical Moments . . . 14
  - 2.2.1 Two-Dimensional Second Order Moments . . . 15
  - 2.2.2 Two-Dimensional Third Order Moments . . . 16

3 Extracting Properties of Nuclear Recoil Tracks 19
- 3.1 The Algorithms For Identifying Track Direction . . . 20
  - 3.1.1 The Current Algorithm . . . 20
  - 3.1.2 New Algorithm . . . 21
- 3.2 Straight Simulated Nuclear Recoils . . . 24
  - 3.2.1 Drawing Straight Tracks . . . 24
  - 3.2.2 Applying the New Algorithm . . . 26
  - 3.2.3 Trials of Algorithms on Straight Tracks . . . 28
- 3.3 Nuclear Recoil Tracks Simulated with SRIM . . . 31
  - 3.3.1 Trials of Algorithms on SRIM Generated Tracks . . . 33
- 3.4 Methods for Excluding Poor Quality Events . . . 34

4 Application of the New Algorithm to Real Data 38
- 4.1 Application of the New Algorithm to Events in the Nuclear Recoil Band . . . 39

4.2 What's Next? 42

A Mathematics for Drawing Straight Tracks **44**

B Matlab and Python Codes **47**
B.1 The Algorithms 47
B.1.1 Current Algorithm 47
B.1.2 New Algorithm 49
B.2 Drawing Nuclear Recoil Tracks 52
B.2.1 Drawing Straight Tracks 52
B.2.2 Tracks from SRIM Data 56
B.2.3 Creating Track Images from Nuclear Recoil Tracks 60

Bibliography **62**

List of Figures

1.1 Solar system Motion . . . 4
1.2 Earth Based WIMP Detector . . . 4
1.3 Graphical detector cross section . . . 6
1.4 Detector Photograph . . . 7

2.1 Example Plot . . . 10
2.2 Normal Distribution Example . . . 13
2.3 Negative Skewness . . . 13
2.4 Positive Skewness . . . 13
2.5 Fitted ellipse example . . . 16
2.6 Two dimensional example distribution with intensity along the third axis. . . . 17
2.7 Image projected into $x - z$ plane. . . . 17
2.8 Image projected into the $y - z$ plane. . . . 17

3.1 Short nuclear recoil track . . . 20
3.2 Long nuclear recoil track . . . 20
3.3 CCD camera image from detector . . . 22
3.4 CCD camera image after application of Source Extractor . . . 22
3.5 New Algorithm Example 1 . . . 23
3.6 New Algorithm Example 2 . . . 23
3.7 Straight Track . . . 25
3.8 Skewness Vector for straight track . . . 26
3.9 Measured angles from skewness vector . . . 27
3.10 Example track after simulation of image properties . . . 28
3.11 Fraction Correct as a function of front fractional energy, holding length constant. . . . 29
3.12 Fraction Correct as a function of front fractional energy, holding length constant. . . . 29
3.13 Fraction correct as a function of track length, holding front fractional energy constant. . . . 30
3.14 Fraction correct as a function of track length, holding front fractional energy constant. . . . 30
3.15 Skewness magnitude distribution . . . 30
3.16 SRIM simulated nuclear recoil track . . . 32
3.17 The same track after convolution, rebinning and noise. . . . 32

3.18 Correct defined by applying the algorithms before convolution. . . . 34
3.19 Correct defined by the initial angle specified in SRIM. 34
3.20 Skewness Distribution from SRIM 35
3.21 A scatterplot of each event's energy-length ordered pair. 36
3.22 The same plot divided into thirds 36
3.23 Top: The fraction of events correct at a given energy with three cuts excluding events below the listed skewness magnitude. Bottom: The fraction of events included at each point in the top plot. (Saxophone Plot) 36
3.24 The fraction correct as a function of which third of the length vs. energy plot is being inspected. Each third only includes 33 percent of the events by definition. 36

4.1 Scatter plot of track length versus energy for data taken from the detector. 39
4.2 The nuclear recoil tracks lie between the red and green lines. 39
4.3 Red events have a skewness magnitude above 0.9. 40
4.4 Angular distribution for all events in the nuclear recoil band. 40
4.5 Skew Magnitude distribution for the nuclear recoil band of real data between 0 and 200 keVee. 41
4.6 Top plot: Fraction of events with correctly identified directions above several different skewness magnitudes. Bottom plot: the fraction of included events at each energy. 42
4.7 Fraction of events with correctly identified directions in several cuts of the length versus energy scatter plot. 42
4.8 Angular distribution for events in the top third of the nuclear recoil band. 43
4.9 Angular distribution for events in the top two thirds of the nuclear recoil band. 43
4.10 Angular Distribution for a skew magnitude cut at 0.1. 43
4.11 Angular distribution for skew magnitude cut at 0.1 in the top third. 43

A.1 Looking at a straight track, on its side, in continuous space. The total energy is E1+E2, and is represented by the total area in the trapezoid. 45
A.2 Looking down on the same track in the x-y plane. The track makes an angle θ with the x-axis. 45

Chapter 1

Introduction, Motivation, and N. Phan's Detector

In this thesis we will be discussing techniques for measuring the direction of nuclear recoil tracks in a low pressure gas detector imaged with a low noise, high sensitivity CCD (charge coupled device). The CCD camera used in this detector is of scientific grade, described in section 1.3.

We will analyze tracks created by recoiling atoms in the gas detector to determine their initial directions. Finding methods for determining the directions is of great interest to directional dark matter searches.

1.1 A History of Dark Matter

Over the course of the last century, astronomers investigating galaxies have determined that they seem to have far more matter than contained in stars and gas. Normal (baryonic) matter reflects and produces light by interacting with the electromagnetic force. Matter also interacts with the gravitational force; all matter attracts all other matter. When astronomers look at other galaxies, they can observe the matter which interacts with light, and they can observe how gravity is interacting with the matter that they can see. In the observations, this has led to a discrepancy, as there appears to be a larger gravitational effect than can be explained by the amount of normal matter that can be observed. This was first discovered by F. Zwicky in 1933. Zwicky used the virial theorem to calculate

the approximate average velocities of matter in the Coma cluster. His calculation based on the amount of luminous (normal) matter resulted in an average velocity of 80 km/s. The actual measurements of the cluster resulted in velocities over 1000 km/s. His calculation indicated that if the galaxies were virialized, there needed to be a matter density 400 times greater than what was observed in order to explain the measured velocities [1].

The astronomical community has since drastically improved the ability to make precise measurements of phenomena across the universe. These measurements allow astronomers to explore key time frames in the early stages of the universe. We know today that in the early universe, in a period referred to as the Epoch of Recombination, the ordinary matter went through a phase transition consisting of a plasma of protons, electrons and photons, to neutral hydrogen and photons. After this epoch, when the universe was $\approx$ 300,000 years old, the hydrogen was largely transparent to most wavelengths of light, and, because of this, existing photons ceased interacting with the atoms. The universe continued to undergo both cooling and expansion, and these photons became redshifted and cooled to 2.7 Kelvin and are now observed as the Cosmic Microwave Background (CMB). The Planck Collaboration has conducted an extensive study of the CMB with the five year Planck satellite mission and is able to provide measurements of many cosmic parameters. The collaboration reports the current matter in the universe is approximately 84% dark matter, while normal matter is the remaining 16% [2].

Furthermore, the Planck measurements are in strong agreement with other independent methods, such as gravitational lensing, and measurements of galactic rotation curves, suggesting cold, massive, collisionless dark matter. The foremost hypothesis is that dark matter consists of a new fundamental particle created in some early state of the universe's development [3].

This type of matter does not interact electromagnetically, and evidence from gravitational lensing and measuring the speeds of rotating galaxies indicates that the dark matter is distributed spherically throughout the galaxy [4]. Given its distribution on the galactic scale, and if it is a new fundamental particle, its features may resemble a heavy neutrino.

There is strong theoretical motivation for believing that the mysterious dark matter interacts through the weak force as well as gravitationally. One of the leading candidates is the Weakly Interacting Massive Particle, called a WIMP. If this

theorized form of dark matter does interact through the weak force, then it can couple with baryonic matter, which implies we may be able to observe its interaction. Many collaborations across the world are working to directly detect the dark matter particles that pass through the solar system. There are Earth-based laboratories with specially designed detectors searching for the rare interactions between these particles and normal matter. Much of the work being done relies on constraints on the nature of the dark matter obtained from astronomical and particle physics.

Current estimates of the properties of the distributed dark matter are based on simulations and kinematics of stars in our galaxy. These indicate that the dark matter particles can be described as a non-interacting gas which has a velocity distribution which is approximately Maxwellian. It is theorized to have an RMS velocity of $\approx$ 230 km/s, which cuts off at an escape velocity of $\approx$ 600 km/s. Furthermore, it has a density (ρ) in the range $0.3 \leq \rho \leq 0.7$ GeV/c$^2\cdot$ cm^3 [5].

1.2 Directionality of Nuclear Recoils - A Signature for Dark Matter

As we have described above, our galaxy is expected to have dark matter distributed approximately spherically throughout a large volume much larger than the disk containing the stars and gas. This dark matter is not expected to be co-rotating with the galactic disk of stars and gas, and because of this the baryonic matter in the disk would see a "wind" of dark matter in the local rest frame. For our solar system, the Sun's velocity ($\approx$ 230 km/s [5]) is directed toward the constellation Cygnus, and the wind appears to emanate from there (figure 1.1). This wind results in two different expected signatures on Earth, but here we only describe the one which is the premise of this work.

The direction of Cygnus circles across the northern sky over a 24 hour period due to Earth's rotation. The WIMPs are expected to produce elastic nuclear recoils in the gas volume, and the direction of these recoils is further expected to be correlated with the direction of the WIMP wind relative to the Earth. This anisotropy in the lab frame would appear to change direction as the Earth rotates about its axis, resulting in a sidereal modulation of the signal. This is illustrated in figure 1.2. The background events (lab-based or those of solar origin) which can

cause nuclear recoils are expected to have very different angular distributions, none of which should be directed from Cygnus. Therefore, this is why this directional signature, if its ever seen, is considered "smoking gun" evidence for dark matter.

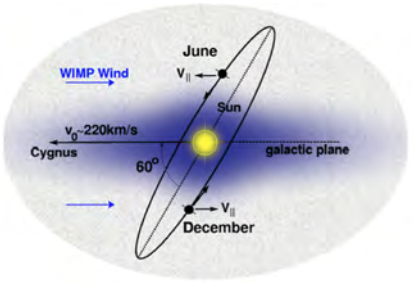

FIGURE 1.1: The Sun's motion and the Earth's plane of rotation. Image obtained from [6]. (The velocity measure here is out of date. The most recent measurements suggest 230 km/s [5])

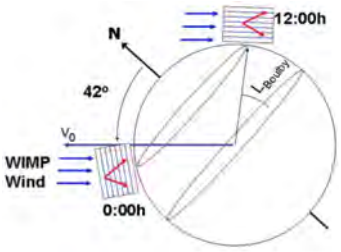

FIGURE 1.2: The observed direction of WIMP induced recoils (in red) on Earth at two times of day. Image obtained from [6].

The challenges for seeing this signature are great, however. We know that dark matter interacts very rarely, conservatively on the order of an event per year per kilogram [7].

In addition, it is extremely challenging to measure the direction of the nuclear recoils produced by a WIMP. For a given track in the detector, there are two components of the track, its axial direction and its skewness. Determining the axial direction of the event is a challenge, depending on the event's energy and orientation in the detector. However, even the capability to specify the axis still leaves an ambiguity as to the direction, was it traveling "forward" or "backward" along the axis? There has been a large amount of work investigating how many events will be needed to rule out an isotropic distribution of nuclear recoil events. If all we have is the axial direction of events, it takes on the order of a few hundred events to rule out the isotropy [3]. However, if we can specify the head and tail of the event and break the ambiguity on the axial direction, we can reduce the number of events needed by an order of magnitude [8]. Because of the very low event rate expected for dark matter, it is to the greatest advantage to reduce the number of events needed to confidently see the anisotropy [3].

1.3 Nguyen Phan's Nuclear Recoil Detector

University of New Mexico PhD candidate Nguyen Phan has developed a low pressure time projection chamber (TPC) operated with 100 Torr carbon tetrafluoride (CF_4) gas and read-out with a CCD camera. The detector consists of a cylindrical aluminum vacuum vessel which is 16 cm in height and 29 cm in diameter, photographed in figure 1.4.

In figure 1.3, we show a graphically reproduced representation of the interior of the vessel which houses an anode wire grid, a cathode mesh, and three CERN GEMs (gas electron multipliers). The GEMs are 7 cm x 7 cm and are made from a 50 μm kapton foil that is cladded on both sides by copper and perforated with 50 μm holes on a 140 μm pitch. Outside the vessel, a mount holds a FLI back-illuminated CCD camera (13 μm pixels at 1024 x 1024 pixels) and 50 mm F1.2 Nikon lens looking down onto the vessel and focused on the surface of GEM 3.

Energetic particles in the gas produce ionization along the length of their track, freeing electrons from the molecules in the gas. These are then caused by the electric field to drift toward the anode and pass through the holes in the GEM. Inside the holes, the electrons are accelerated by a much stronger electric field, and reach an energy high enough to ionize the neutral gas, releasing secondary electrons. These additional electrons drift into GEMs 2 and 3 which facilitate further amplification of the ionization. With these three stages of GEMs, a gas gain of over 10^5 is achieved. In other words, for each electron which enters GEM 1, 10^5 electrons exit GEM 3. In addition to creating electrons in the GEM holes, scintillation light is also produced. This light is imaged by the CCD camera and lens outside the vacuum vessel and provides the two dimensional image of the track.

We measure energy by the intensity of the images produced by the detector. This energy is only a result from the electrons which were liberated by the recoiling atomic nuclei. The CCD camera captures intensity in units of ADU's (analog to digital units). We perform a calibration of the camera images by producing ionization events of known energy. The energy deposited by the event into ionization is measured in keVee (keV "electron-equivalent"). This keVee unit represents the required energy of an electronic recoil event to produce an identical amount of observed ionization. However, this ionization energy does not represent the full energy which is deposited into the gas by the recoiling atomic nucleus. This

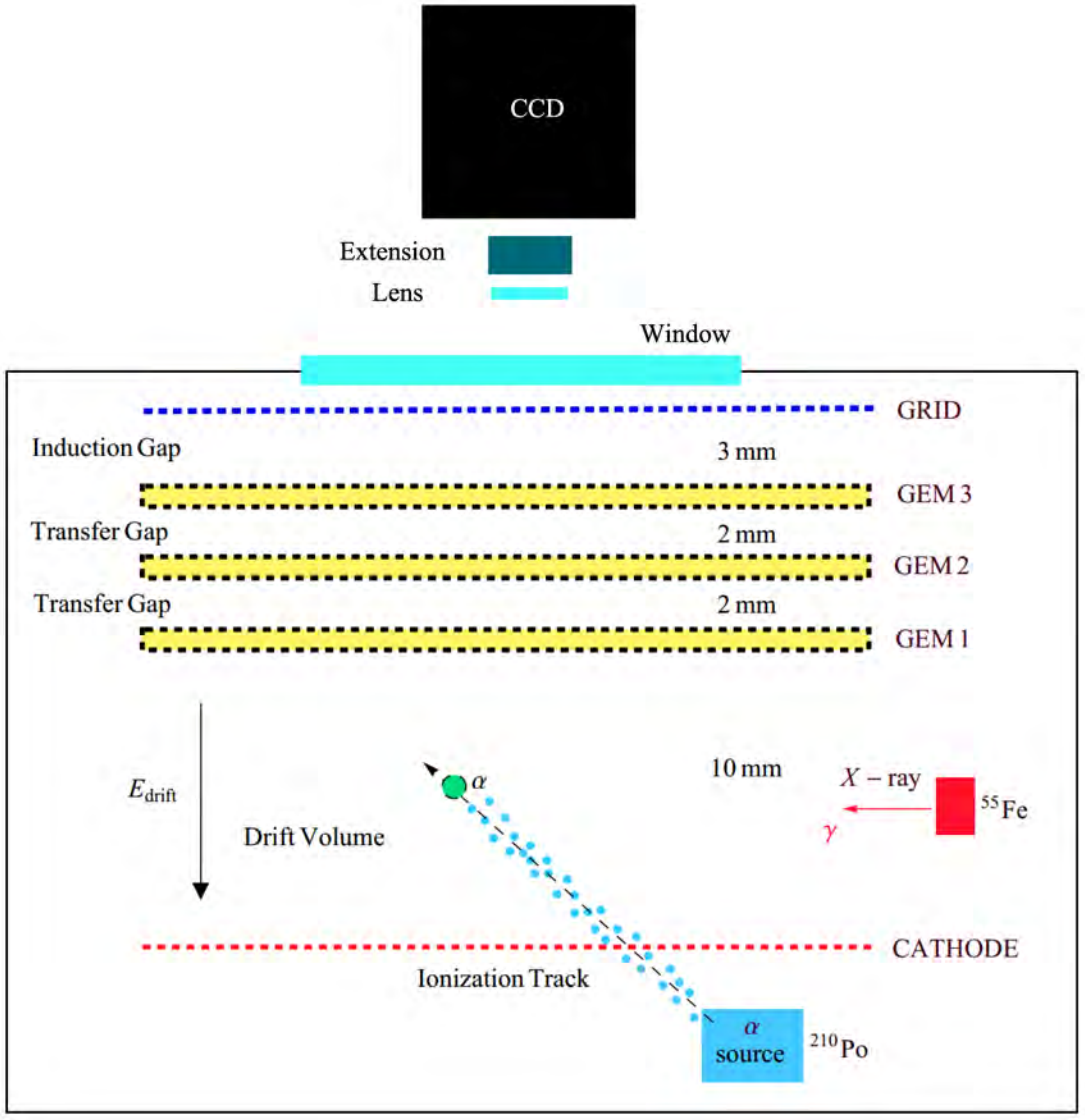

FIGURE 1.3: A cross sectional view of the internal working of the detector (not to scale) [9].

nucleus, unlike electrons, also deposits energy in the form of heat and phonons. The full recoil event energy is described in units of keVr (this specifies the initial nuclear recoil kinetic energy and distinguishes this energy from keVee), and we can convert from keVee to keVr through the use of a quenching factor, which is given by theoretical calculation [10].

The following chapters in this thesis are devoted to finding a method of assigning head-tail directionality to the nuclear recoil events captured by this detector. In doing so, we hope to provide a method for requiring minimal events in order to ultimately rule out isotropy in the search for galactic dark matter.

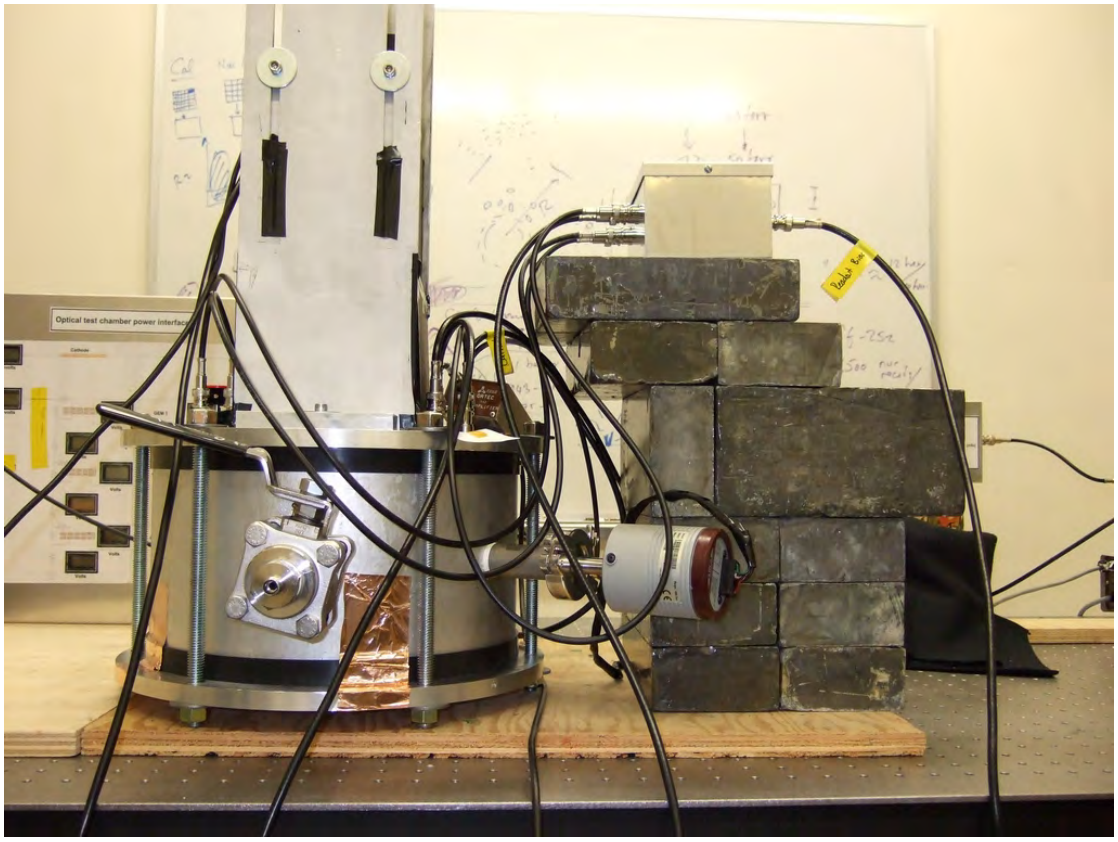

FIGURE 1.4: A photo of the detector [9].

Chapter 2

Univariate and Multivariate Statistical Moments

In the majority of the images taken by the detector, it is very difficult to determine by eye the direction of an individual nuclear recoil track. In order to extract the direction we will utilize a group of expressions, called moments, from statistics. These moments allow us to determine the mathematical characteristics of a set of data. We will investigate what we can learn from these moments and how we might apply them to our tracks.

2.1 Univariate Statistical Moments

The following discussion utilizes the moment definitions described in the paper "Visual Pattern Recognition by Moment Invariants" by M.K. Hu [11].

2.1.1 Definitions

Moments are used in statistics and physics to characterize the properties of a given mathematical object. Each moment is specified by its "order." For any number of dimensions, the zeroth moment is simply the total sum of the distribution. For some set of numbers, it is just the sum of all the numbers in the set. If the set describes, for example, the distribution of the mass of an object, then the zeroth order moment is the total mass of the object.

The first order moment is also easy to imagine and familiar. It is the average value of some set of numbers. In one dimension, assuming all the values have equal weight (we will return to weighting soon) we can find the first moment, also known as the mean, by summing all the values in our set and then dividing by the total number of objects we added together. For example, the zeroth moment of the set consisting of $S = \{1, 2, 3, 4, 5, 6\}$ is 21, and the first moment is 3.5.

We can also use the mean to find an average location of a set of numbers. Imagine that we have a set of numbers, $F = \{10, 11, 12, 13, 14\}$. We can assign an index to each member of the set. We can say that $F_1 = 10$ or that $F_4 = 13$. Now imagine that we summed over all of the values and then divided by the total number of values that we summed over. We would get $\frac{10+11+12+13+14}{5} = 12$. That is the average value of the elements of the set. Suppose instead that we wanted to know the average value of the index. Consider the figure 2.1. We can see that if that picture were actually a picture of a piece of wood, then it would be heavier on one side than on the other. This means that one side is more weighted than the other. We can imagine trying to balance it on a point, and the point would need to be more to the right than the lengthwise center of the plot. We can find where that location is by examining the weighted first moment with respect to the indices. Instead of summing over just the set values, we will sum over the product of the index and the value of the set element at that index. This utilizes the set values themselves as weights, to judge how "strongly" a given index contributes to the sum. We will then normalize the resulting sum by the total "weight" of the set, or the sum of just the elements. Earlier we called this sum the zeroth moment. Using conventional mathematical notation, and calling the "n$^{\text{th}}$" order moment M_n we have for the first two moments in our example:

$$\begin{aligned} M_0 &= \Sigma_{i=1}^{5} F_i = 50 \\ M_1 &= \frac{1}{M_0} \Sigma_{i=1}^{5} i \cdot F_i = 3.167 \end{aligned} \tag{2.1}$$

What we have found with M_1 in 2.1 is that the weighted average index location of the data is at location 3.167. This can be visually verified by inspecting the image, and noticing that the point which splits the area in half is roughly between 3 and 4 by eye. What we have done is chosen the sequence index as our variable of interest and found its weighted average, using the set element at each index as a respective weight.

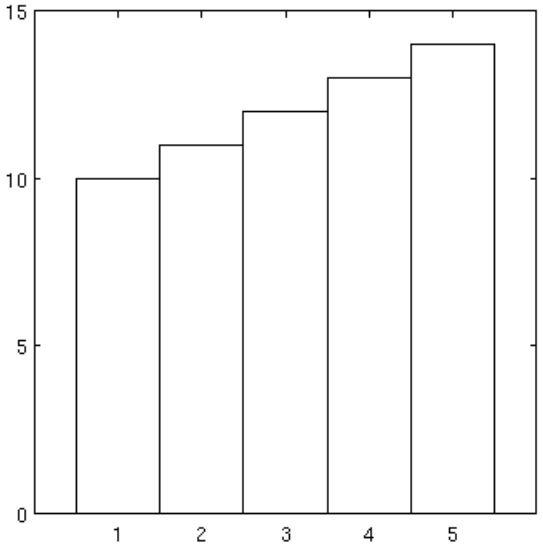

FIGURE 2.1: Plot of the example set F, the x-axis is the index of the set item. The y-axis is the set item itself.

More generally, if we have some density function ρ, which describes the intensity of some scalar value as a function of x we can find the first two moments as

$$M_0 = \int_x \rho(x)dx \tag{2.2}$$

$$M_1 = \frac{1}{M_0}\int_x x\rho(x)dx \tag{2.3}$$

Discretely this becomes:

$$M_0 = \Sigma_i \rho(x_i) \tag{2.4}$$

$$M_1 = \frac{1}{M_0}\Sigma_i x_i \rho(x_i) \tag{2.5}$$

In physics, ρ is commonly the mass density of some object. Imagine we had a straight bar with some length and a mass density $\rho(x)$. we would be able to find its total mass by simply integrating $\rho(x)$ over the length of the rod. We could find the center of mass of the rod by finding M_1, which represents the x-coordinate of the center of mass.

In this paper, we will denote the mean value of a spatial variable (the first moment) with an over-bar. The average x-value will be denoted $\overline{x}$, average y-value as $\overline{y}$.

The second moment, M_2 by itself does not have a clearly intuitive meaning like the zeroth and first moments. We will not be using the second moment in its raw form, however, so we will simply present it mathematically here and discuss a different version of its presentation momentarily.

$$M_2 = \frac{1}{M_0} \int_x x^2 \rho(x) dx \tag{2.6}$$

Or discretely:

$$M_2 = \frac{1}{M_0} \Sigma_i x_i^2 \rho(x_i) \tag{2.7}$$

In the above equations we are normalizing the non-zero moments with respect to the zero moment M_0. This is a normalization of convenience, as it allows us to meaningfully compare the moments of different distributions. It is important to note that not all authors utilize this normalization in their definitions of the moments.

Finally, the third statistical moment is simply an extension of this higher powers process, but again does not have a simple intuitive explanation. We will also not be using this form of the third moment, but it is presented for completeness.

$$M_3 = \frac{1}{M_0} \int_x x^3 \rho(x) dx \tag{2.8}$$

Or discretely:

$$M_3 = \frac{1}{M_0} \Sigma_i x_i^3 \rho(x_i) \tag{2.9}$$

When we inspect the expressions given in equations 2.1 - 2.9 it is apparent that the values are dependent on our choice of coordinate system. However, these same moments can be given a much stronger meaning if we center our coordinate system on the mean value of the coordinate. We will therefore define a new set of moments and discuss their meanings. Moments calculated relative to a coordinate system centered on the mean value of the coordinate are called central moments, and will be denoted μ_n where "n" is the order of the moment. The mean of the coordinate is given by the first moment, so we have that $\overline{x} = M_1$.

$$\mu_2 = \frac{1}{M_0}\int_x (x-\overline{x})^2 \rho(x) dx \tag{2.10}$$

$$\mu_3 = \frac{1}{M_0}\int_x (x-\overline{x})^3 \rho(x) dx \tag{2.11}$$

And it is straightfoward to take the discrete case by taking the integral into a sum and applying the appropriate summation indices to the x variable. Note that the $\overline{x}$ is a constant, and does not get summed over.

Even more generally we have:

$$\mu_{n\geq 2} = \frac{1}{M_0}\int_x (x-\overline{x})^n \rho(x) dx \tag{2.12}$$

Now that we have defined our single dimensional moments, we may inspect their properties.

2.1.2 Univariate Moment Properties

Equation 2.12 provides us with moments equal to or higher than two. It is well known in the community that these central moments are translationally invariant [11].

The second order central moment (2.11) characterizes the width of the distribution about its mean, and is called the variance of the distribution. The canonical example of the variance is the width of a Gaussian or Normal distribution. The

square root of the variance is also very important in mathematics and physics. We denote the variance as σ_x^2 and its positive square root is called the *standard deviation* of the distribution. The standard deviation is denoted by σ_x. In a Normal distribution, approximately 68% of the total distribution lies within one standard deviation of the mean value. This example is illustrated in figure 2.2.

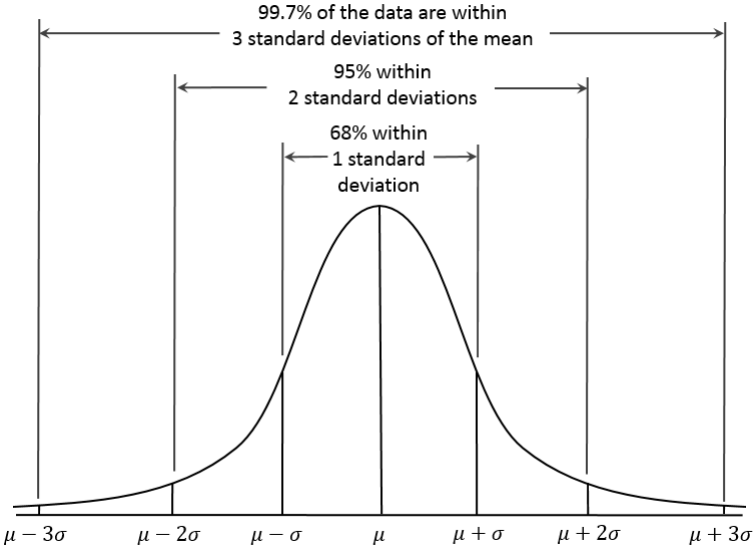

FIGURE 2.2: An example Normal distribution. Note, μ here denotes the mean of the set, and not the central moment. The size of σ characterizes the width of the distribution. Image obtained from [12].

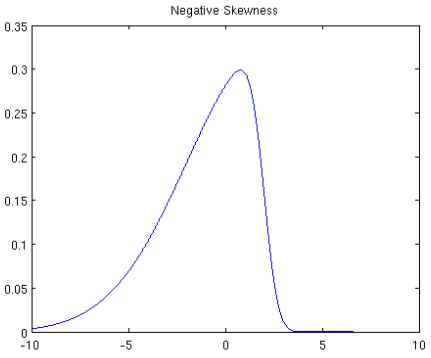

FIGURE 2.3: Negative Skewness

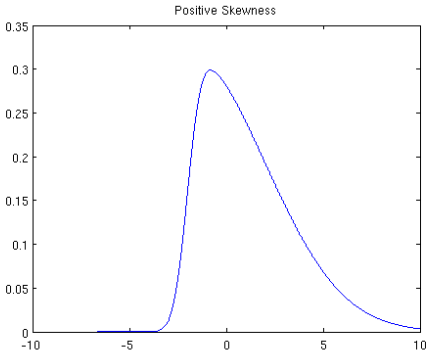

FIGURE 2.4: Positive Skewness

The third central moment (2.11) gives a measure of how asymmetric the distribution is about its mean, and is referred to as the skewness. In one dimension, if a distribution has a positive skewness, then it has a longer tail in the increasing direction of the variable. If its skewness is negative, then the longer tail is in the decreasing direction of the variable, relative to the mean (the first moment). See figures 2.3 and 2.4 for examples of negative and positive skewness. It is important

to note that if the skewness is identically zero this does not imply that the distribution is symmetric. However, if the skewness is non-zero then the distribution is certainly asymmetric.

We now have four ways of characterizing some one dimensional set of data. We can describe the total size of the data by simply summing all of its parts; this gives the zeroth moment. We can describe the center-of-mass of the data by summing up all of the indices weighted by the data values themselves, and then dividing by the total sum of the data, which is known as the mean or the first moment. We can also find how extended the distribution is by inspecting the magnitude of the square root of the second moment, and we can determine which side of a distribution is more extended by looking at the sign on the third moment. Using these ideas, we will extend the moments into two dimensions, and briefly discuss higher dimensional generalizations.

2.2 Multivariate Statistical Moments

For this project, we are interested in inspecting the moments of images. Each image contains a set of pixels and each pixel then has some corresponding intensity. This intensity represents the value of the data point in the set, and the x-y coordinates represent the data point's location within the set.

In two dimensions, the zeroth and first order moments become:

$$M_{00} = \int_y \int_x \rho(x, y) dx dy \tag{2.13}$$

$$M_{10} = \frac{1}{M_{00}} \int_y \int_x x\rho(x, y) dx dy \tag{2.14}$$

$$M_{01} = \frac{1}{M_{00}} \int_y \int_x y\rho(x, y) dx dy \tag{2.15}$$

For orders greater than one, we will use the centralized moments. These moments are now calculated relative to a coordinate system centered on the mean x value ($\overline{x}$) and the mean y value ($\overline{y}$). It is conventional to refer to the order of the moment as being order $p + q$.

$$\mu_{pq} = \frac{1}{M_{00}} \int_y \int_x (x - \overline{x})^p (y - \overline{y})^q \rho(x, y) dx dy \tag{2.16}$$

2.2.1 Two-Dimensional Second Order Moments

Given that $p + q$ characterizes the order, and we can see if p is 2 and q is 0 then we simply get the variance in the x variable. Similarly, if p is 0 and q is 2, then we will get the variance in the y variable. If both p and q are 1, however, we get an unfamiliar second order moment. This new quantity is referred to as the *covariance* between x and y. We will label this covariance σ_{11}^2. This value is a measure of how much the data points along one axis change as we move along the other axis. Namely, it is a measure of how the two variables are correlated with one another.

Using these three variances, we can build a matrix referred to as the covariance matrix of the distribution, below called D. It has the following form:

$$Cov(D) = \begin{pmatrix} \sigma_{20}^2 & \sigma_{11}^2 \\ \sigma_{11}^2 & \sigma_{02}^2 \end{pmatrix} \tag{2.17}$$

This is a symmetric matrix, and it can be diagonalized. In doing so, we compute the eigenvectors and eigenvalues of the matrix. The eigenvectors characterize unit vectors along a new orthogonal set of axes. These new axes represent a new set of basis vectors in which there is no covariance between the variables. In other words, if we transformed from our coordinate system to this new coordinate system given by the eigenvectors of the covariance matrix, then we computed the covariance matrix again in this new basis, we would find that all of the off-diagonal terms would be zero.

Furthermore, the eigenvalues of the covariance matrix correspond to the variances along the new axes specified by the eigenvectors. Because the variance characterizes the width of the distribution through its square root, by diagonalizing the matrix, we can characterize the geometric properties of the distribution. In two dimensions, the eigenvalues of the covariance matrix are related to the physical scale of the distribution in the directions of the corresponding eigenvectors. For the sets of data we are interested in, we will be fitting an ellipse to the data. In

two dimensions we will characterize the semi-major axis with the larger of the two eigenvalues, and the semi-minor axis of the ellipse with the smaller eigenvalue.

We will call these eigenvalues λ_1, λ_2 and we will choose to name them such that $\lambda_2 > \lambda_1$. Next, we will consider the quantity: $\sqrt{1 - \frac{\lambda_1}{\lambda_2}}$. For any set of data, we can fit an ellipse to the data. The quantity we are considering specifies the eccentricity of the ellipse. This uniquely determines the shape of the ellipse regardless of the scaling of its semi-major and semi-minor axes.

Next we note that $\lambda_{1,2}$ represents a variance along the direction of the corresponding eigenvector, we can choose a scaling which captures a desired fraction of the distribution.

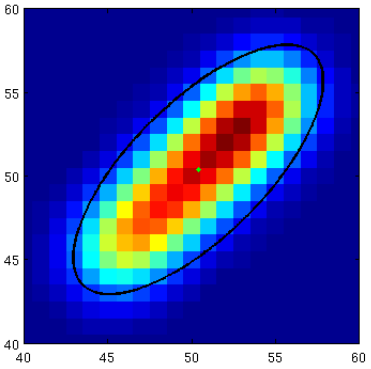

FIGURE 2.5: An example distribution with a fitted ellipse. The scaling is to 2σ on both the semi-minor and semi-major axes. The green dot is the weighted mean position. Red pixels are high intensity, blue is lower intensity.

2.2.2 Two-Dimensional Third Order Moments

There are four third order moments for a two dimensional distribution. They are μ_{30}, μ_{21}, μ_{12}, and μ_{03}. Though there are four, we will only be looking at μ_{30} and μ_{03} in this treatment.

Recall that in one dimension the third moment told us if there was an asymmetry in the shape of the distribution about its mean. We can calculate the third moment relative to the x-axis in a two dimensional distribution, and it will tell us the same information about the distribution. We will now inspect figure 2.5. The data set is two dimensional, but each pixel contains some intensity value. We use this intensity value to map our image into three dimensions, and we can inspect the image from multiple viewpoints in all three dimensions.

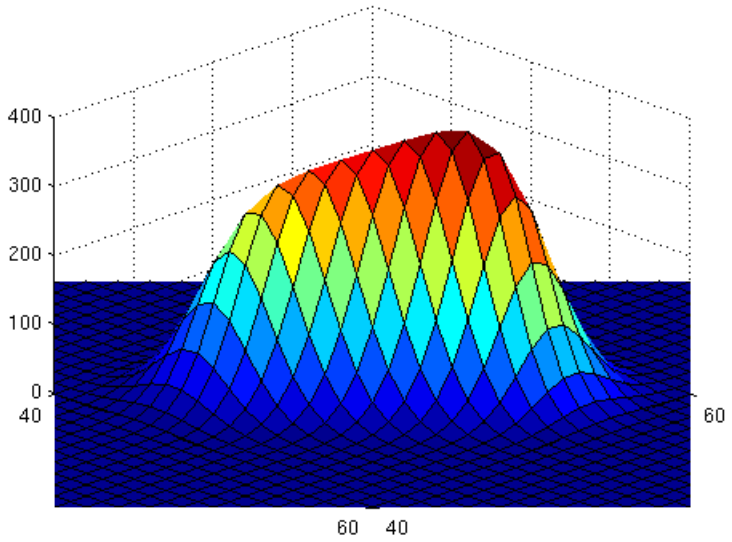

FIGURE 2.6: A three dimensional look at the example distribution. This view is along the semi-minor axis of the ellipse.

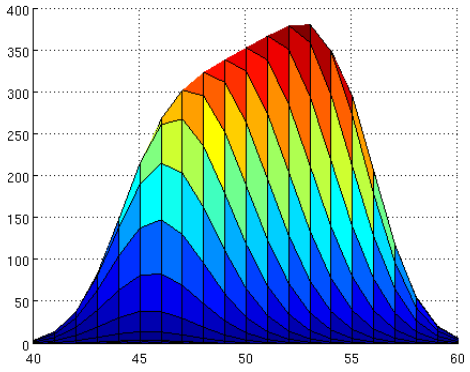

FIGURE 2.7: Image projected into $x - z$ plane.

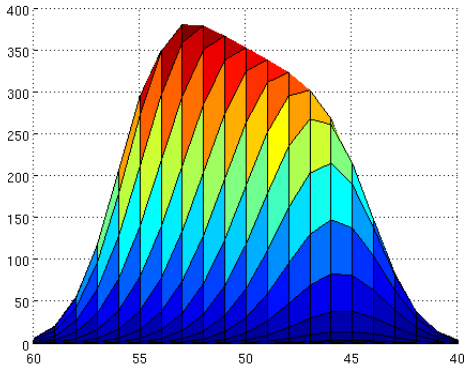

FIGURE 2.8: Image projected into the $y - z$ plane.

We can see that there is an asymmetry in the distribution, as one section of it is notably higher than another section. We can imagine rotating the image to look down the positive y-axis, so that we have the positive x-axis to the right, and the z-axis straight up. From this point of view (2.7) we can see that the image is negatively skewed, the tail tapers to the negative x direction. We can perform the same rotation and look down the positive x-axis, so that positive y is to our left, and z is straight up. This is shown in figure 2.8, and we can see that it is also negatively skewed, the tail is in the direction of negative y. Because of this, if we calculate the moments μ_{30} and μ_{03} we expect them to both be negative.

These are all the moments we will require for our treatment of the data collected by Nguyen Phan's detector.

Chapter 3

Extracting Properties of Nuclear Recoil Tracks

As discussed in Chapter 1, the TPC imaging detector contains low pressure carbon tetrafluoride (CF_4) gas. When neutrons or WIMPs interact with the atoms in our gas, the atoms can be freed from the molecular structure, shedding their orbital electrons, and recoiling as bare nuclei. This nuclei then recoils in some direction specified by the kinematics of the collision. The nuclei will then lose energy as it travels to heating up the gas, causing scintillation in the gas, ionizing other atoms, and knocking other fluorine or carbon nuclei out of the CF_4 structure. If the recoiling atomic nuclei ionize other atoms, this frees electrons from the atoms and those electrons will drift through the electric field. They will come to the GEMs and cause a cascade of electrons, which then produce light which is captured by the CCD.

As the recoiling nuclei travels, we expect it to lose energy along its track, and in the image this is manifested as a declining intensity along the track. This produces a skewness along the direction of the track as the intensity lowers with length. If the energy is quite large and the length of the track is sufficiently long, it is obvious to see this drop in intensity with length, and we present an example in figure 3.2. We refer to the direction of the track as being the direction along which the pixel intensity is falling. In these images, red pixels correspond to high intensity, and blue pixels correspond to low intensity.

Nguyen's detector produces images whose pixels are 160 μm on a side. His detector has a point spread function which is approximately circularly Gaussian with a

width of 350 μm. It has a gain of approximately 250 ADU/keVee. Finally we can determine the noise in the images from the detector. Knowing these parameters allows us to construct images which closely mimic the properties of the images taken from the actual detector [9]. The CCD device used with the detector produces *.fits images. This is an image format type very common to CCD imaging units, and is used extensively throughout astrophysics.

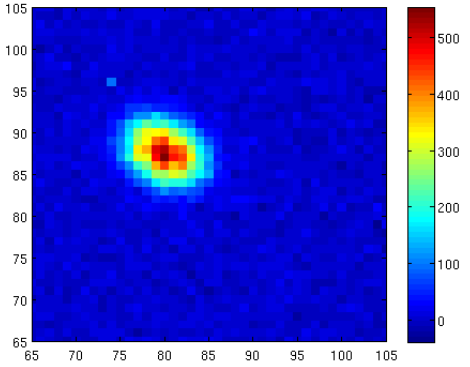

FIGURE 3.1: Image of a nuclear recoil track taken from the detector. Pixels in ADUs.

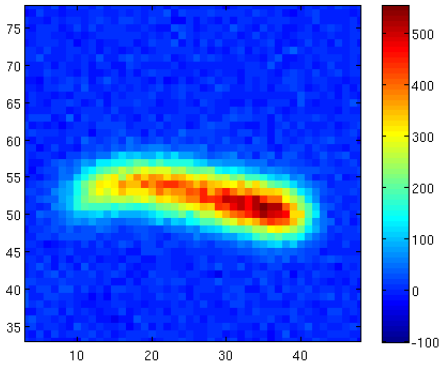

FIGURE 3.2: A nuclear recoil track in which it is easy to see the direction. Pixels in ADUs. The track's direction is from right to left.

3.1 The Algorithms For Identifying Track Direction

3.1.1 The Current Algorithm

In order to determine the direction of more ambiguous tracks like figure 3.1, we currently use the third central moment in one dimension. First we isolate the track on a zero background and calculate the covariance matrix of the track. We then diagonalize it and find the semi-major axis of the ellipse which fits the track. However, there is an ambiguity in the direction of the track, as the second moment only gives us information about the physical extent of the track, not its asymmetric features like the slope or direction of the track.

In order to find the direction of the track, we choose the vector along the semi-major axis of the ellipse that the diagonalization gave us. We then use a method

called the Radon Transform to project the track onto that vector direction of the semi-major axis of the ellipse. We can then look at this new distribution which contains the projected information of the track and calculate its skewness. If the skewness is positive, then we claim that the vector we used in the Radon Transform was the correct vector, otherwise we say that the vector needs to be flipped 180 degrees.

3.1.2 New Algorithm

We began by introducing the tool Source Extractor [13] to the analysis of the nuclear recoil tracks. It is a widely used software package in the astronomical community. It is capable of opening a fits image (this is short for a *.fits file type), finding objects in the image, removing the background, and then creating a new image with only the objects of interest and background removed. It also is capable of outputting a file with information regarding the objects located within the images.

On the images, we used Source Extractor to return the tracks in the images on a flat zero background. Source Extractor's output catalog was also configured to specify the $x-y$ coordinates of the corners of the box in the image containing the object.

Source Extractor can be told to identify images by two sets of parameters. The user can specify a threshold of how high above the standard deviation of the image pixel distribution an individual pixel needs to be in order to be considered part of an object. We can also specify how many pixels need to be above the threshold and contiguous in order to constitute an object.

In our following analysis we will be applying Source Extractor to the images, and then applying both the new algorithm and the original algorithm to the resulting tracks extracted by Source Extractor. This will result in the results obtained from the original algorithm being hybridized between N. Phan's original work, and the new skewness calculation presented below. We concluded the introduction of Source Extractor to the original algorithm allowed a stronger comparison of the moment calculations between the two algorithms without obscuring the differences with the addition of this new tool.

Source Extractor also provides basic noise removal and deblending of objects. We used the default parameters for both of these functions in the software.

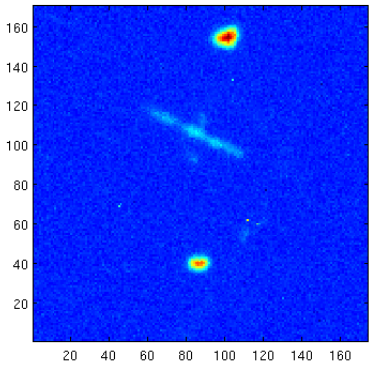

FIGURE 3.3: CCD image taken from detector before application of Source Extractor.

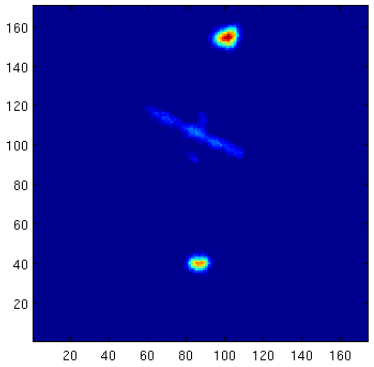

FIGURE 3.4: The extracted objects/tracks on a zero background after the application of Source Extractor.

Next, we take the tracks from the image that Source Extractor output and calculate both the first four moments in x and y, and the covariance between x and y. We next find the eigenvectors of the covariance matrix, and select the one corresponding to the semi-major axis. We will call this vector a candidate direction. Note that this vector is produced by MATLAB's eigenvalue and eigenvector decomposition of the covariance matrix.

Once we have this candidate direction, we then create a new, two dimensional quantity referred to as the skewness vector:

$$\vec{S} = \frac{[\mu_{30}, \mu_{03}]}{\lambda_2^{\frac{3}{2}}}. \tag{3.1}$$

Where $[\mu_{30}, \mu_{03}]$ are the skewness in x and the skewness in y respectively, and λ_2 is the value of the largest eigenvalue of the covariance matrix. We then normalize the skewness vector to have a magnitude of unity, and take its vector inner product with the candidate direction vector. If the angle calculated from this inner product is greater than 90 degrees, then we take as the true direction of the track the negative of the candidate direction, flipping it 180 degrees. If the angle is less than 90 degrees, then we say the candidate direction is the correct direction. This is demonstrated in figures 3.5 and 3.6

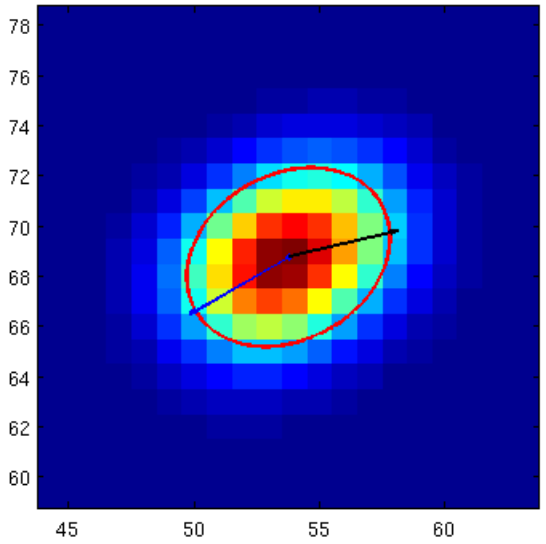

FIGURE 3.5: An example of the new Algorithm. The blue vector is the candidate direction, the black vector is the skewness vector. The magnitudes are equal and both 5 pixel lengths long for ease of viewing.

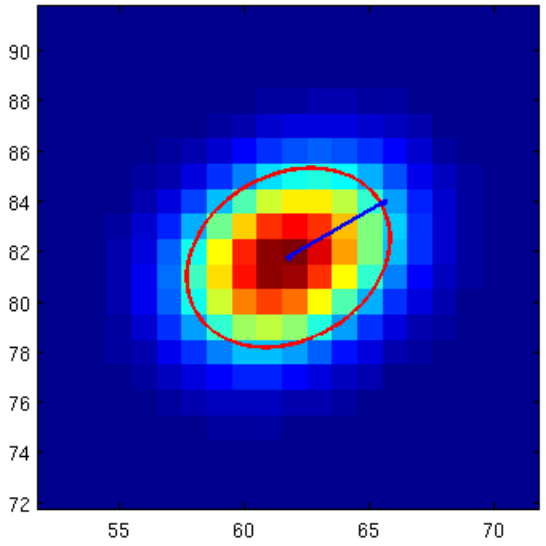

FIGURE 3.6: The blue vector is our decided direction for the track. The magnitude of the vector is larger than unity for ease of viewing.

As a note, it may be useful to perform this same moment vector calculation with respect to the basis in which the covariance matrix is diagonal. While this calculational option is not exercised in this thesis, we hope to further explore this technique in future research.

This new algorithm is distinct from the original algorithm as it allows for the

comparison of two different projections of the nuclear recoil tracks. The original algorithm only explores the projection of the track along the semi-major axis. In the following sections we will explore the efficacy of this algorithm, and explore what information the above quantities can give us regarding the imaged tracks.

3.2 Straight Simulated Nuclear Recoils

To test our new track direction finding algorithm, we simulate various sets of tracks. We began by drawing tracks representing straight nuclear recoils, then we moved to a more sophisticated method of drawing the recoils which allowed for the deviation of the track from its initial direction.

3.2.1 Drawing Straight Tracks

In the first set of simulations, the goal was to draw the path of the recoiling fluorine atom, assuming a perfectly straight recoil path, at various energies, path lengths, and incident angles, and simulate the effects of the detector and imaging CCD device. In drawing our straight tracks, we choose an energy in keVee and then set the pixel intensity along the track utilizing the known conversion factor from ADUs to keVee given previously. We are also able to adjust the slope of the track, which corresponds to the change in energy with respect to a change along the track length. It is important to note, that these simulations un-physically characterize the tracks with a linearly decreasing ionization. Physical nuclear recoil tracks do not display this, however the resulting image will have similar properties to the images produced by the detector. For this reason, we will proceed with these simulations, and then move to a more physically realistic simulation process in the next section. We solve analytically for the continuous intensity as a function of the total energy and the desired fraction in the first and second halves in appendix A. We will refer to lengths in millimeters, and energy in keVee. We defined the slope of our function by how much of the total energy (intensity) lies in the first half of the track, and how much lies in the second half. We can now talk about the percentage of total energy in the front and end parts of the track. This energy ratio will be referred to as the front half "over" the end half, for example a track which contains 65% of the total energy in the front half would have 35% of the energy

in the end half. Furthermore, the front half of the track is always considered to be the half which has the most energy.

Once we had our function, we needed to pixelate the line, and assign the pixels an energy based on the value of the function as it passed through the pixel. We did this by inspecting each pixel and determining if a line projected from the function into the x-y plane would pass through the pixel. If it would, we then computed a line integral through the pixel from the two points where it passed into the pixel to where it passed out of the pixel. We then assigned the pixel's value equal to this line integral. The code for drawing the track can be found in appendix B.

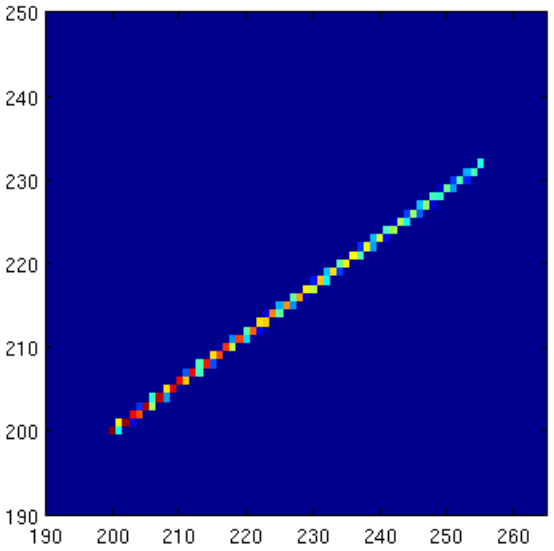

FIGURE 3.7: A straight pixelated track at an angle of 30 degrees to the horizontal. The direction of the track is in the direction of red pixels to blue pixels.

In figure 3.7, we have drawn the track in a finer space of pixels, where each pixel side represents 16 μm. The intensity of the pixels is scaled so that there are 250 ADU/keVee.

We used this finer space of pixels so that we could more easily control the distribution of energy along the track. Because we have a continuous analytic function for the energy along the track length, the finer we bin the pixels, the closer we come to approximating that distribution. However, once we have visually represented the deposited energy in our finer pixels space, we need to use it in our simulation of the detector's image properties.

3.2.2 Applying the New Algorithm

We wanted to determine how accurately the skewness vector represents the direction of the track before we simulate the diffusion aspect of the detector. We found that the skewness vector pointed into the hemispherical half of the ellipse with the lower integrated intensity. (See figure 3.8)

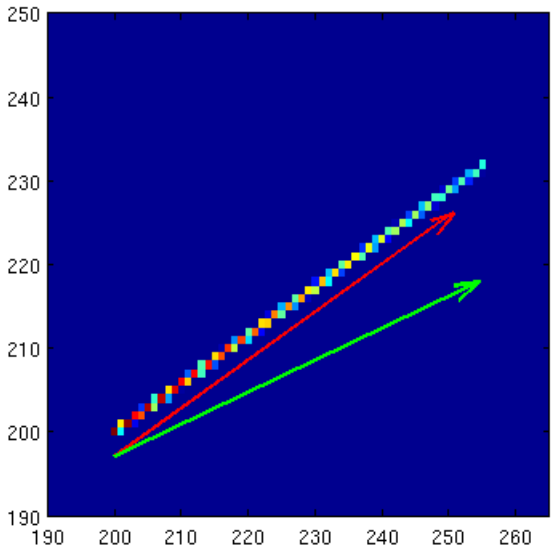

FIGURE 3.8: The straight track in figure 3.7. The red vector is the direction of the vector parallel to the semi-major axis which has the smallest angle to the skewness vector. The green vector is the skewness vector.

The above track has an energy of 50 keVee, and a total length of one millimeter. This length and energy was chosen for two reasons. First, 50 keVee on the order of the expected deposited energy of a WIMP interaction [5]. Second, a 50 keVee track is approximately one millimeter in length, measured from nuclear recoil data taken by the detector described in chapter one. It contains 65 percent of its total energy in the first half of its length, and 35 percent of its energy in the second half. The skewness vector does not always lie along the track itself, but does lie within about 20 degrees of the true track direction. The skewness projected along an axis is related to the length of the track projected onto that axis. Because of this, when the track is close to one axis, the measured skewness along the orthogonal axis does not experience much change for small changes in the angle relative to the parallel axis.

However, the semi-major axis of the best fit ellipse does capture the angle along the track well, but is not able to tell us which portion of the track contains more

intensity. In order to break this ambiguity between two antiparallel vectors along the semi-major axis, we choose the vector whose angle relative to the skewness vector is smallest.

We can see the effectiveness of this method by taking the example track above and rotating it through 360 degrees. We will measure the angle of the skewness vector relative to the positive x-axis, and we will measure the angle resulting from using the skewness vector along with the the angle resulting from the use of the second moment semi-major axis along with the skewness vector. The results of this are shown in figure 3.9

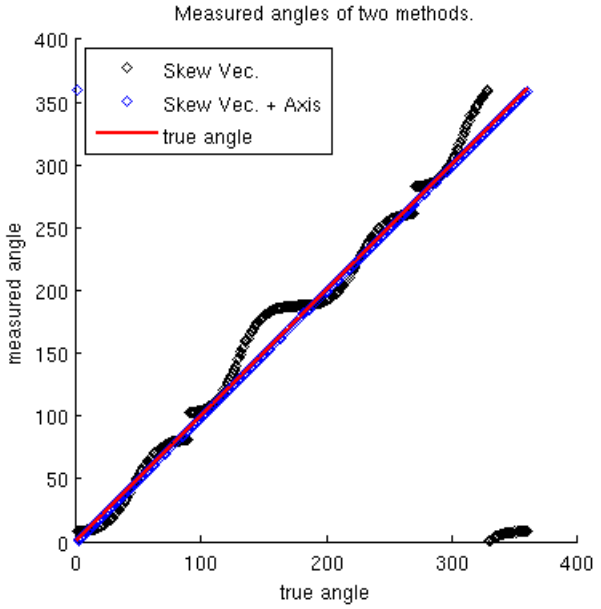

FIGURE 3.9: Measured angles of the skewness vector and the track angle resulting from the combined use of the skewness vector and the semi-major axis. There is a periodicity in the angles. The angles are not measured above 360 degrees, and instead return to 0 at 360.

The strong agreement between the line produced by the blue diamonds and the red line representing the angle at which the track was drawn indicates that this combination of the two moments is a good candidate for further investigation.

Next we need to simulate the diffusion and pixelation of the track. We do this by convolving the two dimensional track image with a Gaussian of the appropriate width, 350μm, which corresponds to 21.875 pixels in the image. Once the track has been convolved, we rebin the image using a bicubic interpolation into a pixel space 10 times smaller, so the new pixels are 160μm on a side. The bicubic interpolation in this track-generation algorithm allows for the rebinning of the image into an

image which is not necessarily scaled by an integer. This was chosen to maintain the plasticity of the track generation scheme. This results in a coarser image.

Finally we need to add noise to the image to fully simulate the detector. To find the appropriate noise profile to add, we selected 10 real images from the detector, removed any objects from them, and inspected the mean value and standard deviation of the pixel values. We found that in the images, the pixels had a mean value of zero, with a standard deviation of 10 ADU's. It is important to note that the images we are considering have already undergone a flat-fielding process by N. Phan. The convolved, rebinned, and noise added image is shown in figure 3.10.

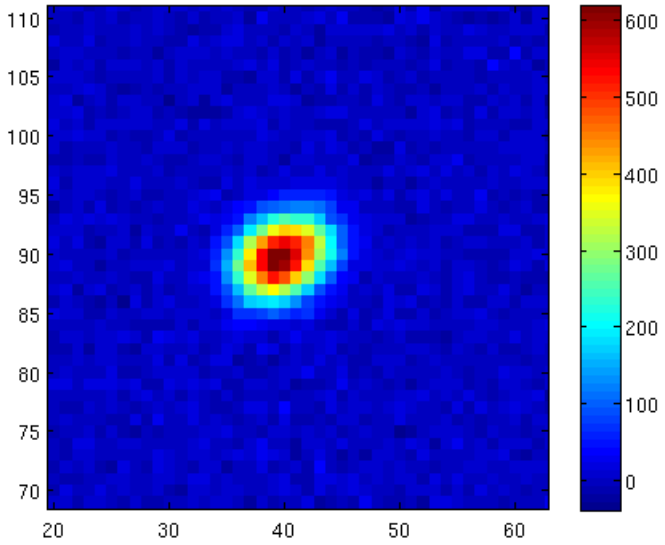

FIGURE 3.10: The example track after convolution, rebinning and noise.

3.2.3 Trials of Algorithms on Straight Tracks

After developing the above method for simulating straight nuclear recoil tracks, we were able to simulate many nuclear recoil tracks and compile statistics regarding the efficacy of the algorithms.

The parameters we varied were the length of the track, the head/tail energy ratio, and the angle along which the slope decreased. We looked at tracks between 2 mm and 0.2 mm in length, we looked at head/tail energy ratios of between 50/50 to 65/35 in integer steps in the numerator. (Each increasing step in the numerator is accompanied by a decreasing step in the denominator.) For each combination of

length and energy ratio we generated 7200 tracks. These tracks were drawn evenly distributed across 360 degrees relative to the x-axis in the image, with 20 tracks per angle. Furthermore, each track was placed at a randomly selected location in each image, and each image had unique noise generated for it. We did not vary the integrated energy along the track, and each had a 50 keVee total energy.

Because we knew the true direction of the track along which the intensity decreased, we were able to determine the fraction of the total number of events which the algorithm correctly identified. Our experiment is interested in ruling out an isotropy in the directionality of nuclear recoils. To this end, we will consider a direction that our algorithm finds to be "correct" if it is within 90 degrees of the true direction of the track. In other words, we are interested in determining the correct directional hemisphere. Given this, we found that the new algorithm performed as well, or better than, the original algorithm. In figures 3.11 - 3.14 we show plots, each of which has the fraction of total events which were correct as the ordinate variable (y-axis). Because the number of combinations of track length and energy ratio is so large we restrict our presentation to two samples of each. First we inspect tracks across all skews holding the length constant at 1mm tracks, and 0.5mm tracks. Then we hold the energy ratios constant at 60/40 and 53/47 and inspect tracks of all lengths between 0.2 and 1mm.

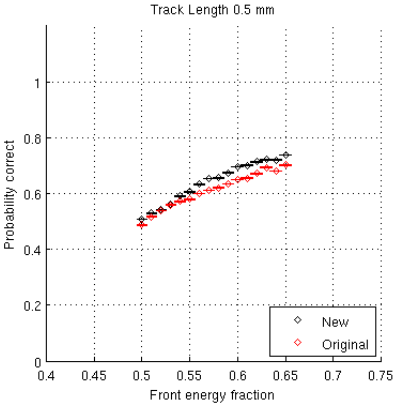

FIGURE 3.11: Fraction Correct as a function of front fractional energy, holding length constant.

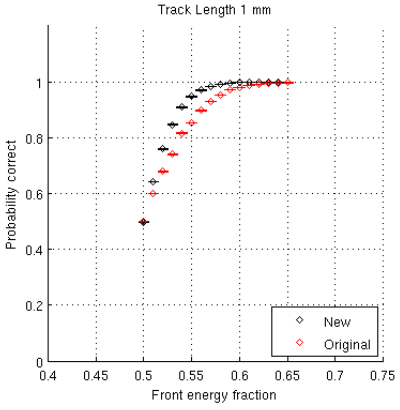

FIGURE 3.12: Fraction Correct as a function of front fractional energy, holding length constant.

We can see from these figures that the new algorithm performs at least as well as the original algorithm. It also provides improvement at lower track lengths and more skewed distributions over the original algorithm.

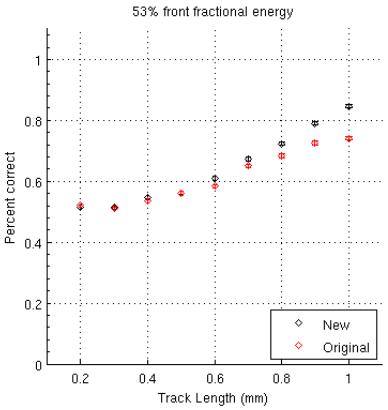

FIGURE 3.13: Fraction correct as a function of track length, holding front fractional energy constant.

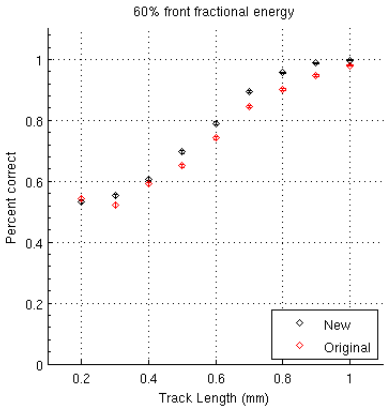

FIGURE 3.14: Fraction correct as a function of track length, holding front fractional energy constant.

In our characterization of these tracks, we also discovered a correlation between the number of correct tracks and the skewness vector's magnitude. We chose a specific track length and fractional energy and created a histogram of the measured skewness magnitudes, distinguishing between tracks whose direction the algorithm correctly and incorrectly identified. This histogram is shown in figure 3.15.

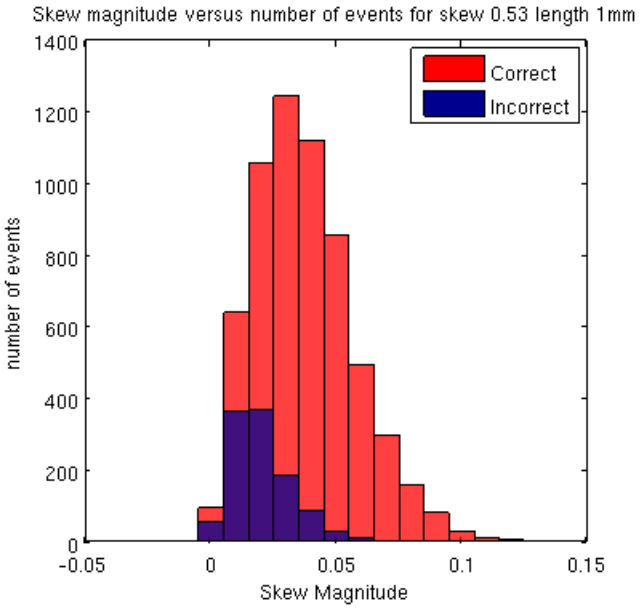

FIGURE 3.15: Number of correct and incorrect events as a function of the skewness vector magnitude.

This plot is telling us that the larger the skewness magnitude, the more likely it is that the algorithm calculated the track direction correctly. Given this, we possibly

have a new method of discriminating against low quality or noisy tracks in the real data.

3.3 Nuclear Recoil Tracks Simulated with SRIM

We next proceeded with our simulations by using a common software package in nuclear physics called SRIM, which stands for The Stopping and Range of Ions in Matter [14]. This toolbox allows for a much more physically realistic simulation of the nuclear recoil tracks, including deviations (straggling) from the initial recoil direction, and an intensity profile which does not decrease linearly along the track length. Within SRIM there is a secondary toolbox called TRIM which allows for the calculation of interactions between energetic ions being propelled into a volume of matter. With this tool we can simulate a single fluorine ion as it travels through a volume of CF_4 gas. We can specify the particle's initial kinetic energy and its initial angle relative to the x-axis in the x-y imaging plane. In the software package, we specified carbon tetrafluoride gas as the target for the fluorine ions. We specified that the target is a gas, and we used the appropriate density for the target, calculated using the ideal gas law from the known pressure in the real gas chamber. This calculation resulted in a density of approximately 0.00047 $\frac{g}{cm^3}$ for CF_4 gas at a pressure of 100 Torr.

We used the quickest mode of the TRIM simulation which only tracks the primary fluorine ion as it passes through the gas. In this mode we do not calculate the energy lost from the initial (primary) fluorine ion into secondary fluorine ions that are freed from the the molecular structure of the gas. In the detector, neutrons can also free carbon atoms resulting in nuclear recoil tracks. However, there are approximately five fluorine nuclear recoils for each carbon nuclear recoil, so in the interest of computation time we only simulated fluorine tracks.

TRIM can run thousands of ions projected into the target volume in a matter of minutes. It outputs a text file with details about each ion's path through the target. The file contains the ion number, the kinetic energy of the ion at each step, its x-y-z location coordinates at each step in angstroms, and how much energy was lost due to electronic stopping at each step in eV per angstrom. It is important to note that the tracks have a full three dimensional coordinate space during the simulation, but are projected into the x-y plane during image creation.

In order to produce an image of the track we must integrate the energy the ion loses due to ionization over the length of the track and then bin that energy appropriately. Energy is also lost to heat and scintillation, but the detector is only sensitive to the loss in electronic stopping. The code which was used to perform this task has been included in appendix B.

Once we had images of the tracks at the appropriate pixel scale of 16 μm per pixel (figure 3.16), we were able to perform the same convolution, rebinning, and noise addition that was performed for the straight tracks. The before and after results are shown in figures 3.16 and 3.17.

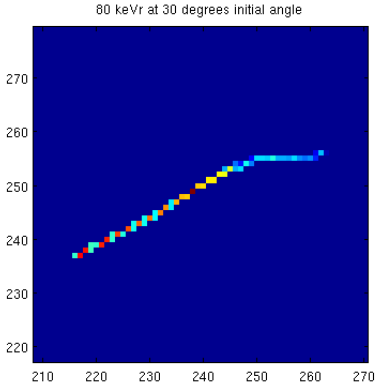

FIGURE 3.16: A nuclear recoil track as simulated by TRIM package in SRIM.

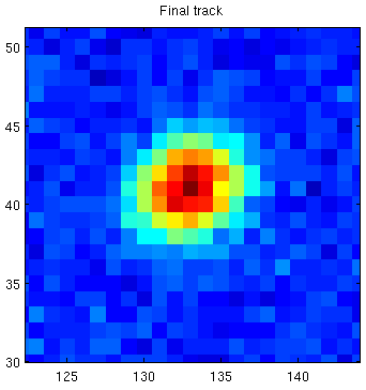

FIGURE 3.17: The same track after convolution, rebinning and noise.

We performed simulations at 50 to 100 keVr incident ion kinetic energy, increasing in steps of 10 keVr. For each energy, we simulated the tracks at incident angles of 0, 30, 45, and 60 degrees relative to the x-axis in the imaging plane. For each pair of energy and angle, we simulated 1000 unique tracks. This led to an overall 24,000 tracks for our simulation statistics.

We also took a subset of the same set of tracks and reflected them across the y-axis to simulate tracks in the second quadrant, in order to determine any angular dependence of the algorithm. We found a very small angular dependence arising from the Source Extractor package. This resulted in approximately one percent of the reflected tracks having a different direction identified than their mirrored but otherwise identical counterparts in the first quadrant. Because there was almost no change in the fraction correct as a function of quadrant, we restricted our investigation to the first quadrant to save time.

Furthermore, the results presented here will be over all of the tracks, independent of the angle and the initial energy. We are not able to strongly discriminate between events as a function of their track angle before our analysis in the real data, so here the data is presented as though all events were taken from one real data set.

3.3.1 Trials of Algorithms on SRIM Generated Tracks

The events we are interested in finding occur rarely, so we want to be able to include as many of the events as possible, but in doing so we open ourselves to more mischaracterized events. We want a method of including as many events as possible, and having as high a probability correct as possible.

In the tracks generated by SRIM, there were two ways that we considered defining the direction of the track. In one method we defined the track's true direction as the initial angle we told the software to generate; in the other, we applied the two algorithms to the tracks before we convolved, rebinned and added noise. For both methods, we determined that the algorithm calculated the correct direction if its measured direction was within 90 degrees of the defined direction of the track.

In the second method of determining track direction, we were able to determine how well the algorithms agreed with their own methods before and after convolution, rebinning, and the addition of noise. We wanted to know: Did the new algorithm agree with itself before and after more strongly than the original algorithm agreed with its own results?

For the second method of determining the track direction, we applied both algorithms to the tracks before convolution, rebinning, and adding noise. The algorithms gave a direction to each track separately from one another. Then we convolved, rebinned and added noise. We applied each algorithm to each track separately and noted each algorithm's calculated direction. Then we compared the new algorithm's calculated direction after the convolution, rebinning, and noise to its calculated direction before convolution, rebinning and noise. We performed the same set of steps for the original algorithm. We found that the two algorithms applied to the data before and after agreed with their own results at the same rate.

However, using the first method, by defining the correct direction of the track to be the initial direction input to the SRIM software, we did see a small improvement in the new algorithm over the original algorithm. The fraction of correct events as a function of measured energy in keVee for these two definitions are presented in figures 3.18 and 3.19.

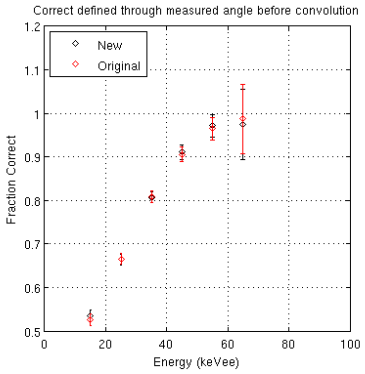

FIGURE 3.18: Correct defined by applying the algorithms before convolution.

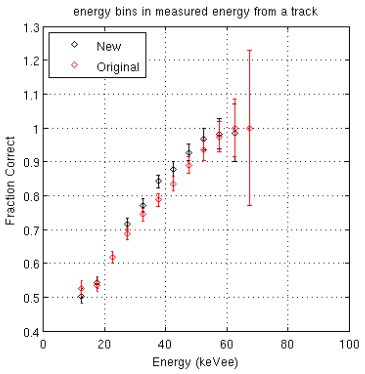

FIGURE 3.19: Correct defined by the initial angle specified in SRIM.

We also inspected the same skewness magnitude distribution, and how it related to the number of events that were marked as correct or incorrect, using the defined direction of the track to be what was input to the software. In this method we saw the same correlation between the fraction of correct events and the magnitude of the skewness vector. As the magnitude of the skewness vector grows, the proportion of tracks whose direction were correctly measured increases. This is shown in figure 3.20.

Although we have only made a slight improvement on the overall percentage of correctly identified events, we have introduced a new parameter which can be used to our advantage. We will discuss this advantage in the next section.

3.4 Methods for Excluding Poor Quality Events

Each of the tracks that were created with SRIM were convolved, rebinned, and had noise added to fully simulate the effects of the detector. This results in each track having a unique energy which is obtained by summing all of the pixels in the track and then converting from ADU's to keVee. The detector produces images with approximately 250 ADU's per keVee, so we can convert from our pixel values

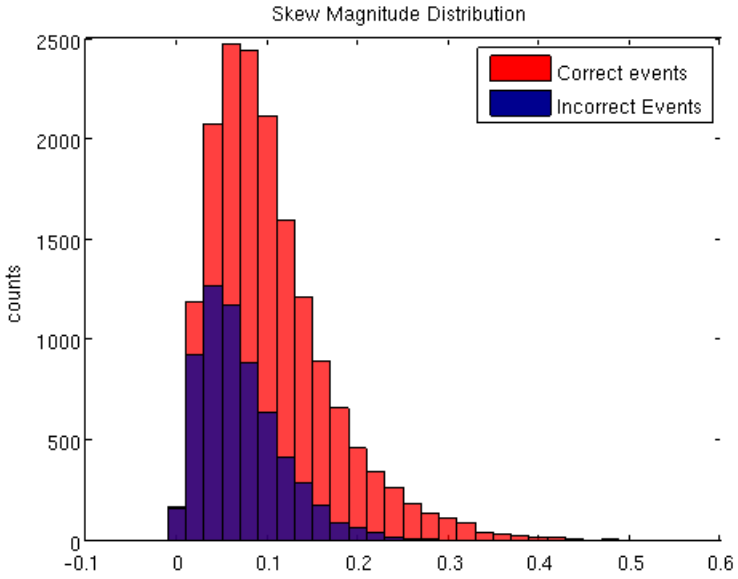

FIGURE 3.20: Number of correct and incorrect events as a function of the skewness vector magnitude for SRIM simulated events.

to the energy in the track. Furthermore, each track also has some length along its semi-major axis. We can calculate this length in pixels by looking at the value of $2\sqrt{\lambda_2}$. This is the length of the semi-major axis in pixels. We then use the relationship that one pixel is 0.160mm in length, and we can calculate the length of each track. We produce a plot of each event's energy versus its track length 3.21.

We are interested in finding events for which we are very confident that we determined the directionality of the track correctly. To this end, we will divide the distribution in figure 3.21 into thirds, producing figure 3.22. We will then inspect the fraction of events in the top third that we got correct, the middle third, and the bottom third. We expect there to be more events correct in the top third, because this represents the tracks with the highest lengths given a specific energy. These longer tracks represent the straightest nuclear recoils. The longest tracks at a given energy may represent the highest quality events because they have the most physical extent to carry information about the direction of the recoil.

We can get the best probability of our algorithm assigning the correct direction to the track if we exclude all but the top third of events, because these events are the straightest tracks. However, this involves disregarding 66% of events.

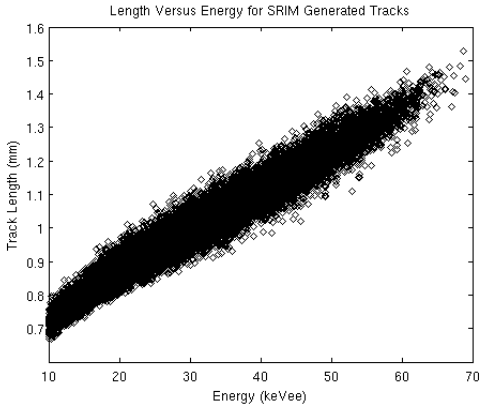

FIGURE 3.21: A scatterplot of each event's energy-length ordered pair.

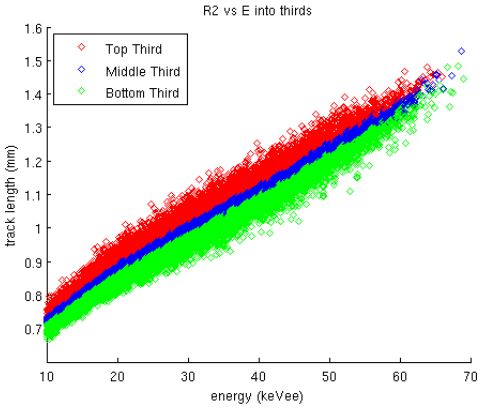

FIGURE 3.22: The same plot, divided into thirds. The blue seems smaller than the red and green because the events are more densely clustered.

Looking at our skewness magnitude distribution, we can see that we have a much larger probability of getting the track direction correct if we only consider events whose skew magnitude is above 0.1 (unitless). To this end, we inspected performing three cuts on the skewness magnitude, excluding events below 0.12, 0.1, and 0.07 respectively. In figures 3.23 and 3.24 we look at the fraction of correct events as a function of those cuts. In figure 3.23 we also inspect how many events at each energy bin and each skewness magnitude cut we actually included in our fraction correct.

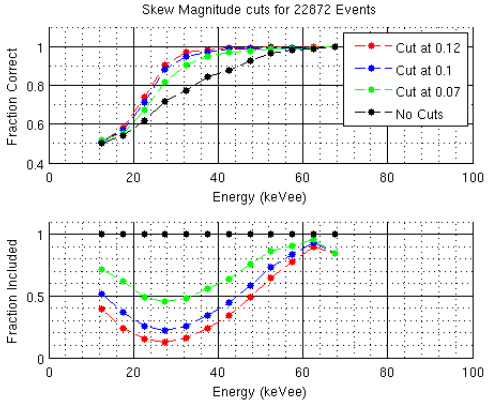

FIGURE 3.23: Top: The fraction of events correct at a given energy with three cuts excluding events below the listed skewness magnitude. Bottom: The fraction of events included at each point in the top plot. (Saxophone Plot)

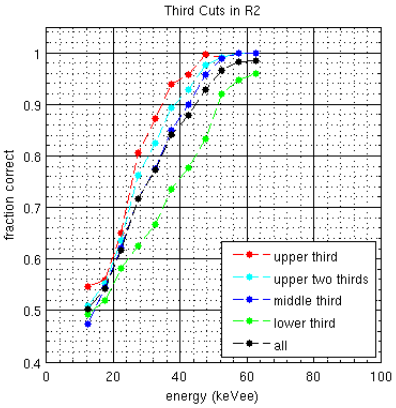

FIGURE 3.24: The fraction correct as a function of which third of the length vs. energy plot is being inspected. Each third only includes 33 percent of the events by definition.

Here, we can see that the green line in the skewness cut, corresponding to a skewness cut of 0.07, includes on average more than half of all of the events in the analysis. It also produces a probability correct which is nearly equal to the top thirds cut. The top third cut excludes 66% of the events. This skewness magnitude parameter appears to give a method of selecting good quality events which includes more of the total number of events. This will allow for a more comprehensive analysis of the data taken from the detector.

Chapter 4

Application of the New Algorithm to Real Data

There are many types of energetic events which can occur and leave tracks inside the gas volume that our detector will image. We have electronic recoils, which occur when an electrons move through the gas, producing ionization and light. We have alpha particles (helium nuclei) which can pass through the gas, leaving very large straight tracks through the gas (on the order of tens of centimeters,) and finally we have fluorine and carbon nuclear recoils which produce the tracks in which we are interested. If we inspect all of the images from our detector, and then view the tracks on a length versus energy scatter plot, we can discriminate between events. Electrons, being much lighter than nuclei, deposit much less energy per length of travel. This leads the plot of track length versus deposited energy to have two distinct classes of events.

If we want to apply the algorithms for real data, first we need to characterize nuclear recoils due to fluorine and carbon nuclei. We can gather data to perform this characterization by placing a source of neutrons near the detector. The neutrons produced by the source interact with the gas molecules in our fiducial volume, causing nuclear recoils when the fluorine or carbon atoms are freed from the molecular bonds. We acquire data during this process, and we can then produce a scatter plot of the length of the event versus the energy of the event. This data is shown in figure 4.1.

There are two large bands of data points, one nearly vertical, and one with a much more shallow slope. The more shallow band consists of the nuclear recoils

in which we are interested. Furthermore, we are really only interested in events below 200 keVee for two reasons. First, because higher than that it becomes easy to discriminate the track direction, and both algorithms reach 100% accuracy. Second, the rate at which WIMPs can deposit energy follows a falling exponential. Therefore the rate of WIMP interactions which deposit higher than 200 keVee is only a very small fraction of the total WIMP events. For this reason we are interested in improving our sensitivity for energies below 200 keVee.

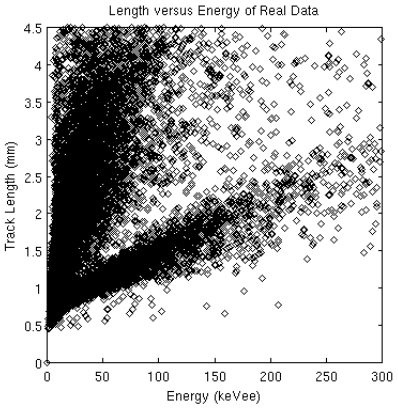

FIGURE 4.1: Scatter plot of track length versus energy for data taken from the detector.

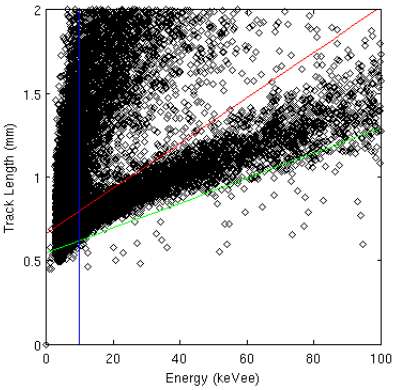

FIGURE 4.2: The nuclear recoil tracks lie between the red and green lines.

We will restrict ourselves to considering events which lie between the red and green lines, and to the right of the blue line in figure 4.2.

4.1 Application of the New Algorithm to Events in the Nuclear Recoil Band

As we investigated the application of the new algorithm to the real data, we came across an interesting piece of information. On the length versus energy plot, we plotted all events with a skewness magnitude of greater than 0.9 in red diamonds, and all other events in black diamonds. We found that there were very few events in the nuclear recoil band of skewness magnitude greater than 0.9. (Figure 4.3)

The number of events in the nuclear recoil band above 10 keVee and below 200 keVee was 4774, of those only 232 events had a skewness magnitude greater than 0.9. It is likely that this skewness magnitude exclusion in the nuclear recoil band

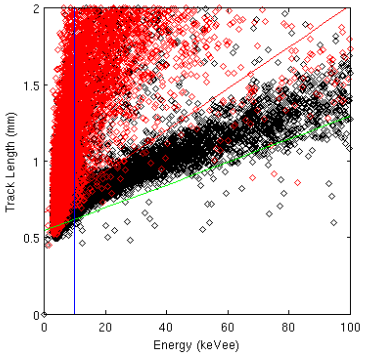

FIGURE 4.3: Red events have a skewness magnitude above 0.9.

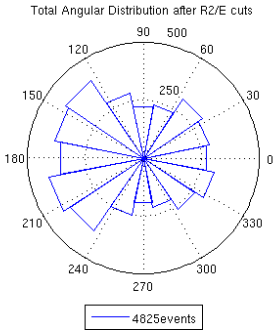

FIGURE 4.4: Angular distribution for all events in the nuclear recoil band.

is unique to this configuration of gases and pressures, but there may exist other skewness magnitude exclusions like this in other gases and pressures.

We applied the algorithm to all tracks in the entire data set, and began by excluding tracks outside of the nuclear recoil band. We also excluded events whose skewness magnitude was above 0.9, resulting in capturing approximately 95% of the nuclear recoil band between 10 and 200 keVee. We then investigated the distribution of measured angles in the set, and we display that distribution as an angular histogram in figure 4.4.

It is easy to see the asymmetry in the distribution of angles from the plot. We know from the experimental setup that for this data set, the neutron source was oriented in the direction of positive infinity along the x-axis, meaning that the neutrons are traveling from right to left, relative to the image. Therefore the nuclear recoils should have an asymmetry in the negative x direction. While we do see that in the angular distribution here, we would like to tighten the distribution so that we have relatively fewer events in the wrong direction, and relatively more events in the correct direction. Morgan, Green, and Spooner have shown that characterizing the nuclear recoil direction will reduce, by a factor of 10, the number of events required to prove an anisotropic distribution of nuclear recoils. However, they have also shown that a 70-80 percent likelihood of correct identification of initial recoil direction is required to obtain this reduction in required events [8].

We will begin by first splitting the nuclear recoil band into thirds, as we did for the simulated nuclear recoil tracks. We will define the correct direction of the track to be in the direction of negative x. We will investigate the probability of finding the

correct direction of the track as a function of which third of the band the track is in. We will also inspect the skewness magnitude distribution to determine how well it matches our SRIM simulated distribution. The distribution is shown in figure 4.5.

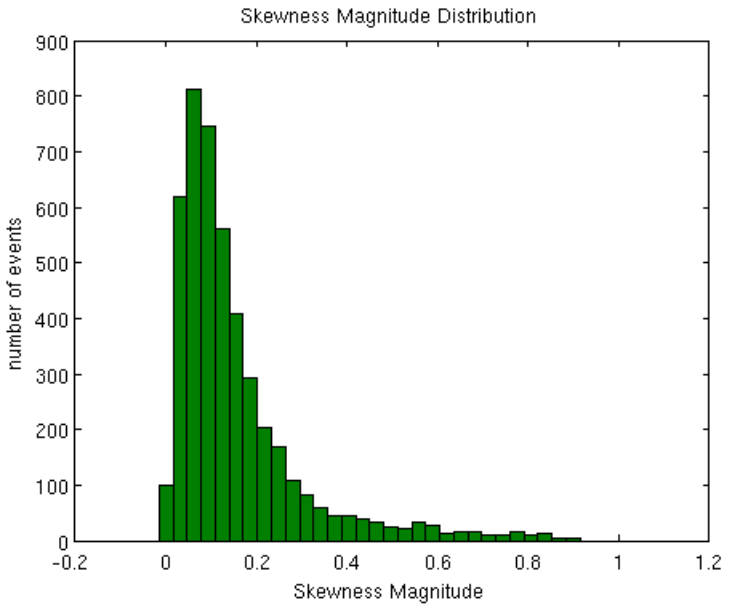

FIGURE 4.5: Skew Magnitude distribution for the nuclear recoil band of real data between 0 and 200 keVee.

Next we adjust the cuts on the skewness magnitude such that the least aggressive cut included approximately half of events, and investigated the probability distribution of getting the correct direction as a function of energy with those cuts. The probability correct curves for both the thirds cut and the skewness magnitude cuts are shown in 4.6 and 4.7. These plots fluctuate more than the equivalent plots in the SRIM data because we have approximately one fifth of the number of events to inspect.

Finally we will present the angular distribution resulting from selecting only events above skew magnitude 0.1 (the green line in figure 4.6), only events in the top third, top two thirds, and the distribution which results in selecting only events with skew magnitude above 0.1 and which are in the top third of the nuclear recoil band. These distributions are in figures 4.8 - 4.11.

Given these angular distributions, and the fractional correct plots, the new algorithm's skewness magnitude allows for the discrimination of lower quality events while discarding fewer events than the previous method. We can achieve a tighter

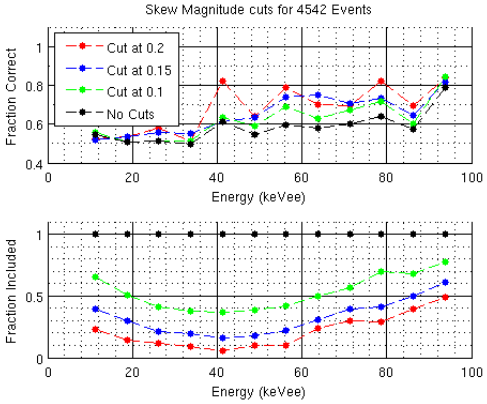

FIGURE 4.6: Top plot: Fraction of events with correctly identified directions above several different skewness magnitudes. Bottom plot: the fraction of included events at each energy.

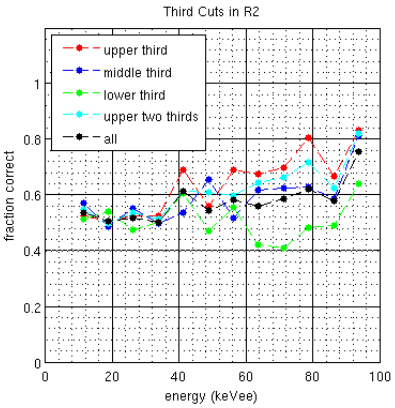

FIGURE 4.7: Fraction of events with correctly identified directions in several cuts of the length versus energy scatter plot.

angular distribution with more events, and retain more events without sacrificing our probability of measuring the correct direction of the track.

4.2 What's Next?

Both the cuts along the thirds of the length-versus-energy plot and the skewness are capable of excluding poor quality events from the analysis of the nuclear recoils. We have two parameters which can be used to indicate how likely it is that the direction indicated by the algorithm is correct. Ideally, in order to exclude as few events as possible, we should next develop a method of weighting these parameters together to create a new and dynamic method of estimating the likelihood of correctly measured track direction.

There also may be other methods of finding the track direction. Given that the detector has some known point spread function, it may be possible to deconvolve the tracks in the images. In the future we would like to investigate the possibility of applying algorithms developed for use by the astronomical community in the deconvolution of stellar objects to our detector images.

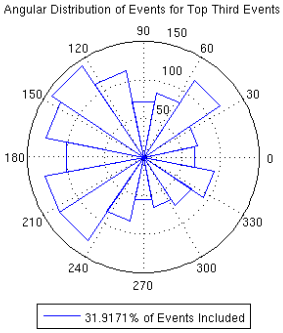

FIGURE 4.8: Angular distribution for events in the top third of the nuclear recoil band.

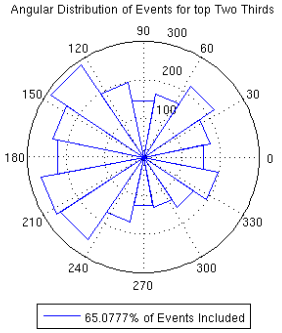

FIGURE 4.9: Angular distribution for events in the top two thirds of the nuclear recoil band.

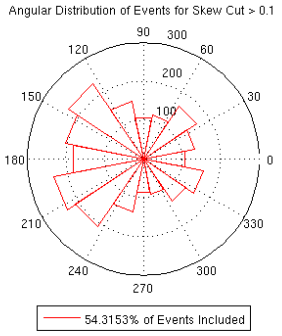

FIGURE 4.10: Angular Distribution for a skew magnitude cut at 0.1.

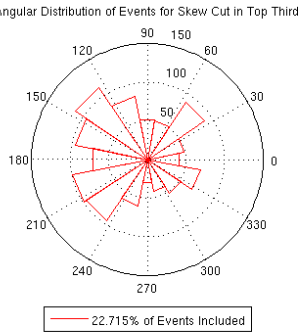

FIGURE 4.11: Angular distribution for skew magnitude cut at 0.1 in the top third.

There exist other methods of analyzing images other than deconvolution and moments. In the image analysis community there has been work done in the decomposition of images into a set of objects called shapelets. If it is possible to decompose our images into their most strongly weighted shapelets, it may be possible to determine the asymmetries in the images. We may be able to use this information to more robustly assign head/tail directionality to individual events.

As we further analyze these images, our goal is to push the threshold for determining the directionality of events to be as low as possible.

Appendix A

Mathematics for Drawing Straight Tracks

In this appendix we will discuss the mathematics behind the drawing of straight track, pixelated images. We will start off by assuming that the track has some intensity value as a function of x and y in continuous space and we can map that intensity value into the z variable. Next, for a given track we know its length in the x-y plane, we know its total energy (intensity), and we know that we want some fraction of the energy to be in one half, and the corresponding fraction in the second half.

We will call the overall track length in the x-y plane b, we will call the total energy E. If we want pE of the energy to be in the front half of the track, by definition we must have $E(1-p)$ in the second half of the track. Using p as a variable that describes the fraction of energy in the front half of the track, we can begin to describe the situation mathematically. (Note: the front half of the track will always be the half which has greater energy.)

In figure A.1 we can see a side view of a track in a continuous space. There are two unknown quantities in the track, h and l, named because h is necessarily the highest point on the track, and l is the lowest.

Using the relationship that $E1 + E2 = E$ and that we know that $E1 = pE$ and $E2 = E(1-p)$, we can solve for what the two heights, h and l must be. After an application of geometry, we find the following:

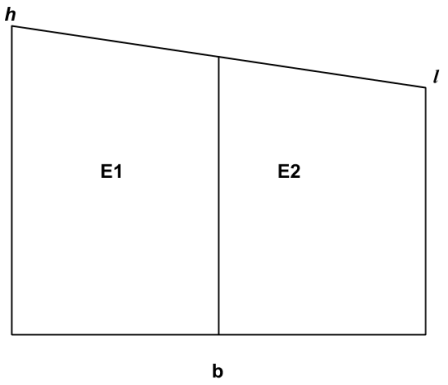

FIGURE A.1: Looking at a straight track, on its side, in continuous space. The total energy is E1+E2, and is represented by the total area in the trapezoid.

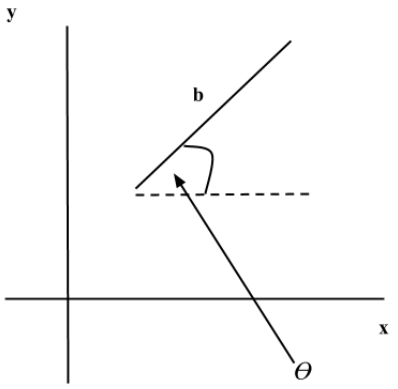

FIGURE A.2: Looking down on the same track in the x-y plane. The track makes an angle θ with the x-axis.

$$\begin{aligned} h &= \frac{2E}{b(1+\frac{3-4p}{4p-1})} \\ l &= \frac{h(3-4p)}{4p-1} \end{aligned} \tag{A.1}$$

And the slope of the line connecting h and l is then just $\frac{h-l}{b}$.

Next, we can note that there is a relationship between x and y, because we have a straight line in the x-y plane that begins and ends at two points. Because it is linear function, we will write y in terms of x and x in terms of y, as we will need both.

$$\begin{aligned} y &= \tan\theta(x-x_0)+y_0 \\ x &= \cot\theta(y-y_0)+x_0 \end{aligned} \tag{A.2}$$

Next, we know that the height of the function, z, is linearly related to the position along the base, which is a line in the x-y plane. This means that we can write z in terms of x or y.

We want the function represented by z to be at l when $x = x_0$ or when $y = y_0$. Next, we will parameterize z along the arc length in the x-y plane, called s. We can see that s is also a function of x:

$$\begin{aligned} s &= \sqrt{(x-x_0)^2 + (y-y_0)^2)} \\ &= \sqrt{(x-x_0)^2 + \tan\theta^2 (x-x_0)^2} \\ &= \frac{(x-x_0)}{\cos\theta}. \end{aligned} \tag{A.3}$$

Or equivalently it is a function of y:

$$\begin{aligned} s &= \sqrt{(x-x_0)^2 + (y-y_0)^2)} \\ &= \sqrt{\cot\theta^2 (y-y_0)^2 + (y-y_0)^2} \\ &= \frac{(y-y_0)}{\sin\theta}. \end{aligned} \tag{A.4}$$

We can now write z as a function of s, and then, by extension, a function of x or y:

$$\begin{aligned} z &= \frac{(h-l)}{b} * s + l \\ &= \frac{(h-l)}{b} * \frac{(x-x_0)}{\cos\theta} + l \\ &= \frac{(h-l)}{b} * \frac{(y-y_0)}{\sin\theta} + l. \end{aligned} \tag{A.5}$$

And finally, to pixelate the actual image, we need to perform a line integral across the pixels that the actual line in the x-y plane falls into. This is done in the code above, and we will not go over the logic here. However, by inspecting the form of z above, it is apparent that if we integrate along y for small values of theta, the z function is nearly infinite, as $\sin\theta = 0$ if $\theta = 0$. Correspondingly if we integrate along x when θ approaches $\frac{\pi}{2}$, z is again singular. It is for this reason that we split the track drawing algorithms into two sections, one for $|\theta| < 45$ and one for $45 < |\theta| < 90$.

Appendix B

Matlab and Python Codes

This code was written for use in a Linux environment, and requires Source Extractor to be appropriately installed. The MATLAB code which is contained here was written and used in MATLAB R2014a. The Python code which is contained here was written and used in Python 2.7.

B.1 The Algorithms

B.1.1 Current Algorithm

Included below is the currently used algorithm for determing the head/tail of a nuclear recoil track in the images investigated in this thesis. This function was constructed in its entirety by Nguyen Phan at the University of New Mexico. Line formatting was performed by Joshua Martin for asthetic appearance in this thesis.

The function specified below requires a track on a zero background and the angle of the semi-major axis of the best fit ellipse. This angle is taken relative to the x-axis in the image.

```
function SkewMoments_M = projection_skewness(TrackMask,TrackAngle)
%%%%%% Track Projection Skewness Calculation

%%%%% Apply Radon transform on the mask of the track, 'TrackMask'
%%%%%(an image of the track where pixels that are not considered part
%%%%% of the track are set to zero but those part of the track are not )
%%%%% at the major axis angle, 'TrackAngle'.
```

```
radbragg = radon(TrackMask, TrackAngle);
%%%%% 'radbragg' is a vector with approximate length of the track major
%%%%% axis containing the projected track values

radbragg(radbragg <= 0) = [];
%%%%% set negative values on the vector edges to zero before calculating
%%%%% distribution moment

% %%%%% Moments
% m2 = moment(radbragg,2);
% m3 = moment(radbragg,3);
% m4 = moment(radbragg,4);
% m5 = moment(radbragg,5);

%%%%% Change vector data to a distribution before calculating moments and skewness

radtrans = radbragg.';
%%%%% switch from column to row vector

dt = zeros(size(radtrans,2),1);
%%%%% preallocate array for speed

for cc = 1:size(radtrans,2)
    dt(cc) = cc;
end

wt = round(radtrans)' ;
%%%%% weighting factors (can increase precision through a scaling factor)

wt(wt <= 0) = 0 ;

dnew = 0;
nnn = 1;
for iii = 1:length(dt)
    ggg = nnn + wt(iii);
    dnew(nnn:ggg-1) = repmat(dt(iii),wt(iii),1);
    nnn = ggg;
end

%%%%% Compute skewness using MATLAB built-in function

radtrans_dist = dnew;
SkewMoments = skewness(radtrans_dist);

%%%%% Compute skewness manually

cmom2 = moment(radtrans_dist, 2);
cmom3 = moment(radtrans_dist, 3);
SkewMoments_M = cmom3./(cmom2.^(3/2));

%%%%% Compute the test statistic for the skewness value for confidence
%%%%% determination (optional, not needed)

k = length(wt);
if (k > 2)
```

```
    G = SkewMoments_M.*sqrt(k*(k-1))./(k-2);
    SES = sqrt((6*k*(k-1))./((k-2)*(k+1)*(k+3)));
else
    G = SkewMoments_M;
    SES = 1;
end
SkewTestStat = G./(SES);

% %%%%% Additional Skewness Definitions
%
% QR = quantile(radtrans_dist,[.25 .50 .75]);
%%%% quartiles of the distribution
% PR = prctile(radtrans_dist,[10 50 90]);
%%%% 10, 50, and 90 percentiles
% meanR = mean(radtrans_dist);
%%%% mean of the distribution
% medianR = median(radtrans_dist);
%%%% median of the distribution
% ssigR = std(radtrans_dist);
%%%% standard deviation of the distribution
%
% %%% Bowley skewness:
% skB = (QR(3) + QR(1) - 2*medianR)./(QR(3)-QR(1));
%
% %%% Kelly skewness:
% skK = (PR(3) - 2.*PR(2) + PR(1))./(PR(3)-PR(1));
%
% %%% Pearson skewness:
% skP = 3.*(meanR - medianR)./(ssigR);
%
% %%% Cyhelsky skewness:
% skC = (size(radtrans_dist(radtrans_dist > meanR),2) -
% ... size(radtrans_dist(radtrans_dist < meanR),2))./(size(radtrans_dist,2));
%
% %%% Moments-based skewness:
% skM = skewness(radtrans_dist);
```

B.1.2 New Algorithm

This new algorithm was developed by Joshua Martin for use in analyzing images taken from the TPC developed by N. Phan. This algorithm requires Source Extractor [13] to be installed. Furthermore it requires the source extractor configuration file "default.sex" to be in the local folder in which the algorithm is run. Finally, it requires the fits image to be analyzed to be present in the folder.

There are four functions which are required: sextractorrobust, which applies Source Extractor to the fits image; extracted_Data which retrieves the information from the Source Extractor catalog file; twodmoments, which calculates all of the

first four two dimensional moments of an image that it is given; and analysisquiet, which synthesizes the other three functions and outputs a matrix whose rows contain various information specified in the function about each track found in the given image.

```
function sextractorrobust(filename)

  unix(['sed -i s/AAAA.txt/' filename '.txt/g default.sex'])
  unix(['sed -i s/AAAA.fits/extracted_' filename '.fits/g default.sex'])
  unix(['sex ' filename '.fits'])
  unix(['sed -i s/' filename '.txt/AAAA.txt/g default.sex'])
  unix(['sed -i s/extracted_' filename '.fits/AAAA.fits/g default.sex'])

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function info = extracted_Data(filename)

% % extracted_Data(filename) takes as an argument the file name of a fits
% image to be analyzed, WITHOUT the .fits extension.  It outputs a matrix
% whose rows are the data from source extractor.

sextractorrobust(filename)

fid=fopen([filename '.txt']);
tline = 0;
j = 0;
while 2>1
    tline=fgets(fid);
    if tline == -1
        break
    end
    j = j +1;
end
clear fid;

fid=fopen([filename '.txt']);
line_parameter = 0;
if j == 14
    error('Source Extractor was unable to find objects for this image')
end
for i = 1:j
    line_parameter = line_parameter + 1;
    tline=fgets(fid);
    if line_parameter > 14
        info(line_parameter-14,:) = str2num(tline);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Results = analysisquiet(filename,scale,weight)
```

```
% % Filename should NOT include the .fits extension.  example, if your file
% % is '~\track.fits' you should just enter 'track' as a string input to
% % analysis.
% analysis outputs the following data:
% [track number, energy, track location in x, track location in y,
% semi-major variance, third moment in x, third moment in y, unweighted
% semi-major variance]

params = extracted_Data(filename);

full_Image = fitsread(['extracted_' filename '.fits']);

A = size(params);
numberofTracks = A(1);
clear A;


for i = 1:numberofTracks

xmin=params(i,2)-2;
xmax=params(i,4)+2;
ymin=params(i,3)-2;
ymax=params(i,5)+2;
xpeak=params(i,6);
ypeak=params(i,7);
x0=params(i,8);
y0=params(i,9);

if xmin < 2 || xmax > 168 || ymin < 2 || ymax > 170
    continue
end

track = full_Image(ymin:ymax,xmin:xmax);
mask = track;
mask(mask~=0) = 1;
[ypeak xpeak] = find(track == max(max(track)));

track_Moments = twodmoments(track,weight);

% % twodmoments is a code which outputs the first four moments of a 2-d
% % data distribution.

mask_Moments = twodmoments(mask,1);
mean = [track_Moments(2),track_Moments(3)];
covar = (1/track_Moments(1))*([track_Moments(4) track_Moments(5); ...
    track_Moments(5) track_Moments(6)]);
[track_vecs, track_vals] = eig(covar);

covar_mask = (1/mask_Moments(1))*([mask_Moments(4) mask_Moments(5); ...
    mask_Moments(5) mask_Moments(6)]);
[mask_vecs, mask_vals] = eig(covar_mask);
```

```
direction = [(mask_Moments(2)-xpeak) (mask_Moments(3)-ypeak)];
direction = 5*direction./norm(direction);

track_Length = sqrt(scale*mask_vals(2,2))*2

energy=sum(sum(track))/250;


Results(i,:) = [i energy track_vecs(1,2) track_vecs(2,2) track_vals(2,2)...
    track_Moments(7)./(track_Moments(1))...
    track_Moments(10)./(track_Moments(1))...
    mask_vals(2,2) direction(1) direction(2) track_Length];

end
fclose('all');
end
```

B.2 Drawing Nuclear Recoil Tracks

Both of the following pieces of code draw nuclear recoil tracks onto a zero background image that is 500 by 500 pixels in size. These pixels correspond to 16 μm in length on a side. Subsection B.2.3 addresses the convolution, resizing and addition of noise to appropriately simulate the diffusion and noise in the detector.

B.2.1 Drawing Straight Tracks

In this piece of code, the user must specify the desired length of the track (in millimeters), the desired energy in the track (in keVee), the desired angle of the track (relative to the x-axis in degrees), the proportion of energy contained within the front half of the track (in a decimal between .50 and .75), and whether the intensity of the track should be drawn as increasing from the angle, or decreasing from the angle.

Furthermore, there are two functions for drawing tracks, one is for drawing tracks whose angle in magnitude is less than 45 degrees from the x axis, this is called shallowtrack. The other is for drawing tracks whose angle in magnitude is between 45 and 90 degrees relative to the x axis. This function is called steeptrack. The reason for the two functions is explained in appendix A. These two files only take arguments between -90 and 90 degrees. Other angles can be created by changing

the direction of the track with the increasing (inc) argument. If inc == 1, the track will be drawn increasing, if inc == -1, the track will be drawn as decreasing.

```
function D = shallowtrack(energy,length,skew,theta,inc)

if abs(theta) >= 45
    error('Please input an angle of magnitudes less than 45 degrees')
end

L=length*1000/16;
p=skew;
D=zeros(500,500);
% t is the angle that the line makes with the x-axis
t=theta;
% starting points of x and y

% cx=round(200*rand+150);
cx = 200;
cy = 200;
% cy=round(200*rand+150);
f=@(x)(tand(t)*(x-cx)+cy);
g=@(y)(cotd(t)*(y-cy)+cx);
%Ending points of x and y
ex=cx+L*cosd(t);
ey=f(ex);
x=linspace(cx-10,ex+10,10000);

h=(energy*2*250/L)*((1+(3-4*p)/(4*p-1))^-1);
l=h*((3/8-p/2)/(p/2-1/8));


if inc>0
    %This is my function
    z=@(x)(((h-l)/L)*((x-cx)/cosd(t))+l);
    %this is the same function but with the appropriate change of
    %integration variable multiplied so that I can integrate it.
    zi=@(x)(((h-l)/L)*((x-cx)/cosd(t))+l)*(1/cosd(t));
else
    z=@(x)((-1*(h-l)/L)*((x-cx)/cosd(t))+h);
    zi=@(x)((-1*(h-l)/L)*((x-cx)/cosd(t))+h)*(1/cosd(t));
end



for i=cx:(round(ex)+1)
    if t>=0
        for j = cy:(round(ey)+2)
            if g(j-.5)<=(i+.5) && g(j-.5)>(i-.5)
                if f(i+.5)<=(j+.5) && f(i+.5)>(j-.5)
                    D(j,i) = integral(zi,g(j-.5),i+.5)*1;
                elseif g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,g(j-.5),g(j+.5))*1;
                elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                    D(j,i) = -integral(zi,g(j-.5),i-.5)*1;
```

```
                end
            elseif f(i+.5)<=(j+.5) && f(i+.5)>(j-.5)
                if g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,g(j+.5),i+.5)*1;
                elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                    D(j,i) = integral(zi,i-.5,i+.5)*1;
                end
            elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                if g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,i-.5,g(j+.5))*1;
                end
            end
            if isnan(D(j,i)) == 1
                j = j
                i = i
                break
            end
        end
    else
        for j = round(ey)-1:cy
            if g(j-.5)<=(i+.5) && g(j-.5)>(i-.5)
                if f(i+.5)<=(j+.5) && f(i+.5)>(j-.5)
                    D(j,i) = -integral(zi,g(j-.5),i+.5)*1;
                elseif g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = -integral(zi,g(j-.5),g(j+.5))*1;
                elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                    D(j,i) = -integral(zi,g(j-.5),i-.5)*1;
                end
            elseif f(i+.5)<=(j+.5) && f(i+.5)>(j-.5)
                if g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,g(j+.5),i+.5)*1;
                elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                    D(j,i) = integral(zi,i-.5,i+.5)*1;
                end
            elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                if g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,i-.5,g(j+.5))*1;
                end
            end
            if isnan(D(j,i)) == 1
                j = j
                i = i
                break
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function D = steeptrack(energy,length,skew,theta,inc)


if abs(theta) < 45
    error('Please input an angle of magnitude greater than 45 degrees')
```

```
end


L=length*1000/16;
p=skew;
D=zeros(500,500);
% t is the angle that the line makes with the x-axis
t=theta;
cx = 200;
cy = 200;

f=@(x)(tand(t)*(x-cx)+cy);
g=@(y)(cotd(t)*(y-cy)+cx);
%Ending points of x and y
ex=cx+L*cosd(t);
if abs(t)==90
    ey=cy+L*sind(t);
else
    ey=f(ex);
end
x=linspace(cx,ex,10000);

h=(energy*2*250/L)*((1+(3-4*p)/(4*p-1))^-1);
l=h*((3/8-p/2)/(p/2-1/8));


if inc>0
    %This is my function
    z=@(y)(((h-l)/L)*((y-cy)/sind(t))+l);
    %this is the same function but with the appropriate change of integration
    %variable multiplied so that I can integrate it.
    zi=@(y)(((h-l)/L)*((y-cy)/sind(t))+l)*(1/sind(t));
else
    z=@(y)((-1*(h-l)/L)*((y-cy)/sind(t))+h);
    zi=@(y)((-1*(h-l)/L)*((y-cy)/sind(t))+h)*(1/sind(t));
end

for i=cx:round(ex)
    if t>=0
        for j = cy:round(ey)
            if g(j-.5)<=(i+.5) && g(j-.5)>=(i-.5)
                if f(i+.5)<=(j+.5) && f(i+.5)>(j-.5)
                    D(j,i) = integral(zi,j-.5,f(i+.5))*1;
                elseif g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,(j-.5),(j+.5))*1;
                elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                    D(j,i) = integral(zi,(j-.5),f(i-.5))*1;
                end
            elseif f(i+.5)<=(j+.5) && f(i+.5)>=(j-.5)
                if g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,(j+.5),f(i+.5))*1;
                elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                    D(j,i) = integral(zi,f(i-.5),f(i+.5))*1;
                end
            elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
```

```
                if g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,f(i-.5),(j+.5))*1;
                end
            end
        end
    else
        for j = round(ey):cy
            if g(j-.5)<=(i+.5) && g(j-.5)>=(i-.5)
                if f(i+.5)<=(j+.5) && f(i+.5)>(j-.5)
                    D(j,i) = -integral(zi,j-.5,f(i+.5))*1;
                elseif g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = -integral(zi,(j-.5),(j+.5))*1;
                elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                    D(j,i) = -integral(zi,(j-.5),f(i-.5))*1;
                end
            elseif f(i+.5)<=(j+.5) && f(i+.5)>=(j-.5)
                if g(j+.5)<=(i+.5) && g(j+.5)>(i-.5)
                    D(j,i) = integral(zi,(j+.5),f(i+.5))*1;
                elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                    D(j,i) = -integral(zi,f(i-.5),f(i+.5))*1;
                end
            elseif f(i-.5)<=(j+.5) && f(i-.5)>(j-.5)
                if g(j+.5)<=(i+.5) && g(j-.5)>(i-.5)
                    D(j,i) = -integral(zi,f(i-.5),(j+.5))*1;
                end
            end
        end
    end

end
```

B.2.2 Tracks from SRIM Data

The following is a set of codes written in Python for extracting data from the SRIM [14] output file, and drawing those into the appropriate track fits images.

This part was coded in Python because Python's file read/write is faster than Matlab's and we were able to include sqlite query functionality which drastically improved read speed of the SRIM output text files.

The file srim_toFits.py requires another code which was written by Peter Nary for use in this project, titled srimReader.py.

```
# srim_toFits.py

'''
Created on Feb 14, 2015

@author: Joshua Martin
```

```
'''


from __future__ import division
import pyfits
import scipy.io
import matplotlib.pyplot as plt
import numpy as np
import copy
import os
import sys
sys.path
sys.path.append('/home/USER/Python/DRIFT/')
from srimReader import srimParser


energy = 50
angle = 60
set = 1001
sampleParse = srimParser("/home/USER/TrackTextFiles/EXYZ_" + \
str(energy) + "keV" + str(angle) + "Deg.txt")

srim_matout = np.empty([1,7])
srim_begend = np.empty([1,4])
if not os.path.exists('/home/USER/SRIM/track' + str(energy) + \
                'keV' + str(angle) + 'Deg'):
    os.makedirs('/home/USER/SRIM/track' + str(energy) + \
                'keV' + str(angle) + 'Deg')

for ionNumber in range(1,set):
    srim_track = sampleParse.getIonValues(ionNumber)
    cx = np.round(((250-100)*np.random.rand()+100),0)
    cy = np.round(((250-100)*np.random.rand()+100),0)
    track = np.zeros([500,500],float)
    srim_pixels = copy.copy(srim_track)
    srim_pixels[:,2:4] = srim_pixels[:,2:4]*(0.0001)*(1/16)
    srim_pixels[:,2:4] = np.floor(srim_pixels[:,2:4])
    srim_pixels[:,2] = srim_pixels[:,2]-srim_pixels[1,2] + cx
    srim_pixels[:,3] = srim_pixels[:,3]-srim_pixels[1,3] + cy
    for line in range(1,np.shape(srim_track)[0]):
        x = srim_pixels[line,2]
        y = srim_pixels[line,3]
        track[y,x] = track[y,x]+srim_track[line,5]*
                \np.sqrt((srim_track[line-1,2]- \ srim_track[line,2])**2 \
                                        +(srim_track[line-1,3]- \
                                          srim_track[line,3])**2 \
                                        +(srim_track[line-1,4]-\
                                          srim_track[line,4])**2)
    hdu = pyfits.PrimaryHDU(track)
    hdu.writeto('/home/USER/SRIM/track' + str(energy) + \
                'keV' + str(angle) + 'Deg/track'  + str(ionNumber) + '.fits', \
                clobber = True)
    srim_matout = np.vstack((srim_matout,srim_pixels))
    srim_begend = np.vstack((srim_begend,[srim_pixels[1,2],srim_pixels[1,3],\
                                          srim_pixels[srim_pixels.shape[0]-1,2],\
```

```
                                        srim_pixels[srim_pixels.shape[0]-1,3]]))

srim_matout = np.delete(srim_matout,0,0)
srim_begend = np.delete(srim_begend,0,0)
np.savetxt('/home/USER/SRIM/Fits analysis/SRIMEnergy' \
           + str(energy) + 'keV' + str(angle) + 'Deg.csv', srim_matout, delimiter=",")

np.savetxt('/home/USER/SRIM/Fits analysis/SRIMBegEnd' \
           + str(energy) + 'keV' + str(angle) + 'Deg.csv', srim_begend, delimiter=",")
```

```
# srimReader.py

'''
Created on Mar 14, 2015

@author: Peter Nary
'''

import sqlite3
import numpy

class srimParser:
    __dbPath = ''
    __badIons = []
    def getIonList(self):
        with sqlite3.connect(self.__dbPath) as conn:
            cursor = conn.cursor()
            ionList = []
            for datarow in cursor.execute("SELECT DISTINCT ionNumber
              FROM Calculations ORDER BY ionNumber"):
                ionList.append(datarow[0])
        return ionList;

    def __getDataRow(self, line):
        elements = line.split()

        for x in range(0, len(elements)):
            item = elements[x]
            if (item[len(item)-1] == "-"):
                elements[x]=item[0:(len(item)-2)]

        ionNumber = int(elements[0])
        kev = float(elements[1])
        xAngstrom = float(elements[2])
        yAngstrom = float(elements[3])
        zAngstrom = float(elements[4])
        eva = float(elements[5])
        ev = float(elements[6])
        return (ionNumber, kev, xAngstrom, yAngstrom, zAngstrom, eva, ev);

    def __addLinesToSqlite(self, textLines, dbfilename):
        rows = []
        for line in textLines:
            try:
                rows.append(self.__getDataRow(line))
```

```
            except ValueError:
                self.__badIons.append(line.split()[0])
            except IndexError:
                self.__badIons.append(line.split()[0])

        with sqlite3.connect(dbfilename) as connection:
            cursor = connection.cursor()
            cursor.executemany('''INSERT INTO Calculations (ionNumber, kev, xAngstrom,
              yAngstrom, zAngstrom, eVA, eV) VALUES (?, ?, ?, ?, ?, ?, ?);''', rows)
        return;

    def __createDB(self, textFileName):
        dbFileName = textFileName.replace('.txt', '.s3db')

        with sqlite3.connect(dbFileName) as connection:
            cursor = connection.cursor()
            cursor.execute('''DROP TABLE IF EXISTS Calculations''')
            cursor.execute('''CREATE TABLE IF NOT EXISTS Calculations (CalcID
              INTEGER PRIMARY KEY, ionNumber INTEGER, kev REAL, xAngstrom REAL,
              yAngstrom REAL, zAngstrom REAL, eVA REAL, eV REAL)''')
        return dbFileName;

    def __convertTxtToSqlite(self, textFileName):
        fileReader = open(textFileName)
        dbFileName = self.__createDB(textFileName)
        currentLine = 0
        batchsize = 500000
        currentBatch = 0
        lineBatch = []
    #     with fileReader as sys.open(textFileName):
        for textline in fileReader:
            if currentLine > 15:
                lineBatch.append(textline)
                currentBatch +=1
                if (currentBatch > batchsize):
                    self.__addLinesToSqlite(lineBatch, dbFileName)
                    currentBatch = 0
                    del lineBatch[:]
            else:
                currentLine +=1

        if (len(lineBatch) > 0):
            self.__addLinesToSqlite(lineBatch, dbFileName)

        if len(self.__badIons) > 0:

            with sqlite3.connect(dbFileName) as conn:
                cursor = conn.cursor()
                for ions in self.__badIons:
                    cursor.execute("DELETE FROM Calculations WHERE \
                    ionNumber={0}".format(ions))

        fileReader.close()
        return dbFileName;
```

```
    def __init__(self, srimPath):
        self.__dbPath = self.__convertTxtToSqlite(srimPath)
        return;

    def getIonValues(self, ionNumber):
        results = []
        with sqlite3.connect(self.__dbPath) as conn:
            cursor = conn.cursor()
            for datarow in cursor.execute("SELECT ionNumber, \
             kev, xAngstrom, yAngstrom," "zAngstrom, eVA, eV FROM Calculations \
             WHERE ionNumber=?", [ionNumber]):

                ion = int(datarow[0]) #ionNumber
                kev = datarow[1] #kev
                xangstrom = datarow[2] #xAngstrom
                yangstrom = datarow[3] #yAngstrom
                zangstrom = datarow[4] #zAngstrom
                eva = datarow[5] #eVA
                ev = datarow[6] #ev
                results.append([ion, kev, xangstrom, yangstrom, zangstrom, eva, ev])

        return numpy.array(results);
```

Once the appropriate tracks have been drawn into fits image files by this Python code, they may be opened and manipulated.

B.2.3 Creating Track Images from Nuclear Recoil Tracks

Once we have our tracks on a zero background we may begin to simulate the effects of the detector. This is done by loading the track image into a matrix in matlab if it is a fits file, like the SRIM data from python are.

Once the image is in a matrix, it can be convolved with a two dimensional Gaussian distribution of width $\sigma = 350\mu m$ This corresponds to a pixel size of 21.875. After that we rebin the image from 500 by 500 to 50 by 50, which results in each pixel being reduced in intensity by a factor of 100. To offset this we then multiply the rebinned distribution by 100. We then insert the new 50 by 50 matrix into a random location in a 170 by 174 matrix of zeros.

Finally we add an appropriate noise distribution to the image. The noise we add has a mean of 0 and a standard deviation of 10 ADUs. Once this is completed we may save our image as a fits file and we have completed the simulation of the detector. The code for this process is immediately below.

```
% Either open your image file from the SRIM simulation by running the following
% line:
% before = fitsread('TRACKNAME.fits')

% Or create a straight track by running one of the following lines:

% before = shallowtrack(energy, length, skew, angle, inc)
% before = steeptrack(energy, length, skew, angle, inc)

L = 50;

kernel = fspecial('gaussian',[500,500],21.875);

before = fitsread(['/home/USER/Matlab/Research - drift/SRIM/track' ...
    num2str(energy) 'keV' num2str(angle) 'Deg/track' num2str(i) '.fits']);
before(before > 10^10) = 0;
before(before == Inf) = 0;

% This ensures the track has the correct scaling in ADUs.
before = (250/1000)*before;

after_conv=conv2(before,kernel,'same');
after_res=imresize(after_conv,[L L]);


CC = clock;
rng(round(10000*(CC(6))),'twister');
T = zeros(170,174);
locx = round(119*rand) + 1;
locy = round(119*rand) + 1;
T(locx:(locx-1+L),locy:(locy-1+L))=after_res;
T = T*100;
% % T(find(T<0)) = 0;
noise=normrnd(0,10,[170 174]);
SimulatedTrack=T+noise;
```

At the end of this code we are left with a matrix called SimulatedTrack. This is our simulated image, it can be viewed as it is, or saved as a fits image to be used in analysis later.

This completes the methods for simulating tracks.

Bibliography

[1] F. Zwicky. Republication of: The redshift of extragalactic nebulae. *General Relativity and Gravitation*, 41:207224, November 2008.

[2] Planck Collaboration: P. A. R. Ade, N. Aghanim et al. Planck 2013 results. I. Overview of products and scientific results. *Astronomy and Astrophysics*, June 2014.

[3] M. Pipe. Limits on spin-dependent WIMP-proton cross-sections using the DRIFT-IId directional dark matter detector. Dec 2011. PhD Thesis, University of Sheffield.

[4] M. Boylan-Kolchin et al. A.D. Ludlow, J.F. Navarro. The mass profile and accretion history of cold dark matter haloes. *MNRAS*, 432:1103-1113, June 2013.

[5] J.D. Lewin and P.F. Smith. Review of mathematics, numerical factors, and corrections for dark matter experiments based on elastic nuclear recoil. *Astropart.Phys.*, 6:87–112, 1996.

[6] Sheffield DRIFT-Cygnus Group, April 2015.

[7] J.F. Macas-Prez J. Billard, F. Mayet and D. Santos. Directional detection as a strategy to discover Galactic Dark Matter. *Physics Letters B*, 691:156162, July 2010.

[8] Ben Morgan, Anne M. Green, and Neil J. C. Spooner. Directional statistics for realistic weakly interacting massive particle direct detection experiments. *Phys. Rev. D*, 71:103507, May 2005.

[9] N. Phan. Personal Correspondence, April 2015.

[10] A. Hitachi. Bragg-like curve for dark matter searches: Binary gases. *Radiation Physics and Chemistry*, 77:1311–1317, 2008.

[11] M.K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, pages 179–187, 1962.

[12] Dan Kernler. Empirical rule.png, April 2015.

[13] E. Bertin and S. Arnouts. SExtractor: Software for source extraction. *Astronomy and Astrophysics, Supplement*, 117:393–404, June 1996. Provided by the SAO/NASA Astrophysics Data System.

[14] J. Ziegler. The Stopping and Range of Ions in Matter (SRIM), 2015.