NTNU

Norwegian University of
Science and Technology

Charge Diffusion Modelling for a Monolithic Active Pixel Sensor Detector with Application to Proton CT

Even Hunnes Helgesen
Hansen

Master of Science in Physics and Mathematics
Submission date: February 2017
Supervisor: Pål Erik Goa, IFY
Co-supervisor: Dieter Röhrich, Universitetet i Bergen

Norwegian University of Science and Technology
Department of Physics

Abstract

A model for charge diffusion in a monolithic active pixel sensor detector was studied with application to a digital tracking calorimeter, proposed for use in proton computed tomography. The model was implemented in C++, using the ROOT library. The resulting detector response yielded clusters of activated pixels with similar shapes as clusters from experimental data. The cluster sizes showed discrepancies at higher values of energy deposition, but similar cluster sizes for lower values of energy deposition. The resulting energy-size relationship for MIMOSA-23 sensor chips was $n = 7.6626 \cdot (E_{dep})^{0.420307}$, assuming attenuation length λ =45 µm. The charge diffusion model also produced results for 30 MeV protons incident to a pALPIDE chip, where the mean cluster size was similar to the experimental values.

Sammendrag

En modell for ladningsdiffusion i en monolittisk aktiv piksel sensor detektor ble studert for bruk i et digitalt tracking calorimeter, foreslått for bruk i proton CT. Modellen ble implementert i C++ med ROOT biblioteket. Den resulterende detektorresponsen ga klynger av aktiverte piksler med lignende fasonger som klyngene fra eksperimentelle data. Klyngestørrelsene viste avvik ved høyere verdier for energideponering, men lignende størrelser for lavere verdier av energideponering. Den resulterende energi-størrelse sammenhengen for MIMOSA-23 sensor chiper var $n = 7.6626 \cdot (E_{dep})^{0.420307}$, antatt at dempningslengden var λ =45 µm. Modellen for ladningsdiffusjon produserte i tillegg resultat for 30 MeV protoner mot en pALPIDE chip, hvor gjennomsnittlig klyngestørrelse samstemte med eksperimentelle verdier.

Preface

The following material is the results of the research for my master thesis in subject TFY4900 at NTNU. It is a study upon the charge diffusion which occur within the epitaxial layer of a monolithic active pixel sensor, motivated by its application to proton CT. The first chapter will act as a motivation and introduction to the concept of imaging with proton beams. The second chapter will introduce the theory related to the charge diffusion and the methods used for analysis. The third chapter contains the results from the implementation of the program code and show plots related to analysis of feasibility of using the model. The fourth chapter contains discussion. The fifth chapter contains the conclusion and suggestions for future work.

My project thesis [1] has been utilized in this report. Chapter 2 uses some of the same theory as the project thesis. Within section 2.1.1 and 2.4 a copy of some of the text has been applied.

The analysis in chapter 3.2 and 3.4 has made use of an existing program code [2], written by Ph.D candidate Helge Pettersen at Helse Vest.

Trondheim, 2017

Even Hansen

Acknowledgement

There are a lot of people who have been of great help for this report. I am thankful for the fantastic help I have received; I don't know how I could have done this without them.

Firstly, I would like to thank my supervisor Dieter Röhrich for excellent guidance and positive spirit. Visiting Dieter at the University of Bergen has been a great educational experience, and I am very thankful for the travel.

I would also like to thank my supervisor at NTNU, Pål Erik Goa, for great guidance through my project thesis and continuous support throughout my master thesis. It has been a great aid to receive continuous feedback on my work.

I want to thank Helge Pettersen for excellent help, great conversations and fantastic guidance throughout the entire work performed in relation to the master thesis. I am certainly overwhelmed by the large, informative and positive responses I have received from the questions I have asked him. In addition, I would like to thank him for inviting me to use his program code. The help he has provided is nothing short of amazing, especially since a lot of it has been during his paternity leave.

Also, thanks to my good friend Einar Karlsen for providing feedback on my report.

Lastly, I would like to thank my dearest Sunniva Skogseth. She keeps me motivated and positive and is a huge contribution to helping me achieve my goals. It means a lot to me.

E. H.

Contents

Acronyms and Abbreviations

CT Computed Tomography

DTC Digital Tracking Calorimeter

LET Linear Energy Transfer

MAPS Monolithic Active Pixel Sensor

MC Monte Carlo

MCS Multi Coulomb Scattering

pCT Proton Computed Tomography

RSP Relative Stopping Power

WEPL Water Equivalent Path Length

WET Water Equivalent Thickness

List of Figures

List of Tables

Chapter 1

Introduction

Already in 1946, Robert Wilson published a paper which discussed the use of accelerated protons and heavy ions for a precise treatment of tumors, utilizing the narrow peak in energy deposition of the stopping particles [5]. This could lead to a precise treatment of cancer tumors while minimizing the damage to the surrounding tissue. Shortly after, Scientists at the Lawrence Berkely Laboratory initiated studies to validate the hypothesis [6].

Proton beam radiotherapy is an increasingly popular treatment for the eradication of tumors in patients. As of January 2015, more than 137 000 patients have been treated with charged particle therapy [7]. The treatment utilizes proton beams, accelerated by a cyclotron or synchrotron, to deliver an energy dose to tumors in order to eradicate the mutated cells. By adjusting the proton beams to specific energy levels, the treatment can act upon a precise volume while leaving the surrounding healthy tissue intact.

Proton therapy needs an accurate dose plan to achieve its goal. Currently, computed tomography (CT) based on x-rays is being used to generate three dimensional images of the structure with respect to radiodensity. To obtain the data on how protons react to the medium, the units of radiodensity has to be converted to proton stopping power, a calculation which introduces uncertainties, such that the resulting uncertainty of the proton penetration depth is in the range of 2-3% [2].

Proton Computed Tomography (pCT) uses the same imaging principle as a conventional CT, but substitutes protons for x-rays. Here, one can directly obtain the proton stopping power by measuring the energy as it leaves the proton accelerator and the remaining energy after the

proton has passed through the patient. This will increase the accuracy of proton therapy and reduce the probability of damage to healthy tissue or secondary malignancies.

A prototype pCT is currently being developed by a group at the University of Bergen, using a Digital Tracking Calorimeter (DTC), containing an array of Monolithic Active Pixel Sensor (MAPS) detectors, sandwiched between layers of absorption material. The detector aims to use the well known effects of the protons interactions with medium along with a detection of trajectory to calculate the properties of the object the proton passed through. By combining a large number of observations from protons, one can create three dimensional maps of the Relative Stopping Power (RSP) in the object. The ultimate goal of pCT is to increase the accuracy of dose planning for a proton therapy, motivated by previous experiments which show that the uncertainties of proton penetration depth can be reduced down to 1 % [8].

There are a number of advantages for pCT [9, 8]. Among them are:

- Lower dose for a given density resolution.
- Lack of beam hardening artifacts.
- Improved dose estimation for proton therapy.

However, the drawbacks of pCT include:

- Expensive and bulky beam sources compared to x-rays.
- Proton energies larger than 300 MeV are needed for scans of adult abdomen and pelvis.
- Multi Coulomb Scattering (MCS) reduces the spatial resolution.

There are methods for producing tomographic images of MCS utilizing pCT. However, this is outside of the scope for this report.

1.1 Background

In a pCT, a computer program uses the location in sensor chips where the proton passed through to generate tracks, finding an approximation to the trajectory through the detector. In order to

detect the remaining energy, previous pCT-detectors have used scintillators or other residual energy range detectors.

The DTC consists of alternating layers of MAPS and absorption material, which are used to track the protons after they have passed through an object. Using an existing program code written by Ph.D candidate Helge Pettersen [2], analysis on the DTC can be performed. The program code can use data obtained from Monte Carlo (MC) simulations or proton beam experiments to track the range of protons and observe the energy deposition in the sensor chips.

As the proton traverse each MAPS, a cluster of pixels are activated, due to an electron shower generated within the sensor chip. The size of this cluster is related to the energy deposited in the epitaxial layer of the chip, hence this value can be used to measure the remaining energy in the proton. By calculating the energy along the path, it is possible to fit a curve to the data, representing Linear Energy Transfer (LET), which further allows accurate range calculations for the protons.

An accurate model for the cluster size with respect to deposited energy in the detector has not been created. Currently a purely empirical model for cluster reconstruction has been implemented, using a two-dimensional Gaussian curve with increasing standard deviation for higher values of deposited energy. Though the empirical model fits the data from the experimental setup, it is advantageous to implement an analytical model based on the concept of the physics involved to attain higher precission.

1.2 Problem Formulation

An analytical model, based on the concepts of the physics involved for charge diffusion in a MAPS, has not yet been implemented in the existing code for the DTC [2]. In the project thesis, a model for the charge diffusion was implemented, using MATLAB to create and analyze clusters at energy levels corresponding to the energy deposit in MAPS of the experimental setup [1].

In the following project, the charge diffusion model will be implemented in C++ using the ROOT library with application to the existing code. The code should be optimized to a degree such that it can be used to simulate large amounts of clusters from MC simulations.

The diffusion model should result in a parametrization of the relationship between cluster

size and proton energy, such that a formula for calculating energy from clusters in experimental setups is created.

In addition, the model should be tested for the available data from a new type of MAPS, the pALPIDE chip, proposed for use in a future prototype of the DTC.

Chapter 2

Theory and Method

The content of this chapter is the theory relevant to the charge diffusion modelling and the method of implementation for the program code.

2.1 Interactions with the Medium

As a proton passes through a medium, there are several factors which will cause the particle to deposit energy and change the trajectory. The three interactions which proton CT is concerned with are the gradual reduction of kinetic energy by ionization of the surrounding medium, deflections of the proton due to the electrical forces in the Coulomb potential and inelastic nuclear interactions which attenuates the proton flux from the beam. The three interactions come together to form the characteristic shape of the Bragg peak.

2.1.1 Stopping

The rate at which a proton loses energy through ionizing forces of a medium is known as the stopping power. This process is a result of the electromagnetic force acting between the proton and an atomic electron [10]. This is expressed in the Bethe-Bloch formula, which in the classical form with relativistic corrections becomes [11]:

$$S = -\frac{dE}{dx} = \left(\frac{ze^2}{4\pi\epsilon_0}\right)^2 \frac{4\pi Z\rho N_A}{Am_e v^2}\left[\ln\left(\frac{2m_e v^2}{I}\right) - \ln(1-\beta^2) - \beta^2\right]. \tag{2.1}$$

Here, S is the stopping power, given in energy per unit length, E is the energy, z is the charge of the incident particle, e is the electron charge, m_e is the electron mass, ϵ_0 is the vacuum permittivity, Z, A and ρ is the atom number, the relative atomic mass and the density of the medium, respectively. N_A is the Avogadro number, v is the particle velocity and $\beta = v/c$, where c is the speed of light.

The classical formula is derived by taking into account the momentum transferred to the electrons as the proton passes by. The result is that the stopping power is to the first order proportional to the square of the transit time, $S = 1/v^2$. From this, it follows that the proton deposits an increased amount of energy per unit length further into the range. Near the end of the trajectory, the energy deposit reaches a maximum, creating a narrow peak. At this position, the proton comes to rest and the energy deposition abruptly drops. This can be seen as a narrow peak in the so-called Bragg curve. When only the local energy deposit by ionizing force is considered, the stopping power translates to Linear Energy Transfer (LET).

The stopping power is the average energy loss per length unit in a medium, caused by a large number of electromagnetic processes. However, for each position along a proton's trajectory, the rate of energy loss follows a Landau distribution [12]. This distribution has a longer tail towards higher energy loss, compared to a Gaussian distribution. If a large amount of energy is transferred to an electron this way, the electron could travel some distance and ionize along its trajectory. However, this effect is neglected for the purposes of diffusion modelling in this report.

2.1.2 Coulomb Scattering

A proton travelling through a medium is subject to electromagnetic Coulomb forces from the nucleus, resulting in a small angular deflection. In most cases the angular deflection from a single nucleus is extremely small. However, a large number of random deflections can cause a measurable angular variation, known as Multi Coulomb Scattering (MCS) [13].

The spatial distribution of angular scattering is approximately Gaussian distributed where the angular deflection is typically very small. For the relevant energy range it can be up to 16°, but usually only a couple degrees [10].

In proton CT with a DTC, it has been shown in previous work that the angular deflection has

a low impact on the MAPS response [1].

2.1.3 Nuclear Interactions

In rare cases, the proton undergoes nonelastic nuclear interactions with the medium. This results in one or more secondary particles knocked off the nucleus. The resulting particles, including the primary particle, tend to have very different characteristics than the incident particle. Just downstream from the reaction, a large energy deposition occurs.

For the tomographic purposes discussed in this report, the nuclear interactions do not contribute to an interesting result. However, the attenuation of protons through nuclear interactions are applied in other types of CT [14]. For protons at therapautic energy levels, approximately 1 % of the protons undergo nuclear interactions per centimeter. For 160 MeV protons, approximately 20 % undergo nuclear interactions before coming to rest. For the type of tomography discussed in this report, the secondary particles will not be followed and are removed from the results.

2.2 Proton Computed Tomography

The principle of pCT is closely related to the principle of conventional X-ray CT. X-ray CT is an imaging technique in which the goal is to achieve a three dimensional projection of the scanned structure with respect to the radiodensity, measured in Hounsfield Units (HU). The technique provides an efficient method to study internal structures of patients by differentiating between the different types of tissue.

For the dose planning of a proton therapy treatment, the goal is to find a Water Equivalent Path Length (WEPL) for which the proton needs to travel to reach the target by considering the RSP in the tissue along the path. This cannot be directly calculated from HU, but must be calculated by pre-determined HU-to-RSP conversion curves [15].

In a pCT, protons substitute the x-rays in the image technique of a conventional CT. In medical use, protons will be accelerated to sufficient energy for the protons to pass completely through the patient, i.e. energies above 200 MeV [13]. By collecting the protons in a detector, the residual energy and direction will be used to estimate the loss of energy along each proton's

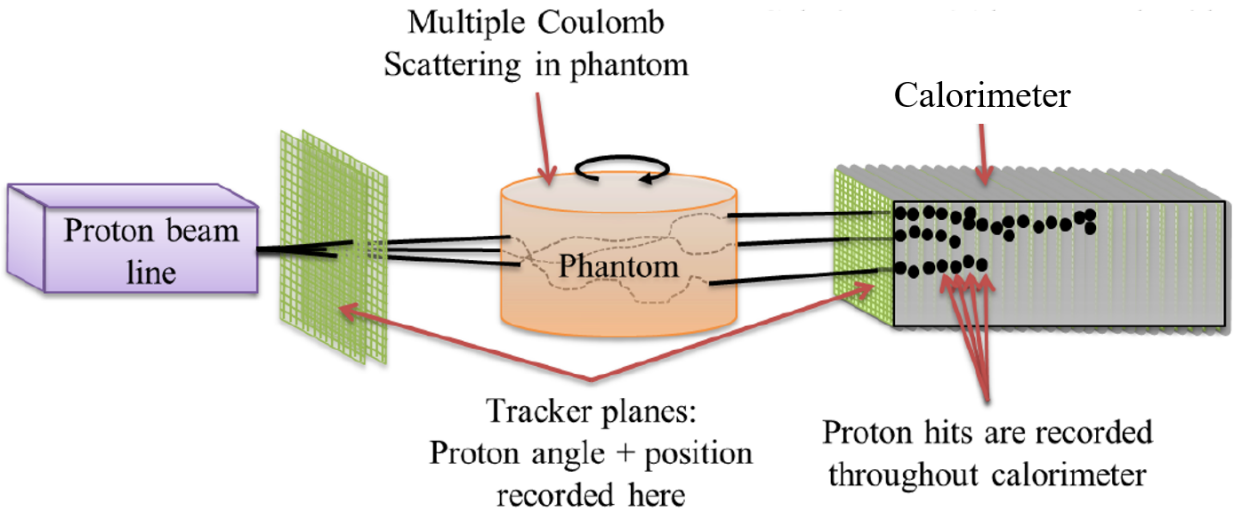

Figure 2.1: The setup for the proposed proton CT apparatus. The protons are accelerated before released through a tracker plane consisting of two layers of MAPS. The proton continues through a phantom, where it is subject to interactions with the medium. Lastly, the proton travels into a detector where the hits are recorded.
(The figure was obtained from Helge Pettersen.)

path. This data is used as basis for reconstructing a map of relative stopping power.

While a proton therapy requires that the protons come to rest within the patient, specifically at the tumor, a pCT requires that the protons pass through the patient and enter the detector. Since the stopping power is related to the velocity, and thereby the energy of the proton, the value of the stopping power is inappropriate for calculating WEPL for therapeutic energy. However, the ratio between stopping power at a point relative to water is approximately constant with energy and its slow variation is well understood [13, 15].

The main limiting factor for pCT spatial resolution is the angular variations caused by MCS. Unlike photons, protons are deflected by electromagnetic forces, causing deviations both in angle and lateral position. By increasing initial energy, the deflection would be reduced at the cost of reduced energy contrast [13]. To mitigate the error from the deflections, the most likely path for each proton is estimated by using the angle and position at entry and exit of the patient.

2.2.1 Proton Detector

Due to the requirements of a pCT, including information on the path and residual energy for the proton, existing commercial detectors are not alone suitable for the tomography. Several research groups are currently developing prototype systems for proton tracking with application to pCT. A current prototype is described in a review [13], using position-sensitive detectors both in front and behind the object in order to obtain the path, and residual energy-range detector distal to the beam source in order to obtain the total disposed energy.

The prototype focused on in this report is a high-granularity DTC, which is currently being developed by a group in Bergen [2]. The design consist of multiple alternating layers of MAPS and absorption material of either tungsten or aluminium. In the current models, there are 24 layers of detectors, where each detector layer consist of 2×2 MIMOSA-23 chips, sandwiched between tungsten absorber plates. The setup is shown in figure 2.1.

The setup for the beam measurements used in this report were performed in 2014 at the Kernfysisch Versneller Instituut in Groningen, the Netherlands. The measured beams were between 150 MeV and 188 MeV. The proton beams with energy 188 MeV will be the main focus for the analysis performed in the report.

2.3 Monolithic Active Pixel Sensors

There are two types of MAPS chips which are proposed for use in the detector, MIMOSA-23 and pALPIDE.

The detectors consist of an array of binary photodiodes. Each sensor corresponds to a single pixel in the chip and are individually activated if charges above a threshold is collected within the shutter readout. The layer of sensors is connected to an epitaxial layer covered in a substrate, creating an electrical potential barrier. MIMOSA-23 and pALPIDE have different specifications, as shown in table 2.1.

A proton traversing the epitaxial layer in a MAPS deposits energy to the electrons, as described in equation (2.1). The energy is transferred to excitation of electrons such that a large number of electron-hole pairs are generated along the proton's trajectory. The electrons diffuse in the epitaxial layer before either being attenuated or reaching the sensor layer.

Table 2.1: Specifications for the MIMOSA-23 and pALPIDE sensor chips. The information is collected from [3, 4].

Type	MIMOSA-23	pALPIDE
Area [mm^2]	19.2×19.2	15×3
Pixels	640×640	512×1024
Pixel size [µm^2]	30×30	28×28
Epitaxial layer [µm]	14	18

The number of electrons generated in the epitaxial layer is calculated by the work function, Φ_i. The work function describes the energy needed to remove an electron from the material i to a position immediately outside of the surface. By assuming that the entire energy deposit of the proton is converted to generating electron-hole pairs, we obtain the number of free electrons:

$$N_{e^-} = \frac{E_{dep}}{\Phi_i} \tag{2.2}$$

where N_{e^-} is the number of electrons, E_{dep} is the energy deposit and Φ_i is the work function.

2.3.1 MIMOSA-23

The MIMOSA-23 chip is the currently utilized MAPS in the prototype of the calorimeter proposed for use in a proton CT [16]. The chip uses rolling shutter, where the data being sent are binary signals for each pixel with a read-out time of 640 µs for a frame.

In this report, the MIMOSA-23 chip will be the main focus for cluster analysis, as the chip was used in the experimental data for the calorimeter setup.

2.3.2 pALPIDE

The pALPIDE chip is a prototype pixel detector proposed for the upgrade of the ALICE Inner Tracking System during the second long shutdown of LHC in 2019-2020. The development of pALPIDE aims to reduce the power dissipation in the experiment by introducing a different read-out concept. The chips make use of a deep p-well, which allows for integrated circuits within the pixel matrix while retaining full charge collection efficiency [4, 17].

The design of the pALPIDE chip has allowed the read-out concept to move away from the rolling shutter, as used in MIMOSA-23, to a method using an in-matrix sparsification circuit

which sends only the addresses of the hit pixels to the periphery [4].

The pALPIDE sensor can be biased, such that a bias voltage causes a drift in the electron collection. This can be used to adjust the sensor response, which would be observed as a change in cluster size. Turning off bias voltage causes the electrons to spread by diffusion.

The concept of charge diffusion in pALPIDE is similar to the diffusion which occurs in MIMOSA-23, and it is therefore believed that a similar model can hold for both. A thicker epitaxial layer means that there could be a larger contribution of electrons to the individual sensors, creating a larger area of diffusion for the detector.

2.4 A Model for Charge Diffusion in MAPS

For the MC simulations of the pCT, it has been determined that it is not feasible to follow the individual free electrons generated within the epitaxial layer. The large number of moving electrons would have to be simulated to undergo a large number of interactions while moving through the medium. Due to this, it is advantageous to instead approximate the incident electrons on the detector surface through diffusion models.

In the current iteration of the DTC analysis code, the energy-cluster size relation is modelled from the cluster sizes generated with the technique described in section 3.2. Helge Pettersen et al [2] states that "The model described [...] was used to create a parametrization for the estimation of the amount of deposited energy, given in keV, from the number of activated pixels, n, in a cluster. A polynomial fit to a large number of modelled clusters was used for this purpose:

$$E_{dep} = f(chip)[-4.0 + 3.88n + 1.24 \cdot 10^{-2} n^2 - 1.14 \cdot 10^{-3} n^3 - 1.42 \cdot 10^{-6} n^4]." \tag{2.3}$$

Here, $f(chip)$ is a parameter related to the sensitivity of the chip.

A simple model for charge diffusion in a MAPS was proposed in [3]. The model makes use of the linear energy transfer and assumes that the entire deposited energy is used to generate electron hole pairs along the trajectory. Furthermore, it is assumed that the electrons diffuse isotropically from each traversed point of the proton trajectory. The electrons that diffuse away from the epitaxial layer is reflected on the substrate. Hence, the free electrons that don't move

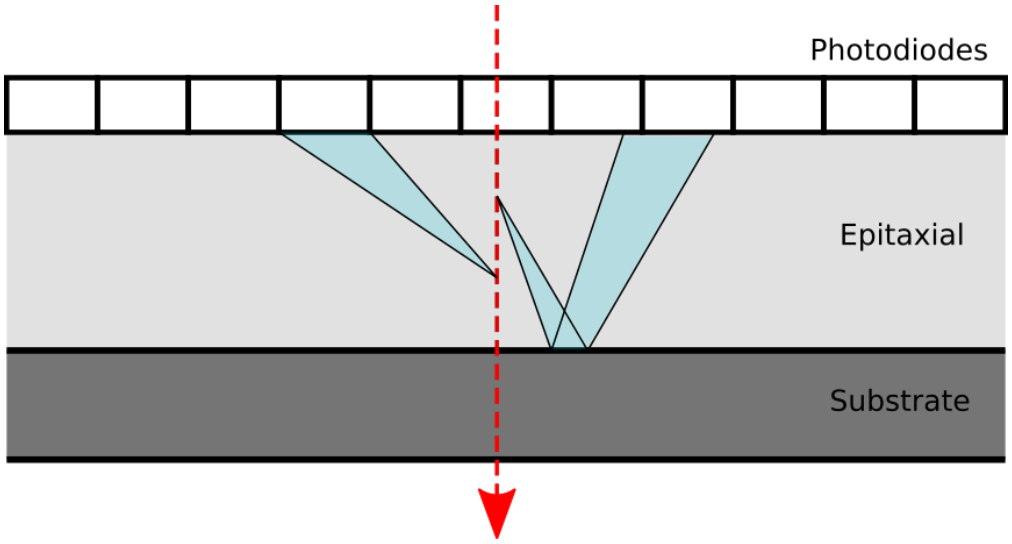

Figure 2.2: A schematic cross section of a MAPS. The proton traverses the epitaxial layer along the red dashed line. From each point along the trajectory, electron-hole pairs are generated and diffused. The electrons moving towards the substrate are reflected. The ratios in this figure do not represent the real MAPS, and are chosen for describing the diffusion concept. In reality, the width of a single pixel is more than twice as large as the depth of the epitaxial layer.
(The figure design is courtesy of [3].)

towards the sensor layer cause a contribution to the total charge on each pixel as seen in figure 2.2. A derivation for the model is shown in appendix A. Probability distribution for a charge in position P to reach the point M on the detector surface becomes (with $\vec{R} = \vec{PM}$):

$$\rho(\vec{R})\mathrm{d}r\mathrm{d}\phi = \frac{\mathrm{d}\Omega}{4\pi}\cdot\exp\left(-\frac{|\vec{R}|}{\lambda}\right) = \frac{hr}{4\pi|\vec{R}|^3}\cdot\exp\left(-\frac{|\vec{R}|}{\lambda}\right)\mathrm{d}r\mathrm{d}\phi, \tag{2.4}$$

or similarly, in Cartesian coordinates

$$\rho(\vec{R})\mathrm{d}x\mathrm{d}y = \frac{h}{4\pi|\vec{R}|^3}\cdot\exp\left(-\frac{|\vec{R}|}{\lambda}\right)\mathrm{d}x\mathrm{d}y. \tag{2.5}$$

Here, λ is the attenuation constant, $\mathrm{d}r\mathrm{d}\phi$ as well as $\mathrm{d}x\mathrm{d}y$ are infinitesimal areas on the detector surface, h is the height of the electron over the detector surface and r is the length of the projection of $\vec{R}$ onto the detector surface. The electrons reflected of the substrate are given by the same equation and transforming h to $h' = 2l - h$ and $\vec{R}$ to $\vec{R}' = (r, \phi, h')$, where l is the thickness of the epitaxial layer.

The epitaxial layer of a MAPS is a thin (14 µm in MIMOSA-23) silicon layer which absorbs

some energy from a proton passing through it. When a high energy proton is collected in the DTC, the energy disposed in the epitaxial layer of the sensor chips, E_{dep}, don't contribute a significant amount to the total energy deposit, E_{tot}, in the DTC. Most of the proton energy is deposited in the absorption plates between the MAPS. Therefore, it can be assumed that $E_{dep} \ll E_{tot}$, which implies that the energy deposition per length is constant within the individual MAPS.

In a previous project [1], it was shown that the angle of the proton relevant for pCT did not contribute significantly to the sensor response. Furthermore a proton with a large angular deviation in its trajectory is discarded from further analysis, as it has a large uncertainty to its most likely path. The width of the sensor diodes is significantly larger than the depth of the epitaxial layer by a ratio 2.1 for a MIMOSA-23 chip.

Since each point in the proton's trajectory is considered an origin of diffusing charges, the entire diffusion model can be expressed as an integral over the proton depth:

$$\begin{aligned} I(x,y) &= \int_{h=0}^{l} \mathrm{d}h[\rho(|\vec{R}|) + \rho(|\vec{R'}|)] \\ &= \int_{h=0}^{l} \mathrm{d}h \frac{r}{4\pi} \left[\frac{h}{(r+h)^{3/2}} \exp\left(-\frac{-(r^2+h^2)^{1/2}}{\lambda} \right) + \frac{2l-h}{(r^2+(2l-h)^2)^{3/2}} \exp\left(-\frac{-(r^2+(2l-h)^2)^{1/2}}{\lambda} \right) \right], \end{aligned} \tag{2.6}$$

where $I(x,y)$ is the charge intensity on the sensor layer of the detector chip. With the assumption of one electron generated per unit of depth. For computational purposes, the integral becomes a sum of discrete units of depth, of size δh:

$$\begin{aligned} &I(x,y) = \\ &\sum_{h=0}^{l} \delta h \frac{r}{4\pi} \left[\frac{h}{(r+h)^{3/2}} \exp\left(-\frac{-(r^2+h^2)^{1/2}}{\lambda} \right) + \frac{2l-h}{(r^2+(2l-h)^2)^{3/2}} \exp\left(-\frac{-(r^2+(2l-h)^2)^{1/2}}{\lambda} \right) \right], \end{aligned} \tag{2.7}$$

Where δh is a unit chosen to be a small unit compared to the depth of the epitaxial layer.

Dividing the intensity $I(x,y)$ by the epitaxial layer thickness l yields a probability density function for hit position of a single electron diffused from the proton trajectory. However, for the

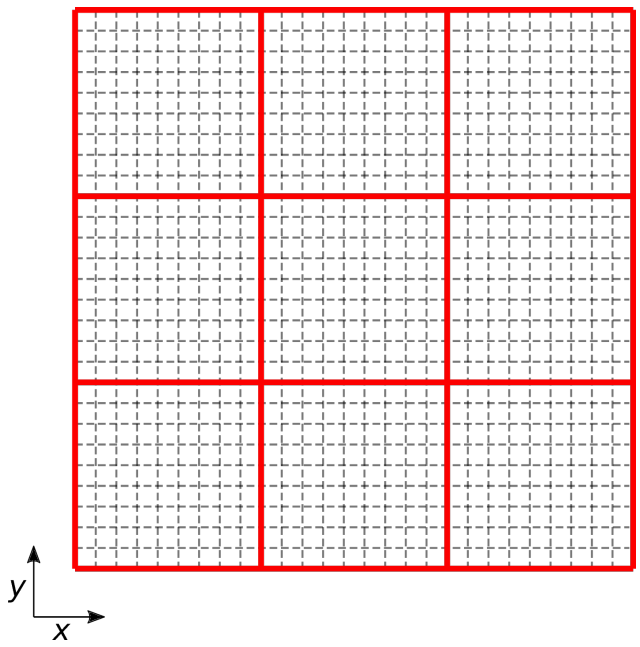

Figure 2.3: The figure shows a sample of the pixel sensor. The red lines outline the area of each pixel on the MAPS. The electron contribution to each individual pixel consist of the sum of electrons of each of the individual smaller areas, shown by the dashed gray lines.

purposes of the implementation of the function, it was chosen to not normalize the distribution.

2.5 Implementation of the Charge Diffusion Model

The charge diffusion code was written in C++ with the ROOT-library [18] to be integrated into an existing code [2] for further analysis.

2.5.1 Intensity Distribution

A pixel in the MAPS is activated if the pixel absorbs a number of electrons above some threshold. Thus, it is useful to create an intensity distribution, showing the expected number of incident electrons per unit area. Since the charge distribution is not equal on each position on the pixels, the pixel on the MAPS have to be divided into smaller areas, as shown on figure 2.3.

A matrix was generated to represent small areas ($9\,\mu m^2$) on the surface of the sensor surface. The size of the areas were chosen to be 1/9 the size of the pixels, such that a pixel on the detector would receive charges from $9^2 = 81$ elements in the matrix.

The elements of the matrix were calculated by using equation (2.7). The number of incident electrons on each sample area is dependent on the size of the area, such that we have

$$I_s(x_s, y_s) = \sum_{h=0}^{l} (\rho(\vec{R}) + \rho(\vec{R}'))\Delta x \Delta y, \tag{2.8}$$

where $I_s(x, y)$ is the number of electrons at sampled element in position (x, y). This distribution is chosen to have one free electron generated per length used in the summation. Since the distribution is similar for every proton energy, the number of electrons incident to each sampled element from a proton of arbitary energy deposition could be found by multiplying the entire distribution with the number of electron-hole pairs generated per length unit, N_{e^-}. This can be found using equation (2.1) and (2.2).

2.5.2 Cluster Generating Function

The cluster was generated by using the intensity distribution, grouping together matrix elements to form the total number of electrons incident on a pixel, such that

$$I(x, y) = \sum_{i=x_s}^{x_s+8} \sum_{j=y_s}^{y_s+8} I_s(i, j) N_{e^-}, \tag{2.9}$$

where $I(x, y)$ is the electron intensity at the pixel in position

A pixel was chosen to be activated if the $I(x, y)$ had a total of 25 charges, in agreement with previous findings [1].

A proton incident on the pixel sensor has an uniform distributed probability of traversing the epitaxial layer in any position, so we cannot assume that the proton moves through the center of the pixel. The location of incidence can have a large impact on the number of activated pixels. To include situations where the proton hits different positions, the function for grouping together elements was given an offset option.

When generating sensor response from MC-data, the hit location was evaluated by selecting a point within the pixel by a uniform random function. This could also create a realistic diversity of cluster shapes and sizes.

When analyzing the cluster sizes versus the deposited energy, each possible hit position was

evaluated with uniform probability, such that a distribution of cluster sizes could be evaluated. For the purposes of studying the deviation of cluster sizes for a single energy deposition value, the unbiased sample variance is used:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N}(n_i - \overline{n})^2, \tag{2.10}$$

where σ is the standard deviation, N is the number of clusters, n_i is the individual cluster sizes, and $\overline{n}$ is the mean cluster size.

2.6 Range and Energy Calculation in DTC

The range calculations were performed by using the existing program code for DTC [2]. This section describes some of the models and algorithms implemented in the program code.

2.6.1 Simulated Data

In order to calculate the range for a proton at a specific energy, MC simulations have been performed. The simulated setup was given similar material properties as the prototype of the DTC to find realistic range approximations. Protons with specific energy were simulated to enter the calorimeter and the reaction in each volume unit along the track was scored, using that the last energy deposit was assumed to be the terminus of the track. By observing a large number of protons and recording the terminus of each track, the penetration depth was found to follow a Gaussian distribution, as expected [19]. By performing the simulation for each energy in a predetermined range, a table for proton range as a function of energy was created.

2.6.2 Experimental Data

In order to obtain the range for a proton in an experimental setup, one cannot simply assume that the last observed energy deposition is the terminus of the trajectory. Since the DTC is layered with sensors chips and absorbing material, the setup would only yield discrete values for energies, corresponding to the layer number of the MAPS. In order to find the location where the proton came to rest, it is useful to observe the energy along the path.

By using the relation between cluster size and energy deposit along the path for a proton, one can observe the characteristic form of a Bragg curve; the energy deposit increases towards the end of the trajectory. The range calculation is preformed by fitting a predefined curve to the data. The equation for linear energy deposit for a proton travelling in a medium is [19]:

$$S(z) = -\frac{\mathrm{d}E}{\mathrm{d}z} = \frac{1}{p\alpha^{1/p}}(R_0 - z)^{(1/p-1)}, \tag{2.11}$$

where $S(z)$ is the linear stopping power at depth z, p and α are coefficients related to the empirical relationship between range and energy through Geiger's law, $R_0 = \alpha E_0^p$, and R_0 is the proton range. For water, it was used that p =1.7547 and α =2.2387 $\times 10^{-2}$ mm MeV^{-1}.

The range straggling of the proton beam is [19]

$$\sigma^2 = \alpha' \frac{p^2 \alpha^{2/p}}{3 - 2/p} R_0^{3-2/p}, \tag{2.12}$$

where α' is a factor that depends on the stopping medium. For water α' =0.087 MeV cm^{-1} [20]. As a consequence, the expected range straggling is $\sigma = 0.013767 R_0^{0.93010}$.

Equation (2.11) is fitted to the simulated data through fitting R_0 and by introducing a scaling factor k, which is multiplied with the entire function to adjust the equation for systematical error of cluster sizes along the trajectory.

A model for the relationship between energy and range for a proton beam in matter is [20]:

$$R_0 = a_1 E_0 \left[1 + \sum_{k=1}^{N=2} (b_k - b_k \cdot \exp(-g_k \cdot E_0)) \right], \tag{2.13}$$

where a_1, b_k and g_k are units that were determined in [2] by fitting the model to range-energy data, as well as the choice of $N = 2$. By obtaining the parameters for both the calorimeter and for water, an accurate method to calculate Water Equivalent Thickness (WET) was achieved. WET differs from WEPL by considering only the projected length of the proton, without considering the curved path.

An inverse of equation (2.13) is derived in [20], such that calculation of the energy E_0 at depth z can be calculated by:

$$E(z) = (R_0 - z) \sum_{i=1}^{N=5} c_k \exp[-\lambda_k (R_0 - z)], \tag{2.14}$$

where c_k and λ_k are coefficients found by fitting the model to the previously acquired range-energy data.

Chapter 3

Results

The charge diffusion model was written in C++, using the ROOT-library. The model was first applied to data acquired from a MC simulation, with constants related to MIMOSA-23, in order to visualize diffusion of the clusters. It was subsequently integrated in a previously written code [2] to analyze the model with the formerly acquired experimental data. Lastly, the model was adjusted to fit the constants related to a pALPIDE chip to compare to the available data.

3.1 Implementation of the Model

The implementation of the model consist of the free parameters shown in table 3.1.

In a previous report, the attenuation coefficient has been found to be λ =45 µm for a MAPS of type MIMOSA-5, which has similar characteristics as MIMOSA-23 [3]. This was used as the base line for generating the initial clusters. The hit position deviation was used to generate clusters of different shape which could affect the total cluster size. The work function was set to Φ_i = 4.85 eV [21].

The charge distribution intensity was generated from the new implementation and is shown

Table 3.1: By varying the parameters in the model, different cluster sizes and shapes are generated for protons at similar energy levels.

Parameter	Description	Main consequence
λ	Attenuation coefficient	Cluster size
Δx and Δy	Hit position deviation	Cluster shape

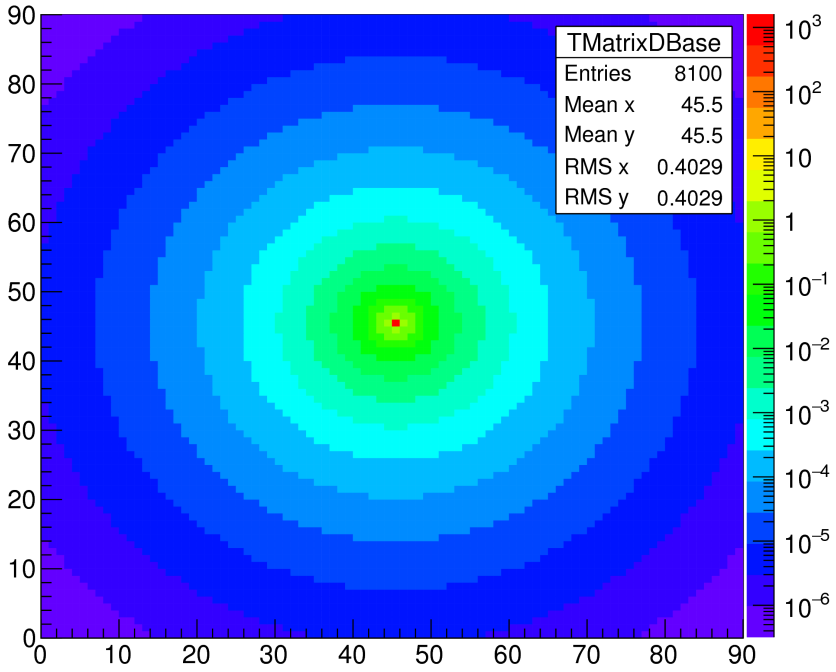

Figure 3.1: Intensity distribution for a incident proton on a sensor chip with an epitaxial layer thickness of 14 µm. *The intensity distribution assumes one electron-hole pair generated per micrometer. Each pixel on the sensor chip consist of* 9×9 *pixels on the intensity distribution, such that the width of each sampled pixel on the intensity distribution is* $30/9\,\mu\text{m} \approx 3.3\,\mu\text{m}$.

in figure 3.1. The distribution shows a sharp peak in the center, which is largely contributed by the electron-hole pairs generated right above the surface of the detector. The contribution of when the proton was directly on the sensor surface, $h = 0$ was discarded to avoid infinite values.

A one-dimensional representation is shown in figure 3.2, obtained by plotting the points across the middle row of the intensity matrix. The distribution shows again that there is a sharp peak in the center, but more importantly, that the intensity drops quickly as the position deviates from the peak.

Clusters were generated by using the intensity distribution, grouping together 9×9 sampled pixels and multiplying by the number of electron-hole pairs generated per micro meter.

The probability for each possible hit position within a pixel on the sensor chip is assumed to be uniformly distributed. Hence, 9×9 different events for a proton hit are equally probable. The parameters Δx and Δy takes into account the hit position, and its variations is shown in figure

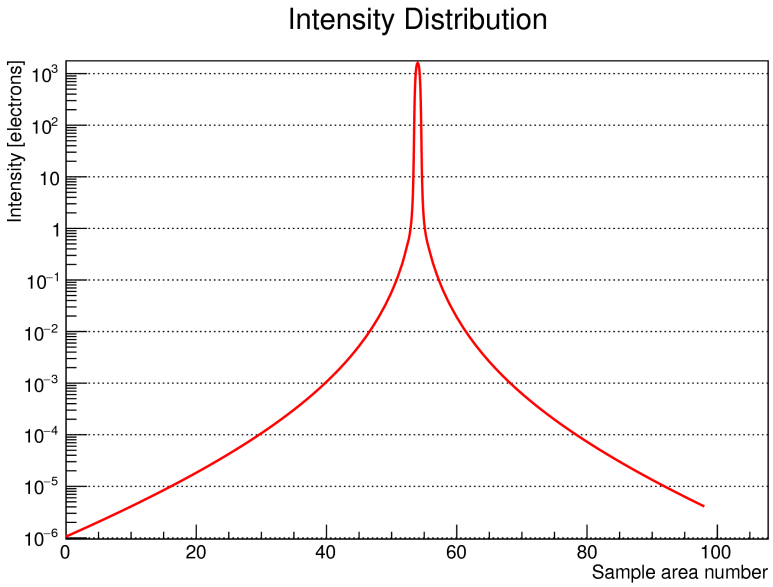

Figure 3.2: A 1D-representation of the intensity distribution on the sensor surface. The curve has a longer tail than a Gaussian distribution, such that a good fit function was not found.

3.3. The larger clusters have lower relative size variation.

3.2 Diffusing Cluster Frames

MC simulations have been performed to generate large data sets for further study. The simulations have used the geometry similar of an experimental setup. The sensor chip frames from the simulations does not provide a diffusion of the hits, but instead denotes hit positions along with a value for energy deposition for each hit. The goal of this section was to produce a model for diffusion of the proton hits in order to recreate a similar response which the MIMOSA-23 chips would produce. A sample of a frame from the experiment is shown in figure 3.4.

The previously implemented code for diffusing the frames was done purely empirically. The diffusion was performed by creating a two-dimensional Gaussian curve, centered at the origin pixel of the proton hit and with standard deviation σ set to a value proportional to the fourth root of the energy deposition E_{dep}. The pixels in the area around the origin were activated by randomly choosing pixels according to the distribution n times, where n is a number proportional to E_{dep}.

A comparison of the charge diffusion models is shown in figure 3.5.

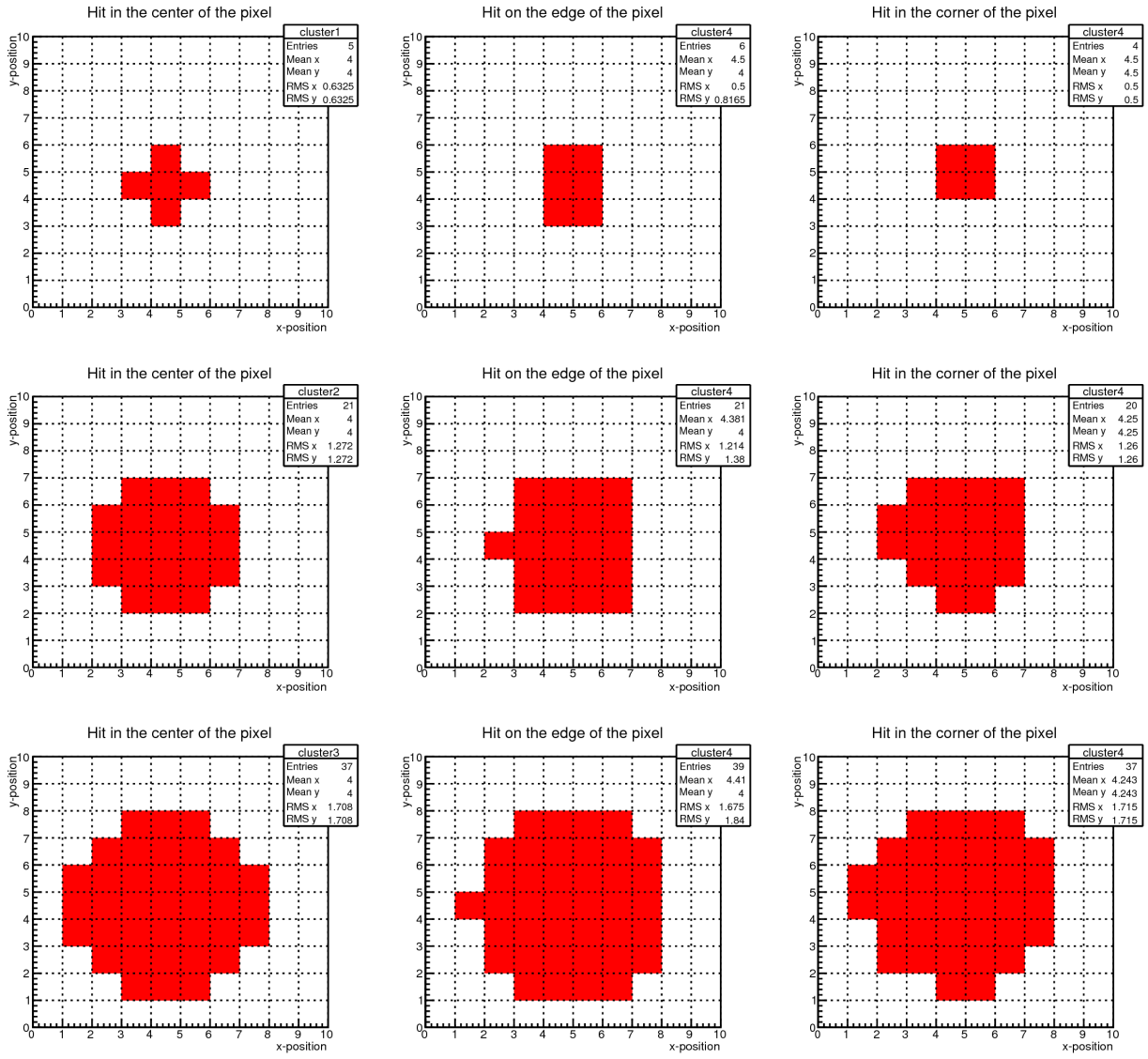

Figure 3.3: Clusters generated from the intensity distribution. The grid outlines the pixels on the sensor chip. The top row is generated from a proton depositing $0.7\,\mathrm{keV\mu m^{-1}}$*. The middle row is generated from a proton depositing* $10\,\mathrm{keV\mu m^{-1}}$*. The bottom row is generated from a proton depositing* $50\,\mathrm{keV\mu m^{-1}}$*. Each pixel is divided into a map of* 9×9 *possible hit positions. The left column has the hit position directly in the middle of the pixel. The center column has hit position on the left edge of the pixel. The right column has hit position in the top-right corner of the pixel.*

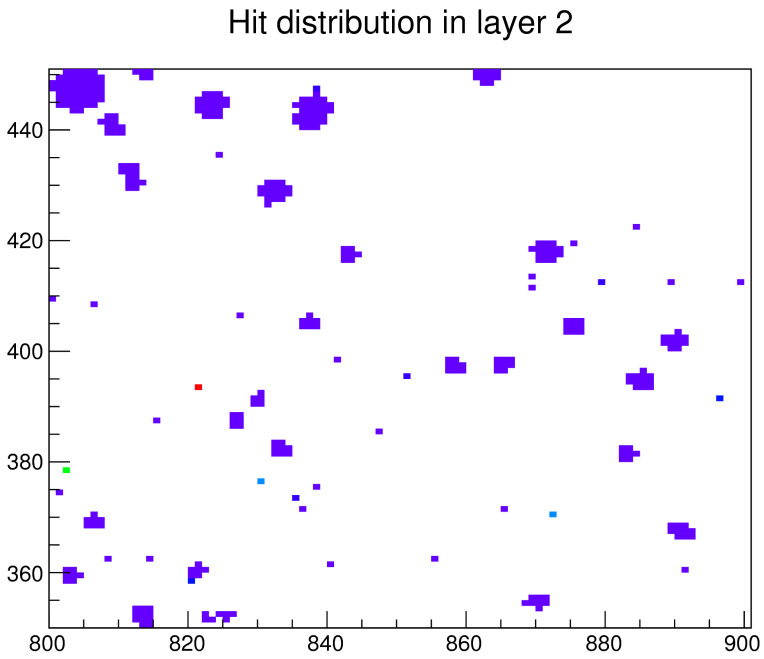

Figure 3.4: A part of the data frame from layer 2 in the experimental setup of the calorimeter. The incident proton beam has mean energy 188 MeV. Recreating both the cluster shapes and the sizes is the main goal of this section. The small (1 pixel) responses in the frame are considered to be noise, and are discarded from future analysis.

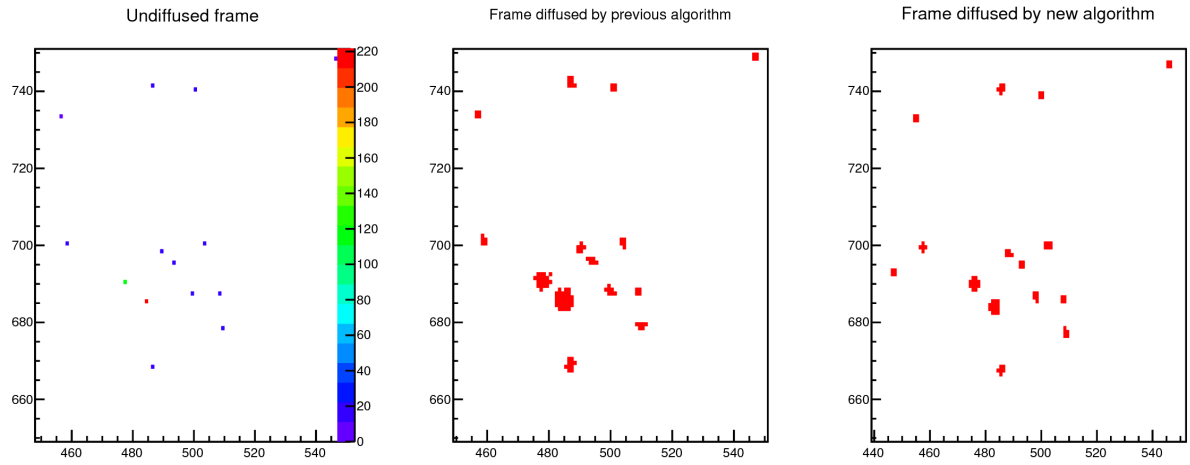

Figure 3.5: The figures correspond to a small section the second layer of pixel sensors in the MC simulation for a 188 MeV *proton beam.*
The figure on the left shows the resulting 2-D histogram of the MC simulation. The proton hits have not been diffused, but are assigned a value corresponding to E_{dep} for each hit. Since low energy particles are not followed in the MC simulation, each colored pixel represents a proton.
The middle figure shows the resulting frame after the empirical diffusion model.
The figure on the right shows the resulting frame after the new diffusion model.

3.3 Deposited Energy and Cluster Size Relation

The model described in equation (2.5) relates the cluster size to E_{dep} by multiplying the intensity distribution with the number of electron-hole pairs generated per micrometer. The deposited energy is related to the remaining proton energy through the Bethe-Bloch formula, equation (2.1). Here, data acquired from PSTAR [22] was used to obtain data points for stopping power. By fitting a simple curve to the data, the energy deposition was approximated:

$$E_{dep} \approx 43.95(E_{prot})^{-0748}. \tag{3.1}$$

Clusters were generated at different values of E_{dep} and plotted in figure 3.6. For each value of E_{dep}, each available cluster was created, thus a cluster for each of the 81 available hit positions within the origin pixel is represented. All possible cluster sizes were examined to find the empirical standard deviation for each energy. The cluster sizes were determined to depend on the deposited energy with the relation

$$\text{size} = p_0 \cdot (E_{dep})^{p_1}, \tag{3.2}$$

where p0 and p1 are parameters used for fitting the curve to the data. In the particular instance shown in figure 3.6, the clusters were generated using the baseline values for the model, resulting in $p_0 = 7.66260e+00$ and $p_1 = 4.20307e-01$. A comparison between this relationship and the relationship obtained from the empirical diffusion model is shown in figure 3.7.

3.3.1 Attenuation Coefficient

The charge diffusion model relies on a correct estimation for the attenuation coefficient, λ. Since a larger attenuation coefficient corresponds to a larger mean travel distance for the electrons, each pixel on the sensor receives a larger amount of pixels, increasing the probability of activating the individual pixels.

By varying λ and creating new clusters to different values of E_{dep}, the data fit with the values shown in table 3.2. Some of the resulting functions are shown in figure 3.6.

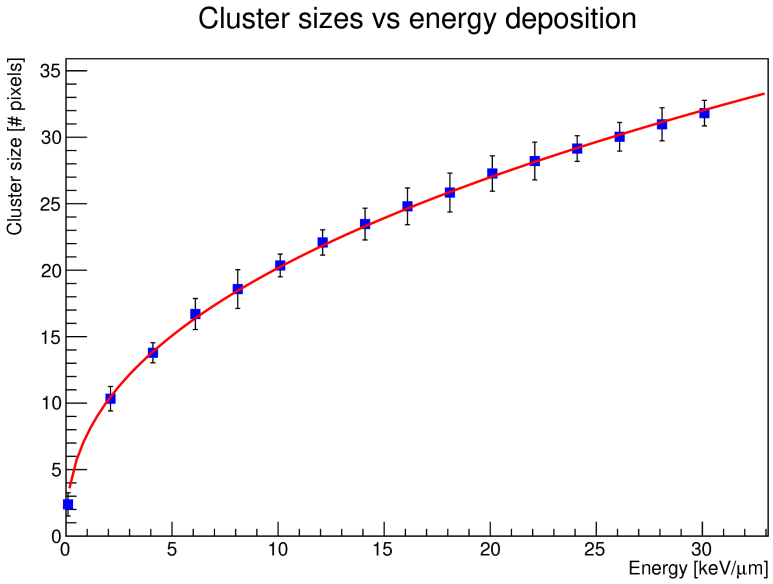

Figure 3.6: The cluster sizes created at different energy levels. The cluster sizes varied slightly by evaluating different hit positions for the proton. The blue points show the expected value for cluster size and the error bars show the standard deviation. The fitted curve was size = $7.66260(E_{dep})^{0.420307}$.

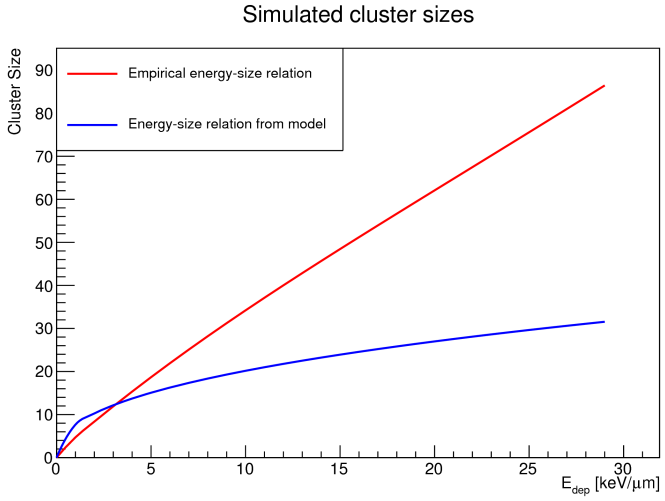

Figure 3.7: The energy-size relations for the the new charge diffusion model and the empirical charge diffusion model. The empirical model reaches larger cluster sizes by having higher order of terms.

Table 3.2: Parameters for the cluster size-energy relation for an assortment of values for λ. $\lambda = \infty$ means the electrons are not attenuated. The parameters are related to equation (3.2).

Attenuation coefficient [µm]		
λ	p_0	p_1
30	6.16421e+00	3.81925e-01
35	6.66920e+00	4.00565e-01
40	7.24521e+00	4.09395e-01
45	7.66260e+00	4.20307e-01
50	8.07114e+00	4.30877e-01
55	8.48959e+00	4.37136e-01
60	8.72351e+00	4.47309e-01
65	8.99963e+00	4.54506e-01
∞	1.67988e+01	5.66446e-01

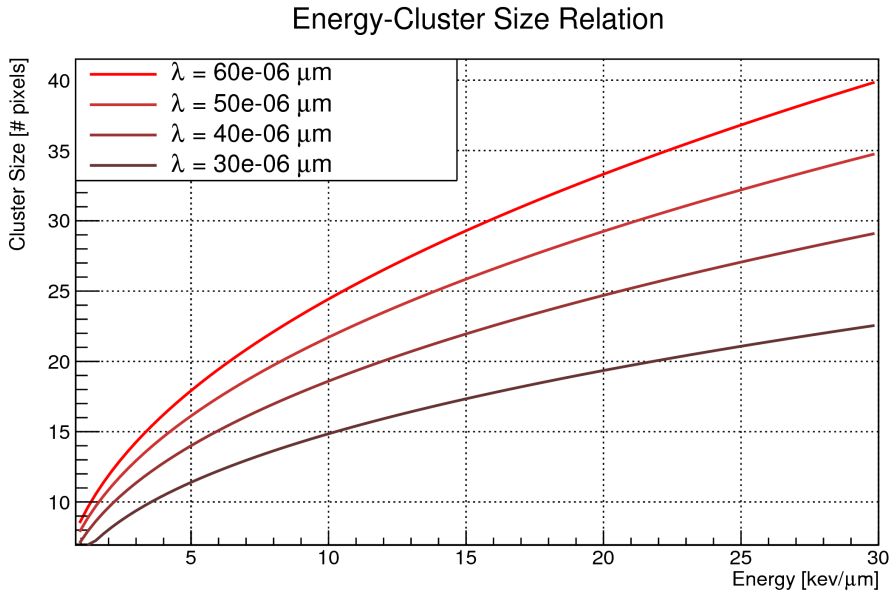

Figure 3.8: Energy-size relation for some attenuation coefficient values. Increasing the attenuation coefficient increases the cluster size, as expected. Similar for all the functions is the tendency to increase cluster size slowly at high energy deposit.

3.4 Finding the Proton Range

The resulting approximation for the relation between cluster size and energy deposition was implemented into the existing code. By using data acquired from a previous experiment, the validity of the model could be tested. The experiment was conducted with a 188 MeV proton beam. An analysis was created using the cluster size-energy correlation with λ =45 µm and ϕ =4.85 eV in figure 3.11.

The energy from the clusters were calculated by using the energy-size relationship from equation (3.2), using the values obtained by setting λ =45 µm, as in table 3.2. The range was found by extracting the range parameter R_0 from fitting equation (2.11) to the data. Some of the fits are shown in figure 3.9, with comparison to the fit using the empirical cluster model.

For a 188 MeV proton beam in water, the range is R_0 =233.4 mm, calculated with equation (2.11). Due to MCS in the detector, the projected proton range $R_{0,proj}$ is slightly shorter than the total range R_0. According to PSTAR [22], the detour factor is $R_{0,proj}/R_0$ = 0.9972, such that the observed range should be $R_{0,proj}$ =232.7 mm. The range straggle standard deviation is σ =2.1949 mm≈0.94 %, calculated with equation (2.12).

By using the range R_0 for the individual proton tracks, the remaining energy could be calculated for each depth in the calorimeter on each track. For observations on the relationship between the cluster sizes and remaining energy, using the ranges obtained using λ=45 µm, the data was plotted and is shown in figure 3.10.

The range calculation was performed using the different energy-size relations obtained from the different values of attenuation coefficient λ, as shown in table 3.2. The resulting WEPL and energy is shown in table 3.3.

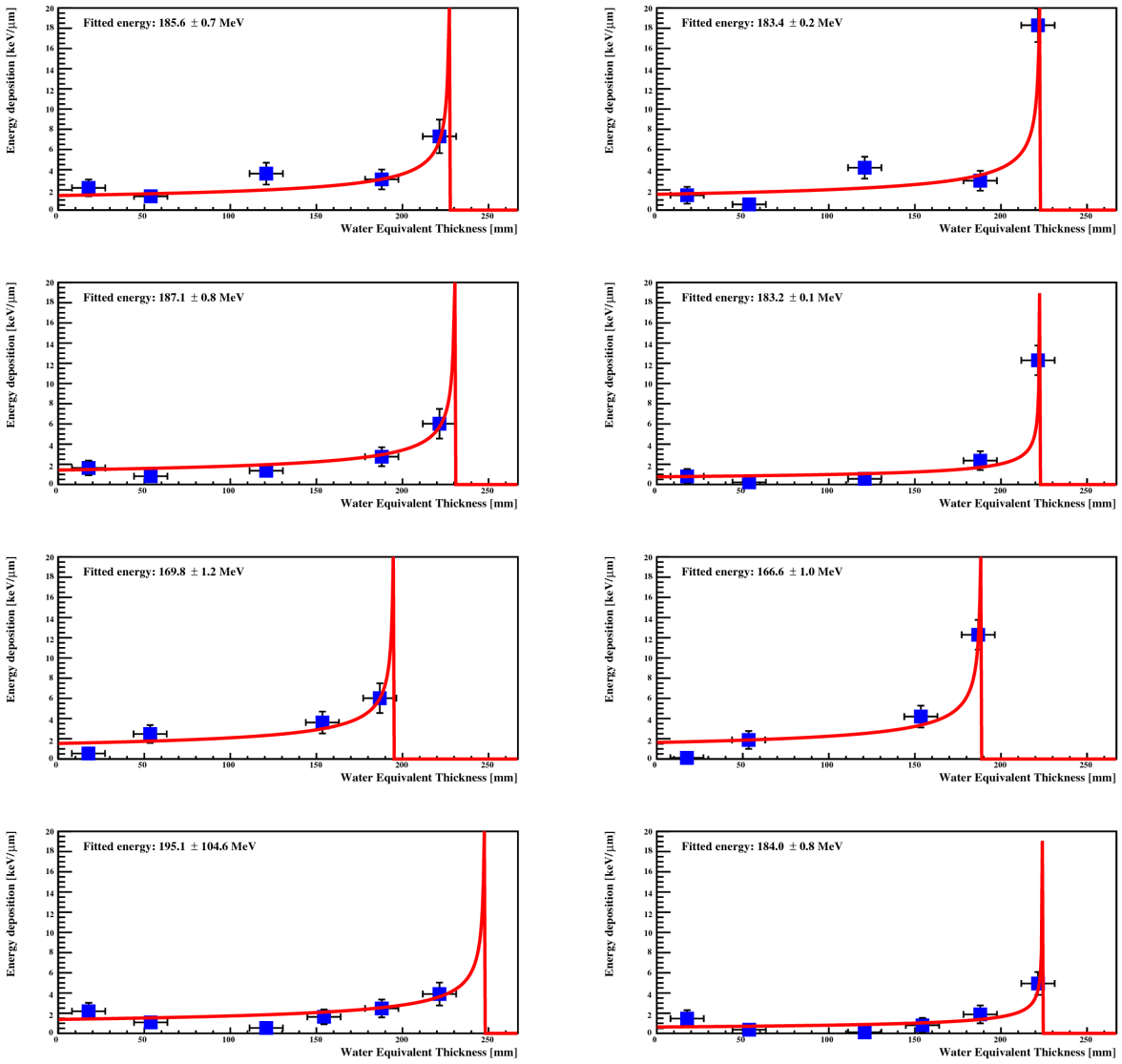

Figure 3.9: Fitted functions for cluster sizes observed along the track. ***The left column*** *of figures show the data from the empirical model.* ***The right column*** *of figures show the data from the new model. The new model shows a clear discrepancy on the larger energy deposit near the end of the tracks. The size-energy relation expects a much higher energy deposit at the larger cluster sizes, which causes the fit function to yield a shorter range.*

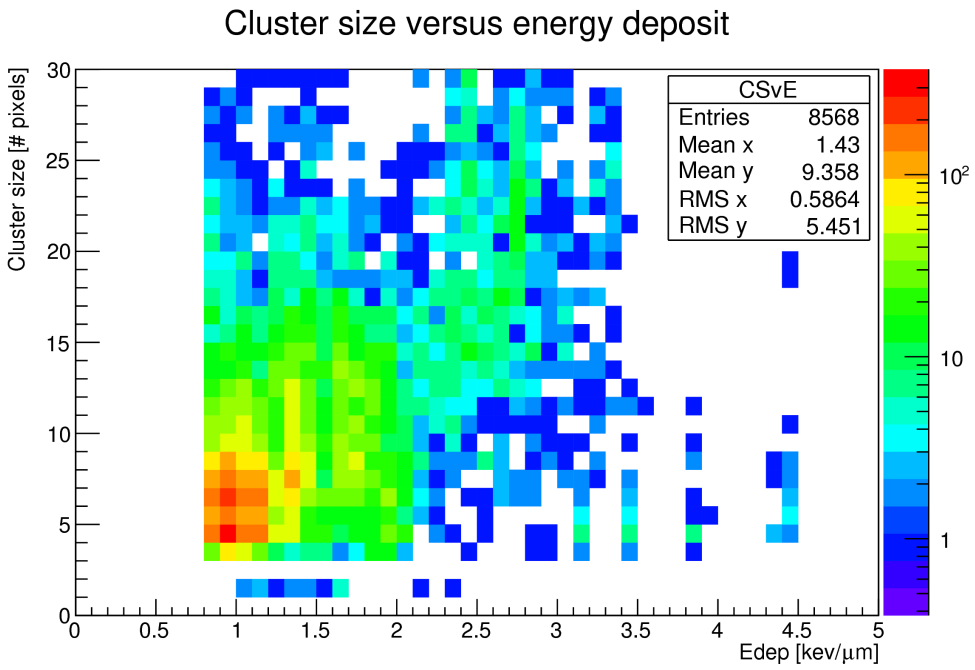

Figure 3.10: Histogram for the cluster sizes versus energy deposition. The energy deposit is found by using equation (2.14) *for the remaining energy at the layer depth, having found the total range. The data shows a sharp peak at* $E_{dep} = 1\,\mathrm{keV\mu m^{-1}}$*, as this corresponds to a proton energy of* 188 MeV. *The cluster size of 4 pixels is a likely shape for low energy depositions, where the activated pixels usually form a simple square.*

Table 3.3: The result from the range fits for different values of attenuation coefficient λ*. The proton beam has initial energy* 188 MeV.

λ	WEPL [mm]	Energy [MeV]
30	229.11 ± 26.59	186.37 ± 12.55
35	229.50 ± 26.44	186.55 ± 12.47
40	229.76 ± 26.53	186.67 ± 12.50
45	229.93 ± 26.44	186.75 ± 12.46
50	230.03 ± 26.30	186.80 ± 12.39
55	230.23 ± 26.35	186.89 ± 12.41
60	230.26 ± 26.54	186.91 ± 12.50
65	230.56 ± 26.55	187.05 ± 12.49
∞	232.51 ± 26.50	187.97 ± 12.43

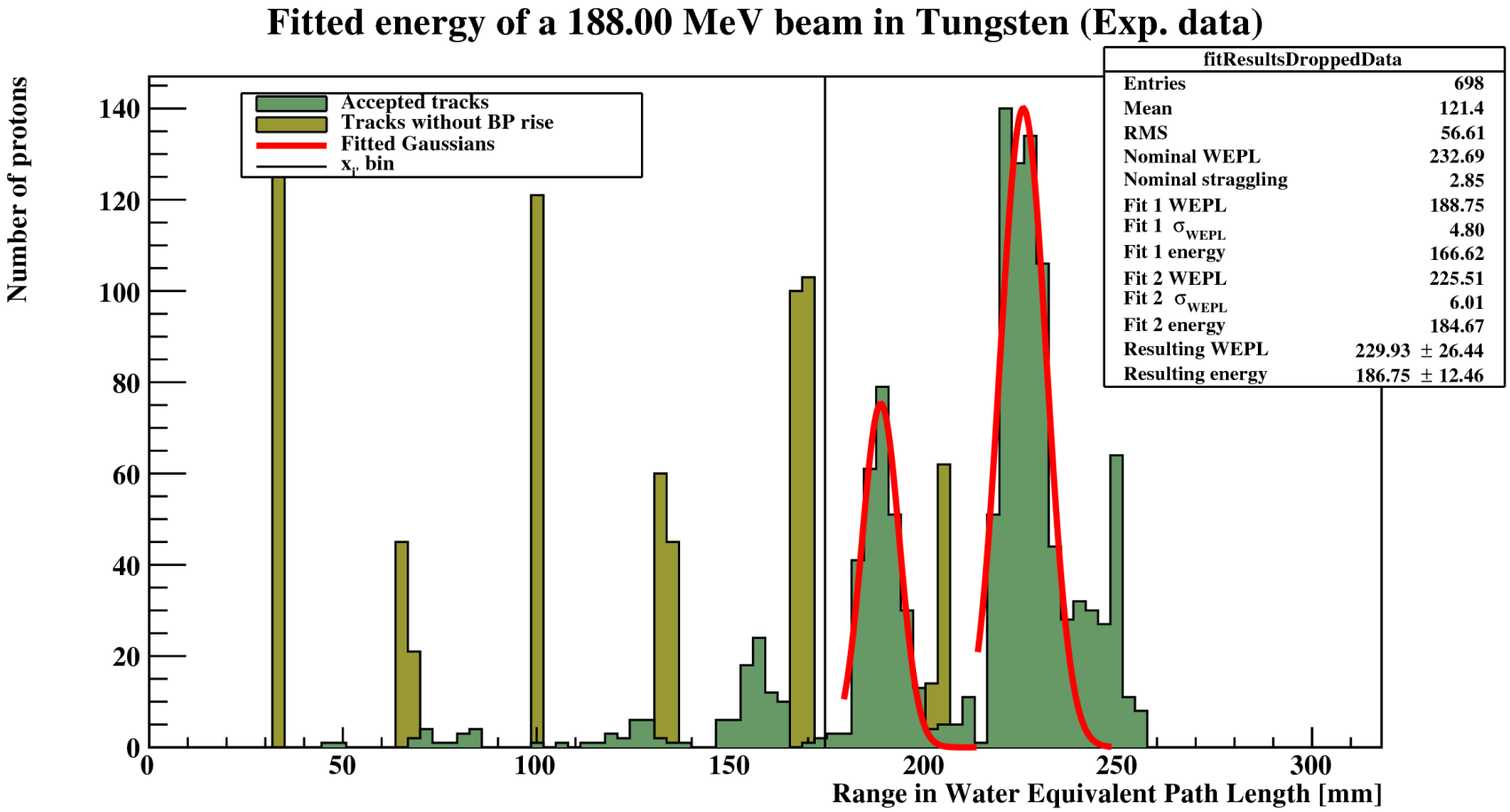

Figure 3.11: Histogram for proton ranges calculated utilizing the results from the new diffusion model. As expected, the proton ranges are Gaussian distributed in the histogram. However, the results fit two Gaussian distributions, which is not in agreement with the fact that a approximately monoenergetic beam was used. This is further discussed in section 4.5.

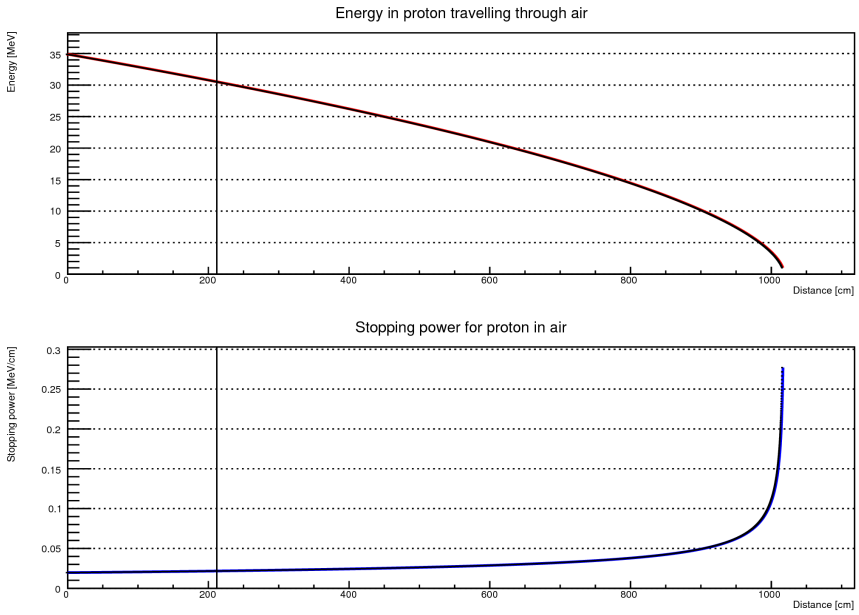

Figure 3.12: The energy and stopping power for a proton travelling through air. The vertical line shows the position of the pALPIDE chip, where the remaining energy is E =30.629 MeV. This position is far from the Bragg peak, such that the energy straggle is relatively small.

3.5 Analysis from the pALPIDE Chip

The pALPIDE chip has been tested with proton beams with initial energy 34.9 MeV, propagating through 212 cm of air [23].

By fitting stopping data from PSTAR [22] for protons travelling through air, it was obtained that

$$S = -\frac{dE}{dx} = 0.2725 \cdot E^{-0.7465}. \tag{3.3}$$

Using this, the remaining energy along the path was computed by stepwise reducing the energy, as shown in figure 3.12. It was found that the remaining energy after 212 cm is ≈30.629 MeV. This agrees with the study from the beam test, where the remaining energy is approximated to be 30 MeV.

From Monte Carlo simulations, it was found that the energy straggling for the beam is σ =0.021 MeV. Using this, the cluster size distributions was created for the cluster model, using the geometries in table 2.1. According to the group conducting the experiment, the energy deposit required to create an electron-hole pair in the pALPIDE epitaxial layer is 3.6 eV [23]. This was substituted for

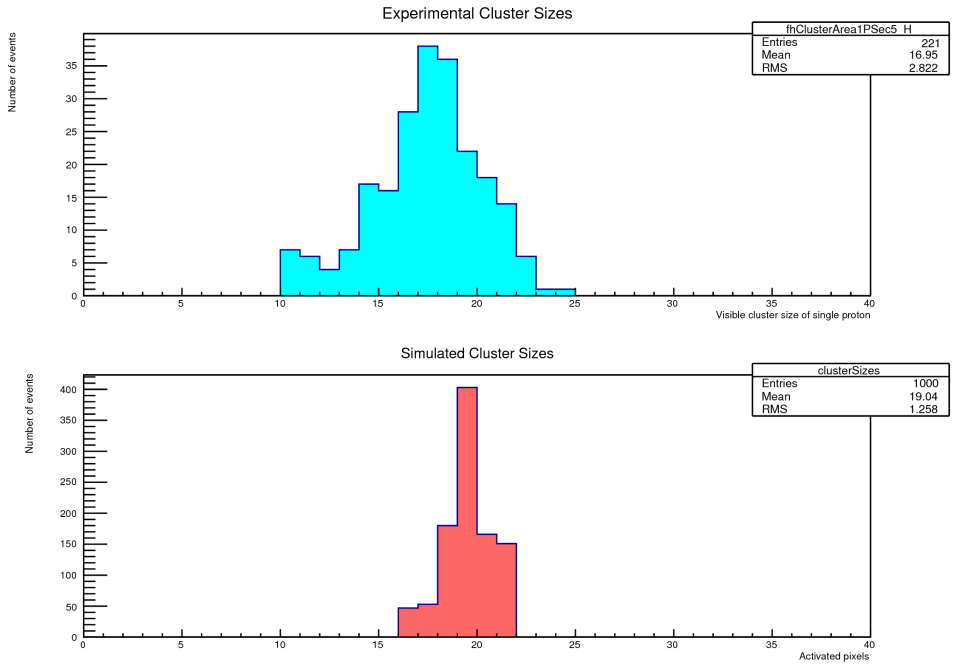

Figure 3.13: Histogram of cluster sizes for the pALPIDE chip. The blue histogram is the experimental values. The red histogram is the simulated values. It can be seen that the mean value is relatively close, but the variance of the simulated values is lower.

the work function in the simulation. The thickness of the epitaxial layer was adjusted to 18 µm.

The energy deposition in the epitaxial layer was calculated from equation (3.1), sampling random number from a Gaussian distribution with μ =30.629 MeV and σ =0.021 MeV. A histogram was created for the cluster sizes from the simulated model and is compared to the experimental results in figure 3.13.

Chapter 4

Discussion

By looking at the results of the diffusion model, it is clear that the implementation is not perfect. Figure 3.4 and 3.5 show some discrepancies. However, the model seems to partly fulfill the goals, as figure 3.11 show values which are close to the expectations.

4.1 Implementation of the Model

The implementation of the model was successful. By using an algorithm which performed the most resource intensive process initially and only once, the code had a feasible run time. The electron intensity was initially created and stored as a matrix of double precision floating point number. The cluster generating functions would only access the matrix for determining activated pixels, and not generate new intensity distributions. For each cluster, the code was therefore able to run at a upper bound of $\mathcal{O}(N^2)$ where N is the width of the area a pixel would be projected onto.

The code could be further optimized by precaching the various intensity distributions available for the different hit positions within a cluster. This could reduce the runtime significantly by eliminating the need for the intensity distribution generating function, as well as the pixel grouping function.

4.2 Intensity Distribution

When analyzing the intensity distribution, it is observed that an extremely high number of electrons are incident in the center sample area, seen in figure 3.1 and 3.2. According to the implementation, only one electron should be created per micrometer of epitaxial layer. Therefore, the intensity matrix should have a sum of 14. A quick observation shows that the distribution instead has a sum of several thousand.

The reason for this inaccuracy lies in the size of the sample areas. A sample area has one ninth of the width of a sensor pixel, ≈3 µm. When summarizing the contributions to the sample pixels, there is a contribution which occurs right above the sensor, with $h = 0.1\,\mu m$. Due to equation (2.5) where $|\vec{R}| = h$ and the fact that the electron is diffused at a much lower height than the width of a sample area, the calculation becomes very inaccurate.

However, the contribution in the center sample area is usually uninteresting as the center pixel will almost always get more than the threshold number of incident electrons and become activated. The calculation is still holds for the rest of the sample areas on the intensity distribution.

4.3 Diffusion Comparison

The figures of the calorimeter frames reveal that there are some discrepancies between the experimental data and the simulated clusters. The clusters in the experimental data shows a larger variety in cluster size and shape.

The cluster shapes are not completely random in the simulated diffusion. The shape is always as round as possible, following from the deterministic implementation of the intensity distribution and the size of the pixels. The experimental data, figure 3.4 shows some clusters which are impossible for the diffusion model to create, for example the cluster located at pixel $(840, 440)$, which features a dent on the left side.

The random variations of the simulated clusters could have been improved by adding noise to the pixels. The fake rate of 10^{-5} suggest that an average of ≈ 4 electrons are incident on each pixel throughout each cycle of the sensor. Adding this contribution could create clusters with different shapes, but at the cost of increasing the simulation time.

The energy deposit in the epitaxial layer is in reality Landau distributed for each position, such that the proton could deposit larger amounts of energy to some electrons. This could cause electrons to travel a larger distance and ionize along their trajectory. As a result, asymmetric cluster shapes could be created. This factor was not included for the diffusion model, but could have introduced a larger standard deviation for the cluster sizes.

4.4 Energy-Size Relation

The function which described cluster size versus energy was fitted to the simulated values with a very small error. However, the cluster sizes increases slowly at larger values of deposited energy. This does not agree with the function obtained from the empirical diffusion model. The comparison between the models is shown in figure 3.7. Since the empirical model increases the standard deviation in its Gaussian function, it is capable of having a more linear fit.

From figure 3.10, no clear relation between cluster size and energy is visible. For a range of E_{dep} between 1-2keVμm^{-1}, there exist a lot of clusters with sizes between 3 and 15 pixels. This implies that other factors take precedence to a lot of the cluster sizes. Some of these factors could include variations in epitaxial layer thickness, variations in material density and contributions from the internal noise within the MAPS.

Furthermore on figure 3.10, an area of larger clusters is observed between 2 and 3 keVμm^{-1} with cluster sizes between 12 and 25 pixels. This area is likely represented by protons which are about to reach the Bragg peak. The positioning of the area verifies that an energy-size model is feasible for use in future studies as well.

4.5 Range Calculation

The range calculations were performed by fitting function (2.11) to the data, which yielded the data in figure 3.9 and 3.11. In the latter figure, it is observed that two separate Gaussian distributions fit the analysis. This is believed to be a result of two factors:

- The function fitted to the data has an affinity towards applying the peak at the last observed point, i.e. the last sensor chip the proton traversed. When the range R_0 lies be-

tween two sensor chips, the real Gaussian distribution has a tail which overlaps with the consecutive sensor layer, creating a new Gaussian distribution, as discussed in [2]. The leftmost data points seen in figure 3.9 are of low importance, as the function for Linear Energy Transfer (LET), equation (2.11) is increasing very slowly at the range before the peak.

- The energy-size relation (equation (3.2)) causes the energy to rise rapidly compared to the cluster size. Thus the calculated energy at the uttermost detector is too high, causing an even stronger affinity to place the Bragg peak near the sensor layer.

Another cause of error has been introduced by using a scaling factor for the fit function. If a data point is unusually high, the entire function could scale to suggest an unrealistic energy transfer for the material, rather than observing it as an inappropriate value. In figure 3.9, it can be seen that some curves have initial energy deposit $E_{dep} \approx 1.5$ keV, which would correspond to a proton with initial energy slightly above 85 MeV, yet it travels the full distance that a 188 MeV proton would be assumed to do.

Discrepancies in the cluster size could emerge from the experimental data when two protons are within short range of each other. Two overlapping clusters could have been interpreted as one cluster of larger size, causing the energy approximation to be too high and introducing errors in the fitted function.

Even though the range straggle for a 188 MeV proton beam is small, less than 1 % according to equation (2.12), the deviation from the fits were large as a result of the range fit, above 10 %.

Interestingly, the variation in the attenuation coefficient λ, table 3.3, shows that using no attenuation yields the most energy with closest mean estimation. However, the standard deviation is close to constant for the variety of range calculations, and the values cannot compete with the empirical model.

4.6 pALPIDE

Unfortunately, there was not a lot of available data from the pALPIDE sensor chip to be analyzed. To the authors knowledge, only a histogram of cluster sizes from a mean 30 MeV proton beam

was made available. However, the generated clusters from the diffusion model fit decently with the data, as seen in figure 3.13. The discrepancy in variation for the cluster sizes can be a result of a variety of factors:

- Noise can have activated pixels, creating some larger clusters and leaving other as smaller clusters.
- The simplification in the charge diffusion model could yield unrealistically low variation when the number of generated electron-hole pairs is relatively small. In addition, the nature of the Landau-distributed energy loss of the protons was not considered.
- Discrepancies in the physical properties in the MC simulations versus the real conditions could yield low values for energy straggle.

From the experimental data, it can be seen that cluster sizes of <9 pixels are absent, even though the geometry suggest that 9-pixel clusters are highly probable. This suggests that those clusters have been discarded, and it is possible that the sensor chip has more noise than observed.

For the experimental setup, it is possible that two protons within a close range create clusters which overlap, such that the computer program would interpret it as a single, larger cluster. This could cause both larger deviations and higher mean cluster size values.

In order to find the energy-size relation for the pALPIDE, the cluster sizes at different energy levels need to be experimentally tested. Only then can the model be fitted for energy-size relationships.

Chapter 5

Conclusion

5.1 Conclusion

The implementation of the showed some discrepancies between the cluster sizes and energy deposition at large values of E_{dep}. However, the lower values of E_{dep} yielded cluster sizes and shapes that fit the experimental values well. As a result, it is believed that the cluster model is appropriate only to lower energy deposition values, while another model would be needed to take into account the higher polynomial orders of the energy-size relation. A combination of the described diffusion model and an empirical term could fit the data more appropriately.

5.2 Directions for Future Work

For future work, the natural next step is to examine the model with an additional term to recreate the clusters at high values of energy deposition. Currently, the empirical model for clusters does a better job at representing the sensor response from a proton.

Additional experimental values for the pALPIDE chip is required to analyze the feasibility of the diffusion model applied to it. Using a spectrum of energies would make it possible to observe a relationship between energy and size and analyze it for use in the DTC.

To obtain better resolution of the remaining energy of a proton incident to the DTC, it would be advantageous to use additional layers of sensor chips and thinner absorber plates. Observing the Bragg peak in the detector as a result of cluster size would be simplified.

Since the pALPIDE chip can be biased to control the cluster sizes through adding a voltage, it is possible that a certain voltage yields more accurate estimations for the energy depositions in each layer. It could be advantageous to find an appropriate bias for each layer in the DTC to obtain the best energy approximations.

Bibliography

[1] E Hansen. Charge diffusion modelling for a monolithic active pixel sensor detector with application to proton ct. Project Thesis, 2016.

[2] HES Pettersen, J Alme, AVD Brink, M Chaar, D Fehlker, I Meric, OH Odland, T Peitzmann, E Rocco, H Wang, S Yang, C Zhang, and D Röhrich. Proton tracking in a high-granularity digital tracking calorimeter for proton ct purposes. (unpublished, accepted 2017).

[3] L Maczewski. Measurements and simulations of maps (monolithic active pixel sensors) response to charged particles-a study towards a vertex detector at the ilc. *arXiv preprint arXiv:1005.3710*, 2010.

[4] M Mager, ALICE Collaboration, et al. Alpide, the monolithic active pixel sensor for the alice its upgrade. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 824:434–438, 2016.

[5] RR Wilson. Radiological use of fast protons. *Radiology*, 47(5):487–491, 1946.

[6] LD Skarsgard. Radiobiology with heavy charged particles: a historical review. *Physica Medica*, 14:1–19, 1998.

[7] M Jermann. Particle therapy statistics in 2014. *International Journal of Particle Therapy*, 2(1):50–54, 2015.

[8] V Giacometti. Modelling and improvement of proton computed tomography. presentation, 2017.

[9] KM Hanson, JN Bradbury, TM Cannon, RL Hutson, DB Laubacher, R Macek, MA Paciotti, and CA Taylor. The application of protons to computed tomography. *Journal of Computer Assisted Tomography*, 2(5):671–674, 1978.

[10] H Paganetti. *Proton therapy physics*. CRC Press, 2016.

[11] J Lilley. *Nuclear Physics*. Wiley, 2001.

[12] L D Landau. On the energy loss of fast particles by ionization. *J. Phys.*, 8:201–205, 1944.

[13] G Poludniowski, NM Allinson, and PM Evans. Proton radiography and tomography with application to proton therapy. *The British journal of radiology*, 88(1053):20150134, 2015.

[14] CT Quiñones, JM Létang, and S Rit. Filtered back-projection reconstruction for attenuation proton ct along most likely paths. *Physics in medicine and biology*, 61(9):3258, 2016.

[15] U Schneider, E Pedroni, and A Lomax. The calibration of ct hounsfield units for radiotherapy treatment planning. *Physics in medicine and biology*, 41(1):111, 1996.

[16] D Fehlker, J Alme, A Van Den Brink, AP De Haas, G-J Nooren, M Reicher, D Röhrich, M Rossewij, K Ullaland, and S Yang. Electronics for a highly segmented electromagnetic calorimeter prototype. *Journal of Instrumentation*, 8(03):P03015, 2013.

[17] G Aglieri, C Cavicchioli, PL Chalmet, N Chanlek, A Collu, P Giubilato, H Hillemanns, A Junique, M Keil, D Kim, et al. Monolithic active pixel sensor development for the upgrade of the alice inner tracking system. *Journal of Instrumentation*, 8(12):C12041, 2013.

[18] R Brun and F Rademakers. Root—an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1):81–86, 1997.

[19] T Bortfeld. An analytical approximation of the bragg curve for therapeutic proton beams. *Medical physics*, 24(12):2024–2033, 1997.

[20] W Ulmer. Theoretical aspects of energy–range relations, stopping power and energy straggling of protons. *Radiation physics and chemistry*, 76(7):1089–1107, 2007.

[21] HB Michaelson. The work function of the elements and its periodicity. *Journal of applied physics*, 48(11):4729–4733, 1977.

[22] MJ Berger, JS Coursey, and MA Zucker. Estar, pstar, and astar: Computer programs for calculating stopping-power and range tables for electrons, protons, and helium ions (version 1.21).

[23] J. Ferencei F. Krizek, T. Vanat. Response of palpide3 to 30 mev proton beam. presentation, 2017.

Appendix A

Derivation of the Charge Diffusion Model

A.1 Introduction

This is a derivation of the charge diffusion model as proposed in reference [3]. The derivation makes use of a rough assumption that electrons diffuse isotropically and reflect on the substrate with an angle of reflection equal to the angle of incidence.

The reference states equation (A.2) and (A.3). In this appendix, a geometrical justification for that result is proposed and the distribution is converted to Cartesian coordinates for computational convenience.

A.2 Derivation

The geometry for the derivation is shown in figure A.1 and A.2.

The probability for an electron to travel at least a distance R in a medium with attenuation coefficient λ is assumed to be

$$P(r > R) = \exp\left(-\frac{|\vec{R}|}{\lambda}\right). \tag{A.1}$$

By assuming a sphere with center in P and radius R, such that $\vec{R} = P\vec{M}$ and the area of the sphere $A = 4\pi|\vec{R}|$, the probability that an electron moving in a random direction from point P

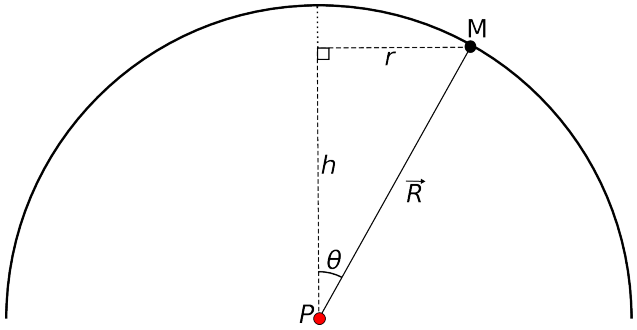

Figure A.1: The geometry for the charge diffusion model. The electrons diffuse uniformly, creating a sphere around the origin P with uniform probability density. $\vec{R}$ is the radius of the sphere, h is the distance between P and the sensor surface and r is the length of the projection of $\vec{R}$ on the sensor surface.

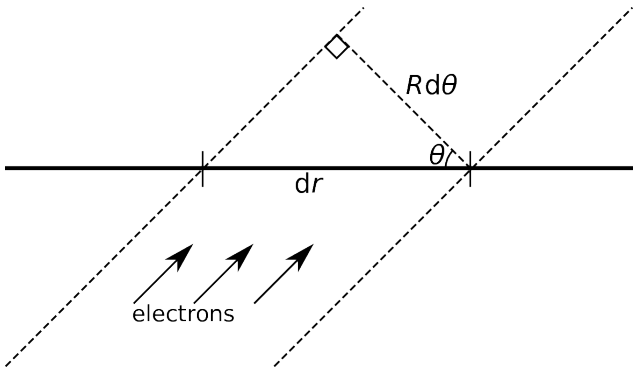

Figure A.2: The geometry of the incident electrons on the sensor surface.

reaches point M becomes:

$$\rho(\vec{R})\mathrm{d}\phi\mathrm{d}r = \frac{\mathrm{d}\Omega|\vec{R}|^2}{4\pi|\vec{R}|^2}\exp\left(-\frac{|\vec{R}|}{\lambda}\right) = \frac{\mathrm{d}\Omega}{4\pi}\exp\left(-\frac{|\vec{R}|}{\lambda}\right). \tag{A.2}$$

Making use of that $\mathrm{d}\Omega = \mathrm{d}\theta\mathrm{d}\phi\sin\theta$ and $\mathrm{d}\theta = (\cos\theta/R)\mathrm{d}r$ (as shown in figure A.2), we obtain:

$$\frac{\mathrm{d}\Omega}{4\pi}\exp\left(-\frac{|\vec{R}|}{\lambda}\right) = \frac{\mathrm{d}\theta\mathrm{d}\phi\sin\theta}{4\pi}\exp\left(-\frac{|\vec{R}|}{\lambda}\right) = \frac{\mathrm{d}r\mathrm{d}\phi}{4\pi}\frac{hr}{|\vec{R}|^3}\exp\left(-\frac{|\vec{R}|}{\lambda}\right). \tag{A.3}$$

In Cartesian coordinates, we set $\mathrm{d}r = \mathrm{d}x$ and $r\mathrm{d}\phi = \mathrm{d}y$. Inserting this into equation (A.3), we obtain:

$$\rho(\vec{R})\mathrm{d}x\mathrm{d}y = \frac{h}{4\pi|\vec{R}|^3}\exp\left(-\frac{|\vec{R}|}{\lambda}\right)\mathrm{d}x\mathrm{d}y. \tag{A.4}$$

The electrons that have initial direction away from the detector surface, away from the substrate, are subject to the electric barrier present at the end of the epitaxial layer. It is here assumed that the electrons have the same reflected angle as incident angle and that we can treat these electrons as having a projected origin at $h' = 2l - h$ beneath the barrier (as stated in [3]), where l is the total depth of the epitaxial layer. This leads to a reflected contribution of

$$\rho(\vec{R}')\mathrm{d}x\mathrm{d}y = \frac{2l-h}{4\pi\left|\vec{R}'\right|^3}\exp\left(-\frac{\left|\vec{R}'\right|}{\lambda}\right)\mathrm{d}x\mathrm{d}y, \tag{A.5}$$

where $\left|\vec{R}'\right|$ is the vector pointing from the projected origin to the point M.

The sum of the two terms, $\rho(\vec{R}')$ and $\rho(\vec{R})$ is the probability distribution function for the electrons incidence. Due to the attenuation coefficient, as charges are trapped in the silicon, the distribution does not integrate to 1, but instead to the probability for a charge to reach any point on the sensor layer.

Appendix B

Program code for the report

This appendix contains code from the project. Some of the code from the project has been directly written into another project (the code from [2]) and is not included here.

B.1 Creating the Cluster

This code creates clusters by applying equation (2.7) to a matrix and grouping the sampled areas into pixels.

```
#include <iostream>
#include <math.h>
#include <time.h>
#include <stdlib.h>
using namespace std;

Double_t attenuationConstant= 45e-06;

//Practical use of the code can be the the following:
Int_t makeNClusters(Int_t N){
  //Setting the constants
  Int_t width=9; //width (in number of pixels) of the diffused area to fill.
   Typically less than 10.
  Int_t groupWidth=9; //sampling width of each pixel. Typically 9.
  Int_t i, j;
```

```
15  Double_t sensorDepthInMicron=14; //Sensor depth. MIMOSA23 is 14.
    Double_t LET;
17  Double_t mpvLET=0.736e03; //Most probable value of Linear energy transfer [
     eV/um]
    Double_t sigmaLET=0.0925744e03; //Sigma of Linear energy transfer [eV/um]
19  TRandom landau; //Random variable for LET
    Double_t threshold=25; //Number of electrons needed to activate a pixel. ~25
       for normal setting on MIMOSA23.
21  Double_t sensorPixelSize=3e-05; //Pixel size on the MAPS. 30um for MIMOSA23.
    Double_t samplePixelSize=sensorPixelSize/groupWidth; //Sampling pixel size.
23  Double_t workFunction=4.85; //Energy needed to free one electron in the
     epitaxial layer. 4.85 eV for silicon
    Double_t electronsPerMicrometer; //Number of electrons hit free per
     micrometer.
25
    TMatrixD cluster(width,width);
27  srand(time(NULL));
    TMatrixD pixelMap=createIntensityMap((width+1)*groupWidth,
     sensorDepthInMicron, samplePixelSize); //Create the intensity map only once
     .
29
    TCanvas *c1 = new TCanvas("c1","Cluster sizes",200,10,700,900);
31  TH1F * clusterSizeHistogram = new TH1F("Clusters", "Cluster sizes", 20, 0,
     20);
    for (Int_t iteration=0; iteration<N; iteration++){
33    LET=landau.Landau(mpvLET, sigmaLET);
      electronsPerMicrometer=LET/workFunction;
35    i = 1+rand() % 9;
      j = 1+rand() % 9;
37    cluster=createCluster(groupMap(pixelMap, width, i, j, groupWidth)*
     electronsPerMicrometer, width, threshold); //Cluster map created with the
     parametres.
       clusterSizeHistogram->Fill(clusterSize(cluster, width));
39  }
    clusterSizeHistogram->Draw();
41  c1->Update();
```

```
}

Int_t clusterSize(TMatrixD cluster, Int_t width){
  Int_t sum=0;
  for (Int_t j=0;j<width;j++){
    for (Int_t i=0;i<width;i++){
      sum+=cluster[i][j];
    }
  }
  return sum;
}

TMatrixD createSensorMap(Int_t width, Double_t sensorDepthInMicron, Double_t
    LET, Int_t offsetX, Int_t offsetY){
  Int_t groupWidth=9;
  Double_t threshold=25;
  Double_t sensorPixelSize=3e-05;
  Double_t samplePixelSize=sensorPixelSize/groupWidth;
  Double_t workFunction=4.85;
  Double_t electronsPerMicrometer=LET/workFunction;
  TMatrixD pixelMap=createIntensityMap((width+1)*groupWidth,
    sensorDepthInMicron, samplePixelSize);
  TMatrixD sensorMap = groupMap(pixelMap, width, offsetX, offsetY, groupWidth)
    ;
  sensorMap=sensorMap*electronsPerMicrometer;
  cluster=createCluster(sensorMap, width, threshold);
  return sensorMap;
}

TMatrixD groupMap(TMatrixD pixelMap, Int_t width, Int_t offsetX, Int_t offsetY
    , Int_t groupWidth){
  TMatrixD sensorMap(width, width);
  if (offsetX>=1 && offsetX<=(groupWidth)+1 && offsetY>=0 && offsetY<=(
    groupWidth)){
    //cout << "proton hits the pixel at \n x = " << offsetX*3e-06 << "\n y = "
     << offsetY*3e-06 << endl;
```

```
      for (Int_t j=0; j<width; j++){
73      for (Int_t i=0; i<width; i++){
          sensorMap[i][j]=groupPixels(pixelMap, groupWidth, i, offsetX, j,
     offsetY);
75      }
      }
77  }
    else {
79    cout << "offset only available between 1 and 10" << endl;
    }
81  return sensorMap;
}
83
Double_t groupPixels(TMatrixD pixelMap, Int_t groupWidth, Int_t x, Int_t
    offsetX, Int_t y, Int_t offsetY){
85  Double_t intensity=0;
    for (Int_t j=y*groupWidth; j<(y+1)*groupWidth; j++){
87    for (Int_t i=x*groupWidth; i<(x+1)*groupWidth; i++){
        intensity+=pixelMap[i+offsetX][j+offsetY];
89    }
    }
91  return intensity;
}
93

95 TMatrixD createCluster(TMatrixD sensorMap, Int_t width, Double_t threshold){
    TMatrixD cluster(width, width);
97  for (Int_t j=0; j<width; j++){
      for (Int_t i=0; i<width; i++){
99      if (sensorMap[i][j]>=threshold){
          cluster[i][j]=1;
101     }
        else{
103       cluster[i][j]=0;
        }
105     //cout << cluster[i][j] << " ";
```

```
      }
107   //cout << endl;
    }
109   return cluster;
  }
111
  TMatrixD createIntensityMap(Int_t size, Double_t sensorDepthInMicron, Double_t
      samplePixelSize){
113   Double_t sensorDepth= sensorDepthInMicron*1e-06;
    TMatrixD pixelMap(size, size);
115   pixelMap=fillWithIntensity(size, pixelMap, sensorDepth, samplePixelSize);
    return pixelMap;
117 }

119 TMatrixD fillWithIntensity(Int_t size, TMatrixD pixelMap, Double_t sensorDepth
     , Double_t samplePixelSize){
    Int_t center=floor(size/2);
121   Double_t distanceFromCenter;
    Double_t distance=0;
123   for (Int_t j=0; j<size; j++){
      for (Int_t i=0; i<size; i++){
125     distanceFromCenter=sqrt((i-center)**2+(j-center)**2)*samplePixelSize;
        pixelMap[i][j]=pixelIntensity(distanceFromCenter, sensorDepth,
     samplePixelSize);
127    }
    }
129   return pixelMap;
  }
131
  Double_t pixelIntensity(Double_t distanceFromCenter, Double_t sensorDepth,
     Double_t samplePixelSize){
133   Double_t const pi = 3.14159;
    Double_t stepSize=1e-07;
135   Double_t intensity=0;
    Double_t distanceFromPoint, indirectDistanceFromPoint, directContribution,
     indirectContribution;
```

```
137   for (Double_t height=sensorDepth; height>0; height-=stepSize){
        distanceFromPoint=sqrt(pow(distanceFromCenter,2)+pow(height,2));
139     indirectDistanceFromPoint=sqrt(pow(distanceFromCenter,2)+pow(2.*
      sensorDepth-height,2));
        //directContribution=height/(4.*pi*pow(distanceFromPoint,3.0))*exp(-
      distanceFromPoint/attenuationConstant);
141     directContribution=height/(4.*pi*pow(distanceFromPoint,3.0)); //No
      attenuation
        //indirectContribution=(2.*sensorDepth-height)/(4.*pi*pow(
      distanceFromPoint,3.0))*exp(-indirectDistanceFromPoint/attenuationConstant)
      ;
143     indirectContribution=(2.*sensorDepth-height)/(4.*pi*pow(distanceFromPoint
      ,3.0)); //No attenuation
        intensity+=(directContribution+indirectContribution)*pow(samplePixelSize
      ,2)*stepSize/(1e-06);
145   }
      return intensity;
147 }
```

code/Cluster.C

B.2 Analysis of Diffused Clusters

This code makes use of Cluster.C to create an intensity map and do analysis on the clusters created from it.

```
#include "Cluster.C"
#include <TCanvas.h>
#include <iostream>
#include <math.h>
#include <TF1.h>
#include <TH2I.h>
#include <TH2D.h>
#include <TGraph.h>
#include <stdlib.h>
#include <TRandom.h>
#include <TGraphErrors.h>
using namespace std;

void makeClusterComparison(Double_t energy1, Double_t energy2, Double_t
    energy3){
    Int_t width=9;
    Int_t groupWidth=9;
    Double_t pixelSize=3e-05;
    Double_t samplePixelSize=pixelSize/groupWidth;
    Double_t workFunction=4.85;
    Double_t electronsPerMicrometer1=energy1*1e03/workFunction;
    Double_t electronsPerMicrometer2=energy2*1e03/workFunction;
    Double_t electronsPerMicrometer3=energy3*1e03/workFunction;
    Double_t threshold = 25;
    Double_t sensorDepth = 14;

    TMatrixD intensityMap=createIntensityMap((width+1)*groupWidth, sensorDepth,
     samplePixelSize);
    TMatrixD clusterMat1(width, width);
    TMatrixD clusterMat2(width, width);
    TMatrixD clusterMat3(width, width);
    TMatrixD clusterMat4(width, width);
```

```
31    TMatrixD clusterMat5(width, width);
      TMatrixD clusterMat6(width, width);
33    TMatrixD clusterMat7(width, width);
      TMatrixD clusterMat8(width, width);
35    TMatrixD clusterMat9(width, width);

37    TCanvas * c1 = new TCanvas("c1","Clusters vs hitposition", 200, 10, 700,
      1400);
      c1->Divide(3,3);
39
      TH2I * cluster1 = new TH2I("cluster1", "Hit in the center of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
41    TH2I * cluster2 = new TH2I("cluster2", "Hit in the center of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
      TH2I * cluster3 = new TH2I("cluster3", "Hit in the center of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
43    TH2I * cluster4 = new TH2I("cluster4", "Hit on the edge of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
      TH2I * cluster5 = new TH2I("cluster4", "Hit on the edge of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
45    TH2I * cluster6 = new TH2I("cluster4", "Hit on the edge of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
      TH2I * cluster7 = new TH2I("cluster4", "Hit in the corner of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
47    TH2I * cluster8 = new TH2I("cluster4", "Hit in the corner of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
      TH2I * cluster9 = new TH2I("cluster4", "Hit in the corner of the pixel;x-
      position;y-position", 10, 0, 10, 10, 0, 10);
49
      clusterMat1=createCluster(groupMap(intensityMap, width, 5, 5, groupWidth)*
      electronsPerMicrometer1, width, threshold);
51    clusterMat2=createCluster(groupMap(intensityMap, width, 5, 5, groupWidth)*
      electronsPerMicrometer2, width, threshold);
      clusterMat3=createCluster(groupMap(intensityMap, width, 5, 5, groupWidth)*
      electronsPerMicrometer3, width, threshold);
```

```
53    clusterMat4=createCluster(groupMap(intensityMap, width, 1, 5, groupWidth)*
      electronsPerMicrometer1, width, threshold);
      clusterMat5=createCluster(groupMap(intensityMap, width, 1, 5, groupWidth)*
      electronsPerMicrometer2, width, threshold);
55    clusterMat6=createCluster(groupMap(intensityMap, width, 1, 5, groupWidth)*
      electronsPerMicrometer3, width, threshold);
      clusterMat7=createCluster(groupMap(intensityMap, width, 1, 1, groupWidth)*
      electronsPerMicrometer1, width, threshold);
57    clusterMat8=createCluster(groupMap(intensityMap, width, 1, 1, groupWidth)*
      electronsPerMicrometer2, width, threshold);
      clusterMat9=createCluster(groupMap(intensityMap, width, 1, 1, groupWidth)*
      electronsPerMicrometer3, width, threshold);
59
      for (Int_t j=0; j<width; j++){
61       for (Int_t i=0; i<width; i++){
            if (clusterMat1[i][j]){
63             cluster1->Fill(i,j);
            }
65          if (clusterMat2[i][j]){
               cluster2->Fill(i,j);
67          }
            if (clusterMat3[i][j]){
69             cluster3->Fill(i,j);
            }
71          if (clusterMat4[i][j]){
               cluster4->Fill(i,j);
73          }
            if (clusterMat5[i][j]){
75             cluster5->Fill(i,j);
            }
77          if (clusterMat6[i][j]){
               cluster6->Fill(i,j);
79          }
            if (clusterMat7[i][j]){
81             cluster7->Fill(i,j);
            }
```

```
83          if (clusterMat8[i][j]){
               cluster8->Fill(i,j);
85          }
            if (clusterMat9[i][j]){
87             cluster9->Fill(i,j);
            }
89       }
      }
91
      c1->cd(1);
93    gPad->SetGrid();
      cluster1->Draw("COL");
95    c1->cd(2);
      gPad->SetGrid();
97    cluster4->Draw("COL");
      c1->cd(3);
99    gPad->SetGrid();
      cluster7->Draw("COL");
101   c1->cd(4);
      gPad->SetGrid();
103   cluster2->Draw("COL");
      c1->cd(5);
105   gPad->SetGrid();
      cluster5->Draw("COL");
107   c1->cd(6);
      gPad->SetGrid();
109   cluster8->Draw("COL");
      c1->cd(7);
111   gPad->SetGrid();
      cluster3->Draw("COL");
113   c1->cd(8);
      gPad->SetGrid();
115   cluster6->Draw("COL");
      c1->cd(9);
117   gPad->SetGrid();
      cluster9->Draw("COL");
```

```
119     c1->Update();
    }
121
    void makeClustersAtEnergyLevels(Double_t start, Double_t increment, Double_t
        stop){
123     Int_t width=11;
        Int_t iteration=0;
125     Int_t groupWidth=9;
        Int_t i,j, min, max, CS, clusterNumber;
127     Double_t clusterSize;
        Double_t sensorDepth=14;
129     TRandom xOffset, yOffset;
        Double_t threshold=25;
131     Double_t pixelSize=3e-05;
        Double_t samplePixelSize=pixelSize/groupWidth;
133     Double_t workFunction=4.85;
        Double_t electronsPerMicrometer, sqDev;
135     TMatrixD cluster(width, width);
        TMatrixD intensityMap=createIntensityMap((width+1)*groupWidth, sensorDepth,
         samplePixelSize);
137     Double_t clusterSizes[10000];
        Double_t clusterSizesMin[10000];
139     Double_t clusterSizesMax[10000];
        Double_t stdDev[10000];
141     Double_t energies[10000];
        Double_t energyLevelCS[100];
143
        TCanvas *c1 = new TCanvas("c1", "Cluster Sizes vs Energy", 200, 10, 700,
        500);
145
        for (Double_t energy=start; energy<stop; energy+=increment){
147        electronsPerMicrometer=energy*1e03/workFunction;
           clusterSize=0;
149        min=101;
           max=0;
151        clusterNumber=0;
```

```
   for (Int_t j=0;j<groupWidth;j++){
      for (Int_t i=0;i<groupWidth;i++){
         cluster=createCluster(groupMap(intensityMap, width, i, j,
groupWidth)*electronsPerMicrometer, width, threshold);
         CS=cluster.Sum();
         clusterSize+=CS/pow(groupWidth,2);
         energyLevelCS[clusterNumber]=CS;
         if (CS>max){max=CS;}
         if (CS<min){min=CS;}
         clusterNumber++;
      }
   }
   sqDev=0;
   for (Int_t i=0;i<pow(groupWidth,2);i++){
      sqDev+=pow(energyLevelCS[i]-clusterSize,2);
   }
   stdDev[iteration]=pow(sqDev/(pow(groupWidth,2)-1.),1./2.);

   cout << "Average cluster size for " << energy << " keV/um \n is " <<
clusterSize << endl;
   cout << "Standard deviation: " << stdDev[iteration] << "\n";
   cout << "-------------------\n";
   clusterSizes[iteration]=clusterSize;
   clusterSizesMin[iteration]=min;
   clusterSizesMax[iteration]=max;
   energies[iteration]=energy;
   iteration++;
}


TGraph *es = new TGraph(iteration, energies, clusterSizes);
TGraph *minis = new TGraph(iteration, energies, clusterSizesMin);
TGraph *maxis = new TGraph(iteration, energies, clusterSizesMax);

TGraphErrors *EnergySize = new TGraphErrors(iteration, energies,
clusterSizes, 0, stdDev);
```

```
185
      es->SetMarkerStyle(21);
187     es->SetTitle("Cluster sizes vs energy deposition;Energy [keV/#mum];Cluster
       size [# pixels]");
      es->Draw("*A");
189     minis->Draw("same,*");
      maxis->Draw("same,*");
191     c1->Update();

193     TCanvas *c2 = new TCanvas("c2", "Cluster Sizes vs Energy errors", 200, 10,
       700, 500);

195 //   TF1 *fitFunction = new TF1("fitFunction", "[0] + [1] * log([2] * x)", 0,
       15);
    //   TF1 *fitFunction2 = new TF1("fitFunction2", "[0] + x * [1] * log([2] * x)
       ", 0, 15);
197     TF1 *fitFunction3 = new TF1("fitFunction3", "[0]*pow(x,[1])", 0, 15);

199     fitFunction3->SetParameters(6, 0.43);

201 //   fitFunction->SetLineColor(kGreen);
    //   fitFunction2->SetLineColor(kBlue);
203     fitFunction3->SetLineColor(kRed);
    //   es->Fit("fitFunction");
205 //   es->Fit("fitFunction2");
    //   es->Fit("fitFunction3");
207 //   c2->cd();

209     EnergySize->SetMarkerStyle(21);
      EnergySize->SetMarkerColor(4);
211     EnergySize->SetTitle("Cluster sizes vs energy deposition;Energy [keV/#mum];
       Cluster size [# pixels]");
      EnergySize->Draw("AP");
213     c2->Update();
      EnergySize->Fit("fitFunction3");
215 }
```

```
217 void drawComparison(){
      TCanvas *c1 = new TCanvas("c1", "Simulated vs experimental cluster size;E_{
      dep} [keV/#mum];Cluster Size", 200, 10, 700, 500);
219   Double_t x1[50],x2[50],y1[50],y2[50];
      Float_t normEdep;
221   for (Int_t i=1;i<51;i++){
        normEdep = i*14.;
223     x1[i]=i;
        x2[i]=i;
225     y1[i]=(0.9317 + 0.2744*normEdep - 0.0003392 * pow(normEdep,2) + 6.03427e
      -7*pow(normEdep,3) - 3.8137e-10*pow(normEdep,4));
        y2[i]=7.6626*pow(i,0.420307);
227     printf("x1[%d] = %.2f, y1[%d] = %.2f\n", i, x1[i], i, y1[i]);
      }
229   TGraph *es1 = new TGraph(30,x1,y1);
      es1->SetTitle("Simulated vs experimental cluster size;E_{dep} [keV/#mum];
      Cluster Size");
231   es1->SetLineColor(kRed);
      es1->SetLineWidth(2);
233   es1->Draw("AC");
      TGraph *es2 = new TGraph(30,x2,y2);
235   es2->SetLineColor(kBlue);
      es2->SetLineWidth(2);
237   es2->Draw("same,C");
      TLegend * leg = new TLegend(0.1, 0.7, 0.48, 0.9);
239   leg->AddEntry(es1, "Empirical energy-size relation","l");
      leg->AddEntry(es2, "Energy-size relation from model","l");
241   leg->Draw();
      c1->Update();
243 }

245 void drawIntensityDistribution2D(){
      Int_t width=11;
247   Int_t groupWidth=9;
      Double_t sensorDepth=14;
```

```
249     Double_t pixelSize=3e-05;
        Double_t samplePixelSize=pixelSize/groupWidth;
251     TCanvas *c1 = new TCanvas("c1", "Intensity Distribution",700,500);
        Double_t x[9*11],y[9*11];
253     TMatrixD intensityMap=createIntensityMap((width+1)*groupWidth, sensorDepth,
         samplePixelSize);
        for (Int_t i=0; i<9*11;i++){
255        y[i]=intensityMap[4*11+6][i];
           x[i]=i;
257     }
        TGraph *ig = new TGraph(9*11,x,y);
259     ig->SetTitle("Intensity distribution;position [pixel];intensity [electrons]
        ");
        ig->SetLineColor(kRed);
261     ig->SetLineWidth(2);
        ig->Draw("AC");
263     gPad->SetLogy();
        gPad->SetGridy();
265     c1->Update();
    }
```

code/MakeClusters.C

B.3 Analysis of Clusters Related to the DTC

This code makes use of the previously written code [2] and applies results from the codes in B.1 and B.2.

```
ï»¿#include <ctime>
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>

#include <TH2F.h>
#include <TH2.h>
#include <TH3.h>
#include <TPolyLine3D.h>
#include <TPolyMarker3D.h>
#include <TRandom3.h>
#include <TLatex.h>
#include <TStyle.h>
#include <TPaveText.h>
#include <TAxis3D.h>
#include <TCanvas.h>
#include <TGraph.h>
#include <TGraphErrors.h>
#include <TEllipse.h>
#include <TLegend.h>
#include <TStopwatch.h>
#include <TPaveStats.h>
#include <TView.h>
#include <TLeaf.h>
#include <TArrow.h>
#include <TF1.h>
#include <Math/ProbFunc.h>

#include "Analysis/Analysis.h"
#include "GlobalConstants/Constants.h"
#include "GlobalConstants/MaterialConstants.h"
```

```
#include "GlobalConstants/Misalign.h"
#include "Classes/Track/conversionFunctions.h"
#include "Classes/Track/Tracks.h"
#include "Classes/Track/Track.h"
#include "Classes/Hit/Hits.h"
#include "Classes/DataInterface/DataInterface.h"
#include "HelperFunctions/Tools.h"
#include "HelperFunctions/getTracks.h"
#include "Cluster.h"

using namespace std;

void drawCSvEComparison(Int_t Runs, Float_t energy, Bool_t recreate){
    Int_t dataType=1;
    Int_t vDivide=4;
    Int_t skipIdx = 100;
    Tracks * tracks = loadOrCreateTracks(recreate, Runs, dataType, energy);
    Track *thisTrack;
    Cluster *cluster = nullptr;
    Float_t trackRange, trackEnergy, trackScale, trackError;
    TGraphErrors *tge = nullptr;

    Double_t clusterSizes[10000];
    Double_t eDeps[10000];

    TH2F *CSvEold = new TH2F("CSvEold", "Cluster size versus eDep; eDep [kev/#
    mum]; Cluster size [# pixels]",300,0,5,30,0,30);
    TH2F *CSvEnew = new TH2F("CSvEnew", "Cluster size versus eDep; eDep [kev/#
    mum]; Cluster size [# pixels]",300,0,5,30,0,30);

    TCanvas *c0 = new TCanvas("c0", "Bragg peak comparison", 200, 10, 700,
    1500);
    c0->Divide(2, vDivide, 0.01, 0.01);
    Int_t trackIdx=0;
    for (Int_t i=0; i<tracks->GetEntriesFast(); i++){
       if (i < skipIdx) continue;
```

```
66
        thisTrack=tracks->At(i);
68
        tge = (TGraphErrors*) thisTrack->doRangeFit();
70      if (!tge) continue;
        trackEnergy = thisTrack->getFitParameterEnergy();
72      trackRange = getTLFromEnergy(trackEnergy);
        trackScale= thisTrack->getFitParameterScale();
74      trackError= thisTrack->getFitParameterError();
        for (Int_t j=0; j<thisTrack->GetEntriesFast(); j++) {
76         cluster = thisTrack->At(j);
           if (getEnergyAtTL(trackEnergy, getLayerPositionmm(thisTrack->getLayer
      (j)))==0.){continue;}
78         if (cluster->getSize()<1){continue;}
           CSvEold->Fill(getStoppingPower(getEnergyAtTL(trackEnergy,
      getLayerPositionmm(thisTrack->getLayer(j)))),cluster->getSize());
80      }
        if(trackIdx<vDivide*2){drawIndividualGraphs(c0, tge, trackEnergy,
      trackScale, trackError, trackIdx++);}
82
        tge = nullptr;
84
        tge = (TGraphErrors*) thisTrack->doRangeEvenFit();
86      if (!tge) continue;
        trackEnergy = thisTrack->getFitParameterEnergy();
88      trackRange = getTLFromEnergy(trackEnergy);
        trackScale= thisTrack->getFitParameterScale();
90      trackError= thisTrack->getFitParameterError();
        for (Int_t j=0; j<thisTrack->GetEntriesFast(); j++) {
92         cluster = thisTrack->At(j);
           if (getEnergyAtTL(trackEnergy, getLayerPositionmm(thisTrack->getLayer
      (j)))==0.){continue;}
94         if (cluster->getSize()<1){continue;}
           CSvEnew->Fill(getStoppingPower(getEnergyAtTL(trackEnergy,
      getLayerPositionmm(thisTrack->getLayer(j)))),cluster->getSize());
96      }
```

```
        if (trackIdx<vDivide*2){drawIndividualGraphs(c0, tge, trackEnergy,
    trackScale, trackError, trackIdx++);}
98      }

100     TCanvas *c1 = new TCanvas("c1", "Cluster size vs deposited energy (exp.
    data), old vs new", 200, 10, 700, 500);
        c1->Divide(2,1, 0.01, 0.01);
102
        c1->cd(1);
104     CSvEold->Draw("COLZ");
        c1->cd(2);
106     CSvEnew->Draw("COLZ");
    }
108

110 void drawClusterSizeVsEDep(Int_t Runs, Float_t energy, Bool_t recreate){
        Int_t dataType=1;
112     Tracks * tracks = loadOrCreateTracks(recreate, Runs, dataType, energy);
        Track *thisTrack;
114     Cluster *cluster = nullptr;
        Float_t trackRange, trackEnergy;
116     TGraphErrors *tge = nullptr;

118     Double_t clusterSizes[10000];
        Double_t eDeps[10000];
120
        TCanvas *c1 = new TCanvas("c1", "Cluster size vs deposited energy (exp.
    data)", 200, 10, 700, 500);
122
        TH2F *CSvE = new TH2F("CSvE", "Cluster size versus energy deposit; Edep [
    kev/#mum]; Cluster size [# pixels]",50,0,5,30,0,30);
124     for (Int_t i=0; i<tracks->GetEntriesFast(); i++){
           thisTrack=tracks->At(i);
126        tge = (TGraphErrors*) thisTrack->doRangeEvenFit();
           if (!tge) continue;
128        trackEnergy = thisTrack->getFitParameterEnergy();
```

```
      trackRange = getTLFromEnergy(trackEnergy);
130   for (Int_t j=0; j<thisTrack->GetEntriesFast(); j++) {
         cluster = thisTrack->At(j);
132      if (getEnergyAtTL(trackEnergy, getLayerPositionmm(thisTrack->getLayer
    (j)))==0.){continue;}
         if (cluster->getSize()<1){continue;}
134      CSvE->Fill(getStoppingPower(getEnergyAtTL(trackEnergy,
    getLayerPositionmm(thisTrack->getLayer(j)))),cluster->getSize());
      }
136  }
    CSvE->Draw("COLZ");
138  gPad->SetLogz();
    Float_t x1[50],x2[50],y1[50],y2[50];
140  Float_t normEdep;
    for (Int_t i=1;i<51;i++){
142    normEdep = (i/10.)*14.;
       x1[i]=i/10.;
144    x2[i]=i/10.;
       y1[i]=3.1*(0.9317 + 0.2744*normEdep - 0.0003392 * pow(normEdep,2) +
    6.03427e-7*pow(normEdep,3) - 3.8137e-10*pow(normEdep,4));
146    y2[i]=6.4778*pow(i/10.,0.404476);
    }
148  TGraph *es1 = new TGraph(50,x1,y1);
    es1->SetLineColor(kRed);
150  es1->SetMarkerColor(kRed);
    //es1->Draw("same,*");
152  TGraph *es2 = new TGraph(50,x2,y2);
    es2->SetLineColor(kBlue);
154  es2->SetMarkerColor(kBlue);
    //es2->Draw("same,*");
156  c1->Update();
//   TGraph *es = new TGraph(iteration, eDeps, clusterSizes);
158
//   es->Draw("*A");
160 //   c1->Update();
}
```

```
162
164
   Float_t getStoppingPower(Float_t protonEnergy){
166    return 43.95*pow(protonEnergy,-0.748);
   }
168
   Float_t getEdepFromCS(Float_t eDep, Int_t lambda){
170    if (!lambda%5){return 0;}
       Int_t choice = (lambda-30)/5;
172    Float_t p0[8]={6.16421e+00, 6.66920e+00, 7.24521e+00, 7.66260e+00, 8.07114e
       +00, 8.48959e+00, 8.72351e+00, 8.99963e+00};
       Float_t p1[8]={3.81925e-01, 4.00565e-01, 4.09395e-01, 4.20307e-01, 4.29850e
       -01, 4.37136e-01, 4.47309e-01, 4.54506e-01};
174    return p0[choice]*pow(eDep, p1[choice]);
   }
176
   Float_t getp0(Int_t lambda){
178    if (!lambda%5){return 0;}
       Int_t choice = (lambda-30)/5;
180    Float_t p0[8]={6.16421e+00, 6.66920e+00, 7.24521e+00, 7.66260e+00, 8.07114e
       +00, 8.48959e+00, 8.72351e+00, 8.99963e+00};
       return p0[choice];
182 }
   Float_t getp1(Int_t lambda){
184    if (!lambda%5){return 0;}
       Int_t choice = (lambda-30)/5;
186    Float_t p1[8]={3.81925e-01, 4.00565e-01, 4.09395e-01, 4.20307e-01, 4.29850e
       -01, 4.37136e-01, 4.47309e-01, 4.54506e-01};
       return p1[choice];
188 }
190
   void drawMoreClusterSizeVsEDep(Int_t Runs, Float_t energy, Bool_t recreate){
192    Int_t dataType=1;
       Tracks * tracks = loadOrCreateTracks(recreate, Runs, dataType, energy);
```

```
194     Track *thisTrack;
        Cluster *cluster = nullptr;
196     Float_t trackRange, trackEnergy;
        TGraphErrors *tge = nullptr;
198
        Double_t clusterSizes[10000];
200     Double_t eDeps[10000];

202     TCanvas *c1 = new TCanvas("c1", "Cluster size vs deposited energy (exp.
        data)", 200, 10, 900, 700);

204     TH2F *CSvE = new TH2F("CSvE", "Cluster size versus eDep; eDep [kev/#mum];
        Cluster size [# pixels]",300,0,5,30,0,30);
        CSvE->Draw("COL");
206     for (Int_t i=0; i<tracks->GetEntriesFast(); i++){
           thisTrack=tracks->At(i);
208        tge = (TGraphErrors*) thisTrack->doRangeEvenFit();
           if (!tge) continue;
210        trackEnergy = thisTrack->getFitParameterEnergy();
           trackRange = getTLFromEnergy(trackEnergy);
212        for (Int_t j=0; j<thisTrack->GetEntriesFast(); j++) {
              cluster = thisTrack->At(j);
214           if (getEnergyAtTL(trackEnergy, getLayerPositionmm(thisTrack->getLayer
        (j)))==0.){continue;}
              if (cluster->getSize()<1){continue;}
216           CSvE->Fill(getStoppingPower(getEnergyAtTL(trackEnergy,
        getLayerPositionmm(thisTrack->getLayer(j)))),cluster->getSize());
           }
218     }
        TF1 **functions = new TF1*[8];
220     Int_t iterator = 0;
        TLegend *leg = new TLegend(0.1, 0.7, 0.48, 0.9);
222     char fname[20];
        char lname[20];
224     sprintf(fname, "lambda=%d", 60);
        sprintf(lname, "#lambda = %de-06 #mum", 60);
```

```
226     functions[iterator]=new TF1(fname, "[0]*pow(x,[1])",0.8,30);
        functions[iterator]->SetParameter(0, getp0(60));
228     functions[iterator]->SetParameter(1, getp1(60));
        functions[iterator]->SetLineColor(kRed);
230     functions[iterator]->Draw();
        functions[iterator]->SetTitle("Energy-Cluster Size Relation;Energy [kev/#
        mum];Cluster Size [# pixels]");
232     iterator = 1;
        for (Int_t lambda = 30; lambda<60; lambda+=10){
234        char fname[20];
           char lname[20];
236        sprintf(fname, "lambda=%d", lambda);
           sprintf(lname, "#lambda = %de-06 #mum", lambda);
238        functions[iterator]=new TF1(fname, "[0]*pow(x,[1])",0.8,30);
           functions[iterator]->SetParameter(0, getp0(lambda));
240        functions[iterator]->SetParameter(1, getp1(lambda));
           functions[iterator]->SetLineColor(kRed-1*iterator);
242        functions[iterator]->Draw("same");

244        iterator++;
        }
246     leg->AddEntry(functions[0], "#lambda = 60e-06 #mum", "l");
        leg->AddEntry(functions[3], "#lambda = 50e-06 #mum", "l");
248     leg->AddEntry(functions[2], "#lambda = 40e-06 #mum", "l");
        leg->AddEntry(functions[1], "#lambda = 30e-06 #mum", "l");
250     leg->Draw();

252     c1->Update();
    }
```

code/FocalCluster.C