Characterization of Optical Properties of Ion Irradiated Diamond: Progress Report

ESROP Global Research Project | National University of Singapore

Name: Liza Zoubakina
Supervisor: Prof. Andrew Bettiol
Dates of project: May 13th, 2019 – August 4th, 2019
Research partner: Jason Yuan
Accompanying graduate students: Mike, Kim, and Chenyuan

Introduction

This progress report provides an overview of what was accomplished during my exchange trip to National University of Singapore (NUS). A list of tasks accomplished each day is provided in the "Daily Progress Log" sections, organized by date, whereas this "Introduction" section provides an overall summary of my work, with references to the corresponding dates in the daily progress log.

Daily Progress Log

Week 1: May 13th to 17th

Settled into the lab at the Physics Department of NUS. Toured various facilities (CIBA lab, my own lab) and met with the people who I will be working with over the next few months.

Graduate students: Kim (optical properties), Mike (thermal properties)

Professor: Andrew Bettiol, worked at NUS for over 20 years. Very chill and interesting dude. Will talk at length about diamonds and quantum physics if you ask him the right questions.

Otherwise, for the first week, my time was devoted to two main activities:

- Reading assigned papers (from Andrew) and learning concepts
  - With a focus on nitrogen vacancy (NV) centres located in synthetic diamonds
- Installing and learning how to use ROOT and KDetSim
  - Software meant to simulate the electrical charge distribution through a diamond.

ROOT is a simulation software used by CERN. It is hard to download and install on your laptop but I have found several sources that help to guide you in this installation. This guide is meant for Windows 10 (but you have to install a Linux Subsystem on your laptop). The steps are below:

1. Enable and install Linux Subsystem on Windows 10 (https://developerinsider.co/stepwise-guide-to-enable-windows-10-subsystem-for-linux/)
2. Install ROOT (https://medium.com/@blake.leverington/installing-cern-root-under-windows-10-with-subsystem-for-linux-beta-75295defc6d4)

It will take awhile for your computer to install and compile ROOT (took about 3 hours for my computer to build it), so don't worry.

Next, once ROOT is installed on your laptop (make sure it's at least Version 6), you should try installing KDetSim.

1. Clone or download KDetSim from this GitHub site: https://github.com/pitzl/KDetSim
2. Make sure you do this in Ubuntu, so for example, type in Ubuntu: "git clone https://github.com/pitzl/KDetSim"
3. Next, follow the steps on the KDetSim readme file. Make sure you edit the Makefile by changing the 17 to an 11 (it's near the top of the document).
4. Also, manually create two directories in the KDetSim folder labeled "obj" and "lib".
5. Now, assuming your KDetSim has compiled (the "make" action has worked), make sure you type in "x. Pixel_1.c" and not TestPixel. Because TestPixel does not exist.

Otherwise, you should be good to download both ROOT and KDetSim on your laptop. If you are encountering "invalid address" errors, you probably messed up your linking. Try reinstalling ROOT (I know, it's a pain), but it should work the next time around.

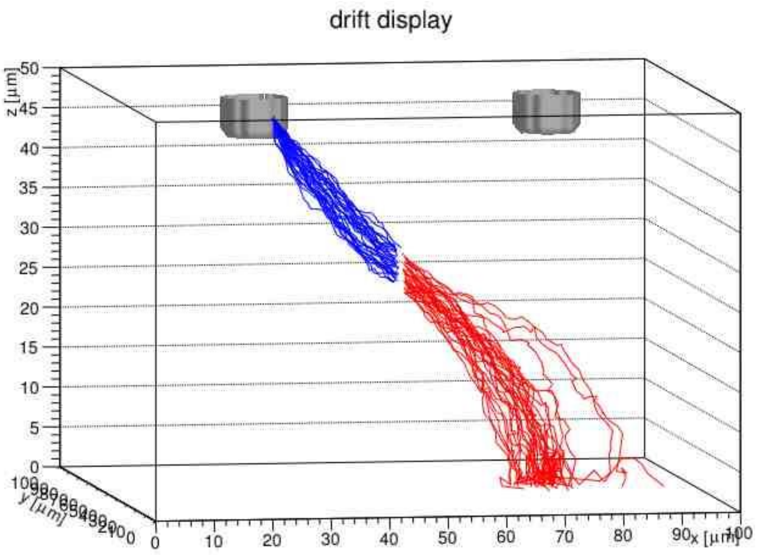
Anyways my laptop is running an Ubuntu subsystem right now, which is great! I learned a lot about how Ubuntu works. (I hate Ubuntu)

Week 2: May 21st – 24th

This was the week of coding. The week of KDetSim.

So yeah, this week we spent the entire time learning how KDetSim and Root work together as a simulation software. We also met a new graduate student, Chenyuan, who is in charge of the optical set-up.

We were tasked with simulating the drift of electrons and holes in a diamond-electrode set up. There are several electrodes placed on the top plane of the diamond surface, which act as high voltage and ground electrodes. In this experiment, the surface of the diamond layer is irradiated by numerous ions that impact the layer. This serves to initiate the movement of charge of electrons (blue) and holes (red) towards their respective electrode.

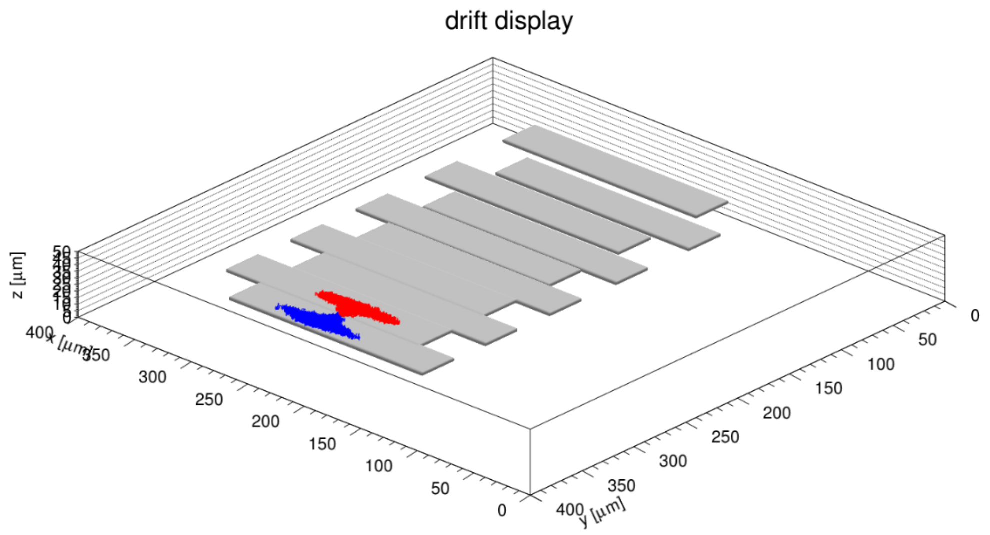


The KDetSim software is difficult to use and apply as there is not a clearly defined user manual that explains how the functions work. As well, it is coded in C++, which is basically a mix of python and C. You cannot modify the source (src) and header (inc) files without recompiling them, so we have been mostly playing around with how you can alter the parameters of the functions. So far, I have used the following files and functions:

- KDetector: this is a base class of all files in KDetSim. It is the foundations where KMaterial, KGeometry, and K3D are constructed on. In the source and header file, it contains the functions necessary for all matrix and Gaussian calculations, as well as the functions necessary for graphing tables and drawings.
- KMaterial: this source file contains the information related to the nature of the material used in the "box", it can range from silicon, air, and various types of diamond. There is also a Gain factor function as well as an Impact Ionization coefficient.
- KGeometry: this source file is essential in the construction of electrodes. It has 3 main electrode functions (ElLine, ElRectangle, and ElCylinder) which help to build the electrode. It also positions the electrodes where you want them.
- K3D: I used this source/header file to build my own simulation. I mostly worked with three example files (Test3D_0.C, Test3D_1.C, and Test3D_2.C) to figure out how I could build a set-up of my own. This 3D detector allow me to build a set-up fairly easily, compared to using KDetector (which would force me to spend more time initializing everything).

- Lots of example files: probably the aspect of KDetSim that allowed me to comprehend (at least a little) of how the files work was by looking through the example files. There are numerous files that each contain different types of simulations, which allows you to learn by example.

So far, Jason and I have accomplished the following objectives:



- Constructed the electrodes according to the experimental set-up.
- Understood the basic components of the software, and in general how C++ works
- Simulated the drift of electrons and holes to two separate electrodes.
- Graphed current vs. time and electric field potential vs. (x,y) coordinates

However, we are currently facing the following obstacles:

- Simulate multiple entry/exit points
- Simulate the effect of ion irradiation (using the ionization impact coefficient)
- Constructing the Gain factor function (in KMaterial)
- Constructing the proper graphs:
    - Charge pulse graph (basically integral of current-time graph)
    - Charge collection efficiency (CCE) versus Bias Voltage

Week 3: May 27th – 31st

This week, we focused on KDetSim and coding up the charge distribution in an ion irradiated diamond. As time went on, my team and I have learned more about the organization and implementation of KDetSim and its various classes. We have learned about changing the material and geometry of the detector and electrodes, as well as applying a particular space charge due to ion irradiation.

However, two important problems that have risen are:

- The ability to graph charge collection efficiency vs. bias voltage
- Calculating the charge multiplication factor (using the KM function in KMaterial).

So far, we have been able to plot charge collection (Q) versus x-y coordinate plane of the diamond detector material. However, we are having trouble with setting a variable voltage ranging from -300 to 300V.

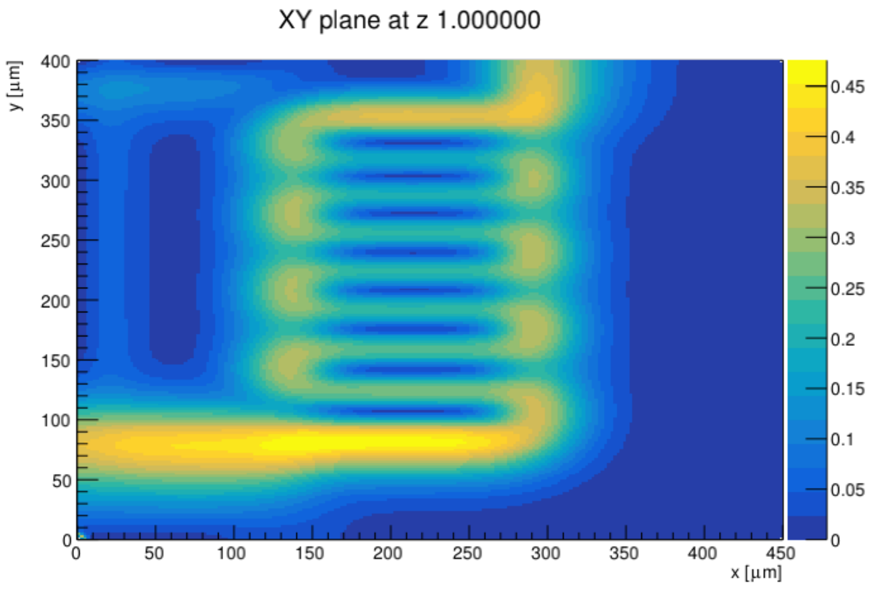


Figure 1: Electric field profile

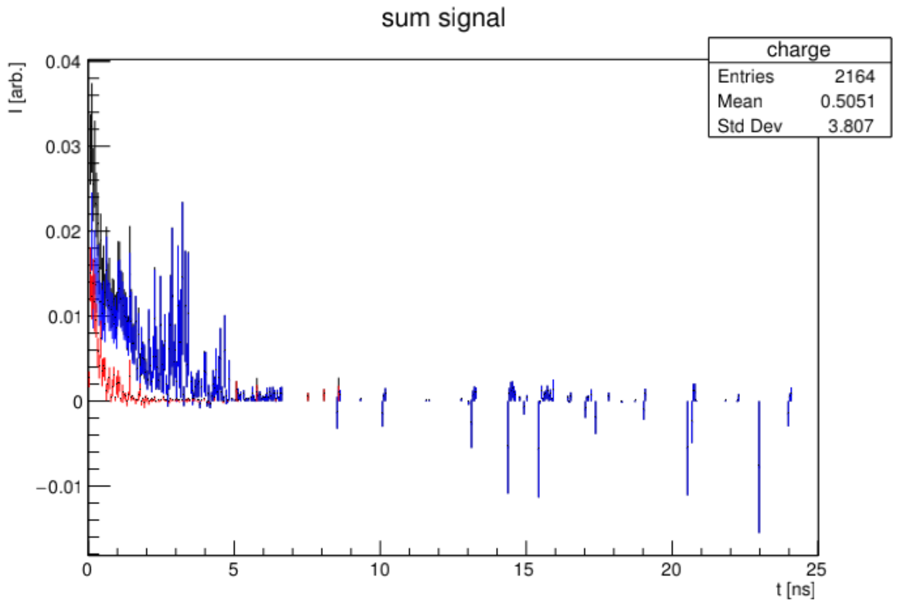


Figure 2: Current vs. Time. Blue = electrons, Red = Holes

We have looked closely through all of the example, source, header, and pixel files in order to guide our process.

Charge Collection Efficiency (CCE) = Total induced charge (Q) / Total charge generated by irradiation (q)

The files that I focused on this week came from the **Pixel** and **examples** folder:

- Pixel_1dj.C provided an example of how to integrate current and plot collected charge (Q) versus changing z-coordinates
- Pixel_9.C provided an example of charge collected (Q) versus two nested loops (x,y) coordinates.
- Vert9 to vert25.C files included files on how to plot charge collected (Q) versus Fluence (F).
- Ccevsf.C described on how to plot charge collection efficiency using qvsf and the vert files. It focused on charge collection efficiency (CCE) per vertical distance (z) in a pixel.

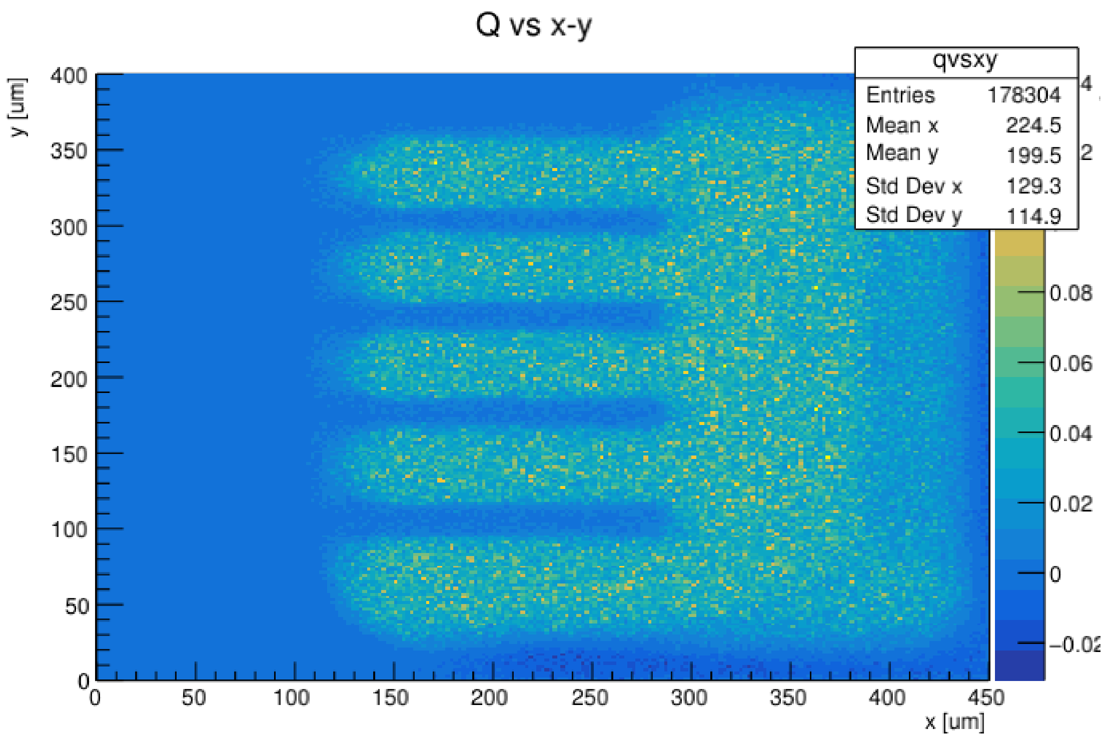


Figure 3: Charge Collection per x-y plane (increment of one). Here, one can see the silhouette of the collector electrode, as it collects the induced charge caused by the free moving charge generated from ion irradiation.

Week 4: June 3rd – 7th

So good news! Both Gregor Kramberger (creator of KDetSim) and Daniel Pitzl (one of the collaborators) have replied back to our emails on Monday. Jason and I sent back a description of our code and related pictures. An overview of our email is presented below.

Current state of our simulation:

- Electrode placement: We have presently constructed 12 co-planar rectangular electrodes using the K3D Detector class. They are as pictured in our attached images (see Drift Display). They each form two branches, the left one is High Voltage and the right branch is the Collector Ground electrode (see XY plane at z 1.0 for the abs. Electric field).
- We are not sure whether we have set-up every voxel correctly in our simulation grid, as we assumed that the boundaries of K3D are reflective by default.
- Ion Irradiated Diamond: We are trying to simulate the ion irradiation of the diamond by modeling the charge collection caused by a surface-wide ion irradiation (seen in the QvsXY picture).
- We are currently assuming that the ion only travels through half-way the thickness of the diamond detector (25 um), as stated in our entry/exit points in our example code (diam9.C). I understand that we may have to input our values into KAlpha (modify and remake the KMaterial source file?).
- We have made many assumptions and simplifications, which we aim to reduce by implementing the functions and code correctly to execute more accurate simulations.
- We plan to have a bias voltage ranging from -300 to 300V. After looking through the pixel files and examples of codes in other scientific papers, we understand that it may be necessary to create multiple files with different values of bias voltage. After constructing each individual file, we can then plot CCE vs. Bias Voltage.

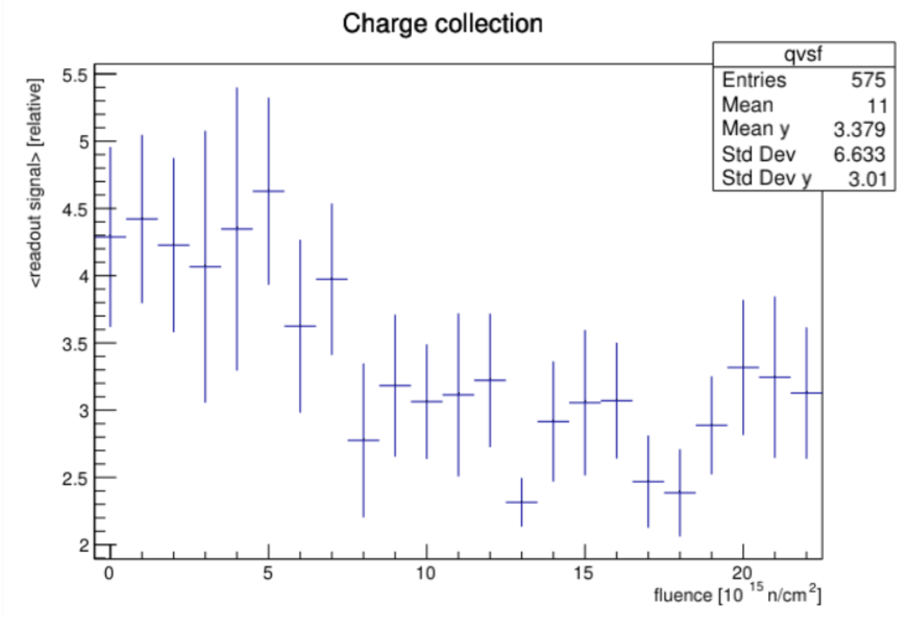
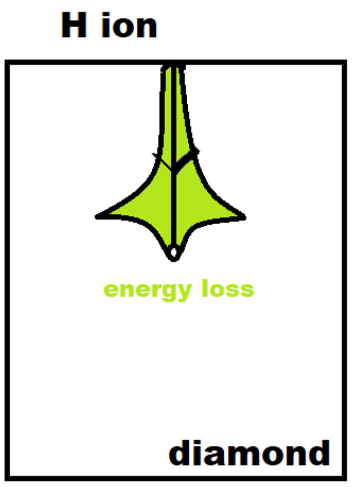


Figure 4: Charge collection (Q) versus. Fluence (F, optical irradiation density)

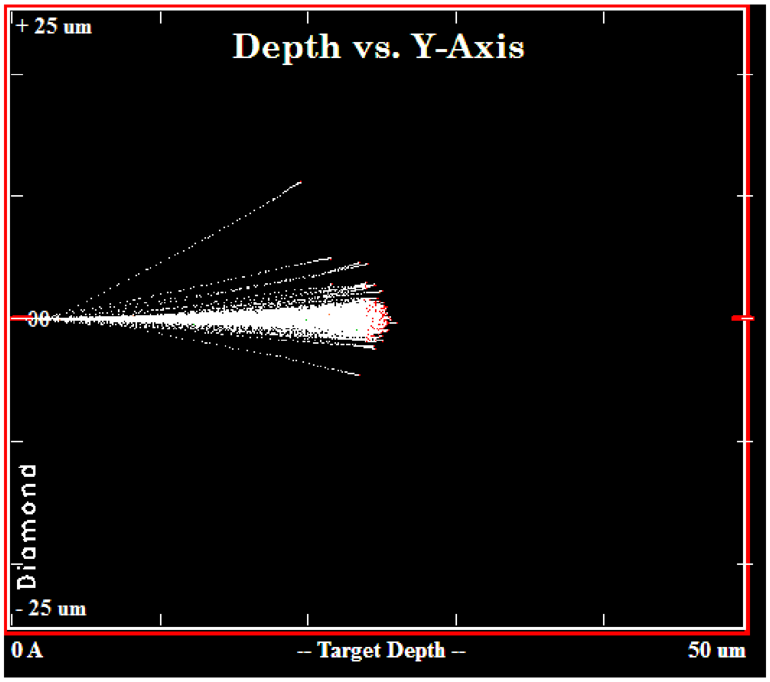
In addition, I started using a software called SRIM (Stopping and Range of Ions in Matter, http://www.srim.org/). This was suggested by Prof. Andrew as a method to obtain ionization values for the deposition of ions into the diamond. This value would then be divided by 13 eV so as to obtain the charge values for an electron-hole pair.

Therefore, my task this week is to model the charge distribution along the pathway of the ion. Here are a few factors that I have found out about already:

- Currently, I am assuming that there is a uniform charge distribution generated around the ion as it goes through the diamond in KDetSim.
- I want to generate a varying charge distribution as the ion goes deeper through the layers of the diamond (currently, the ion range is about 25 um, while the diamond thickness is 50 um).
- From what I understood before, I must input my physics values into the KAlpha function in order to model the deposition of ions into the diamond.
- Is it possible to generate a varying energy loss generated by the deposition of an ion?
- I was thinking of dividing the energy curve into several layers of depth, and generating separate charge distribution parameters for each division of the depth.



This week, I also learned how SRIM works. This software was created a long time ago (the latest version was released in 2013), therefore it is not very appealing to look at. It's nice to go back in time, and see how simulations were run before. Despite its age, the SRIM tutorials are very helpful and descriptive. There are bugs in the simulations that cause it to randomly shut down, however overall, it is a straightforward and easy-to-run modeling software (at least, so far).



To model the ionization of the ions hitting the diamond layer, I modeled three different layers – gold, chromium, and diamond. The ion (Hydrogen) and ion energy (2 MeV) are inputted into the calculation, as well as the layer of diamond is essentially a layer of Carbon, with special modifications:

- Density = 3.5 g/cm^3
- Displacement energy = 45

Figure 5: 2D Graph of the ion paths. The calculated range was about 24.5 um, which is what Andrew correctly predicted.

After these inputs, I ran the monolayer and damage cascades simulations (which takes longer by 10x, probably not necessary but I thought to be safe). Here are a few images that demonstrate the results of the simulations.

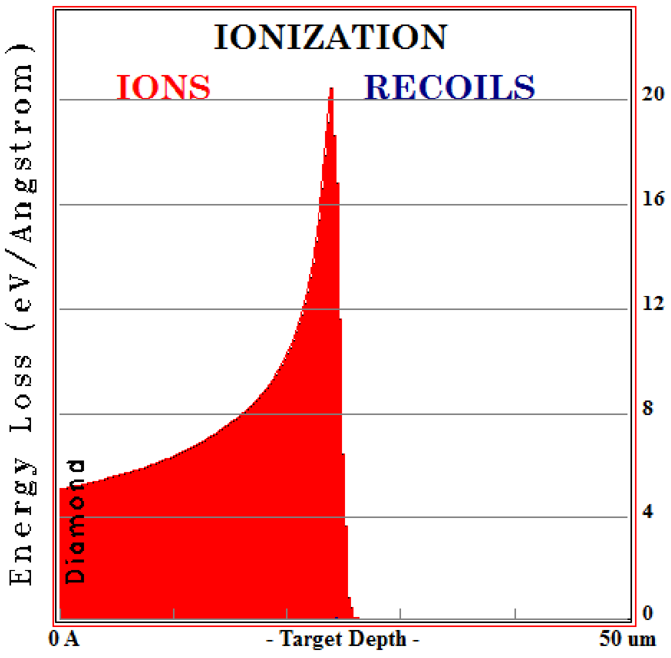


Figure 6: Graph of the ionization/energy loss. These values should be divided by 13.3eV to obtain e-h pair values

Week 5: June 10th – 14th

Still no reply from either Gregor Kramberger or Daniel Pitzl. I sent a follow-up email to both, detailing our progress and outlining our questions about the modeling of charge distribution around the ion.

After collecting data on SRIM, I downloaded the IONIZ.txt and RANGE.txt files and analysed them using Excel. I copy and pasted the text data and converted it into columns (using Text → Column function) and divided the eV values by 13.3 eV (the energy required to generate an electron-hole pair). Then, the data was offset by every 2um so that I could make an accurate histogram.

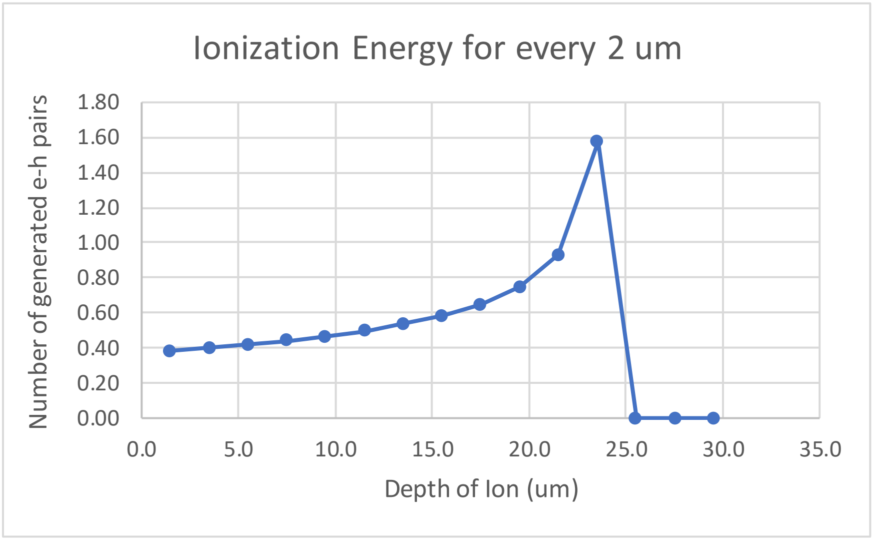


Figure 7: Ionization energy generated from SRIM. This data was calculated via a Monolayer Collision simulation with Gold-Chromium-Diamond layers.

However, after turning on the Landau function (-1→0), I soon realized that this also modeled the curve of energy loss of an ion piercing through matter. At least, this is my guess. After turning on Landau, I received the following charge collection graph.

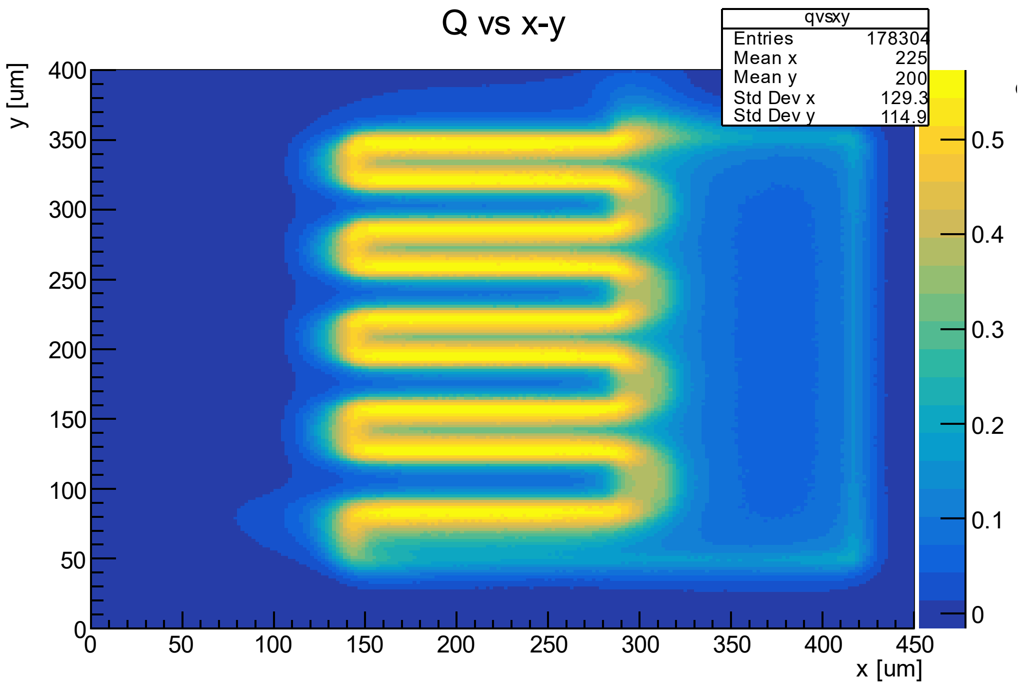


Figure 8: Charge collection graph generated from turning on the Landau function. One can see a more refined, smoother image with charge collection peaking around the electrode edges. This is more similar to previous simulations conducted in the past.

For the last few days, we were stuck in a standstill. No reply from either Daniel Pitzl or Gregor Kramberger. The majority of the time was spent on downloading MATLAB on the supercomputer or rebooting CentOs 7/Windows 10.

We were also focused on constructing Voltage Bias vs. Charge Collection graphs. Therefore, I decided to organize our generated images on this document for presentation.

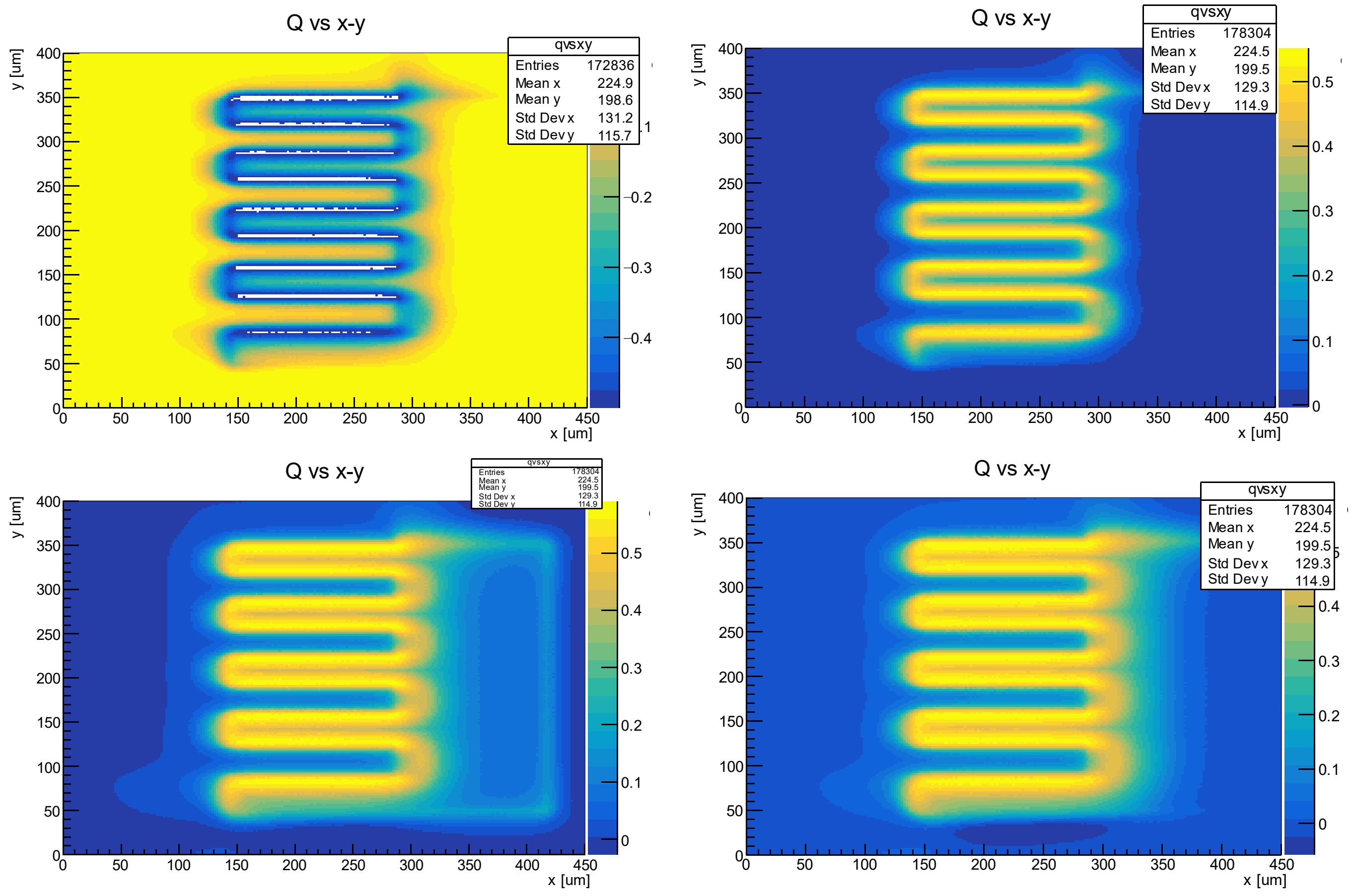


Figure 9: Charge collection vs. X-Y Plane Graphs generated for the following bias voltages [-50V, 50V, 500V, 900V]. There are no major differences in charge collection, but there is a greater dispersal of charge throughout the electrodes.

Week 6: June 17th – 21st, 2019

Still no reply from Dr. Kramberger or Dr. Pitzl. Sent an additional follow-up email to Dr. Kramberger on Friday, but to no avail.

We began modifying the Charge collection vs. Fluence code sections (Q vs. F) to model the relationship between charge and bias voltage. Jason and I achieved two different graphs that were both accurate to the following setups:

a) sandwich detector

b) inter-digitated detector

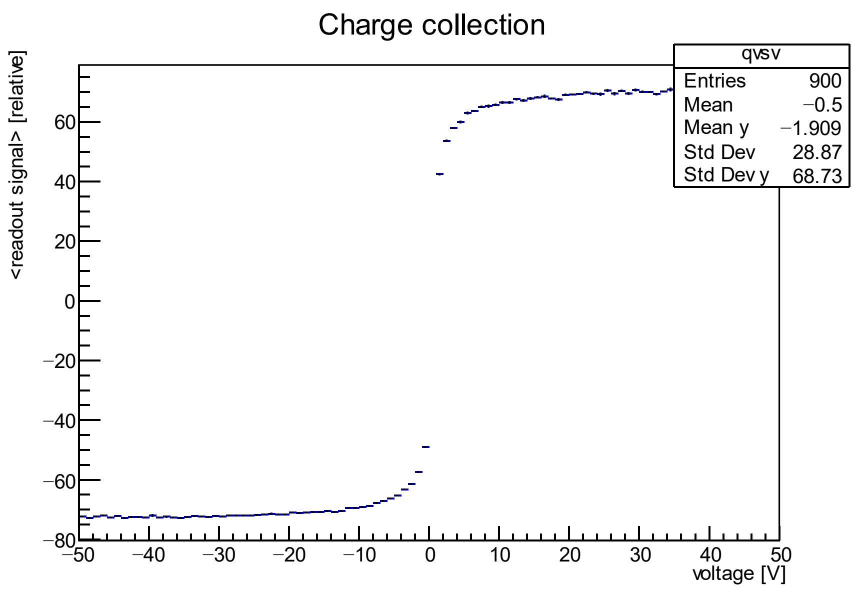


Figure 10: Example of Charge vs. VBias for the inter-digitated detector. It is noticeable that there is no charge multiplication yet.

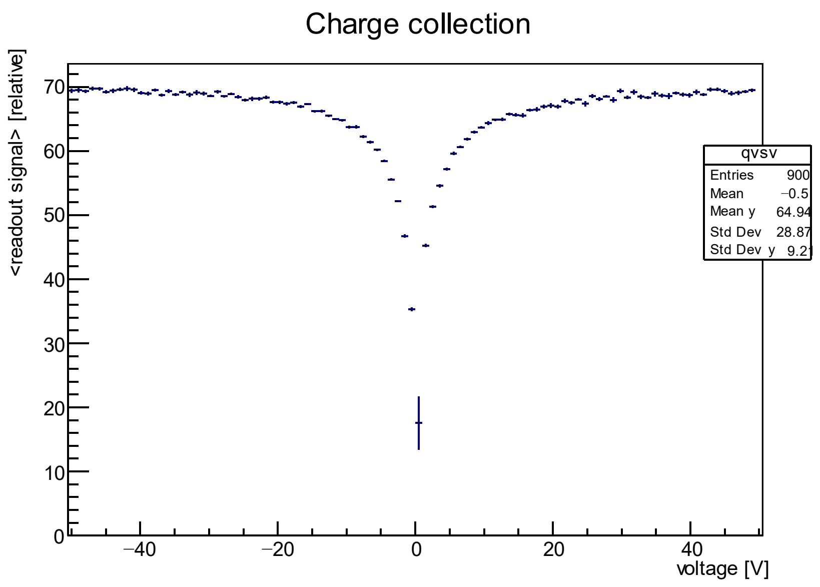


Figure 11: Another example of Charge vs. VBias. This time, the graph is collecting absolute values of induced charge. The saturation signal appears to be at 70%.

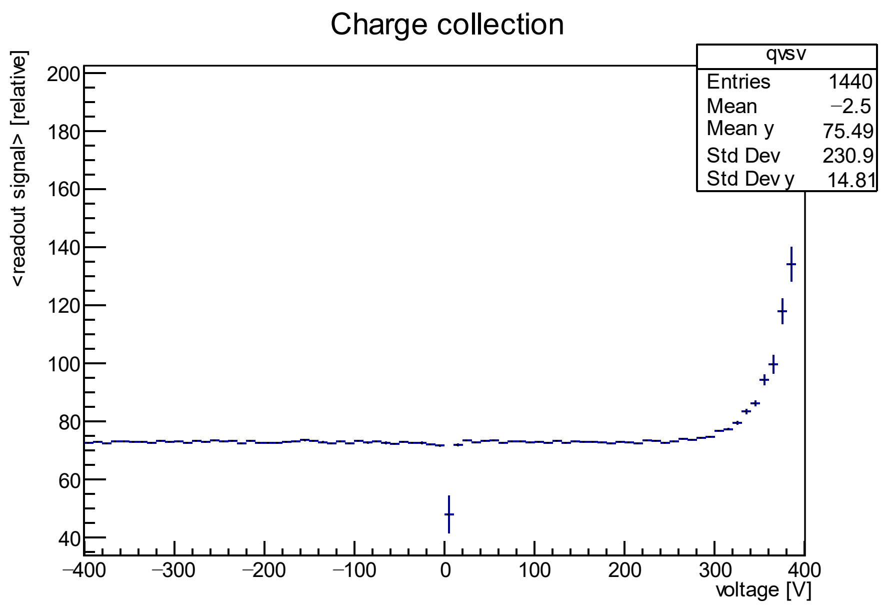


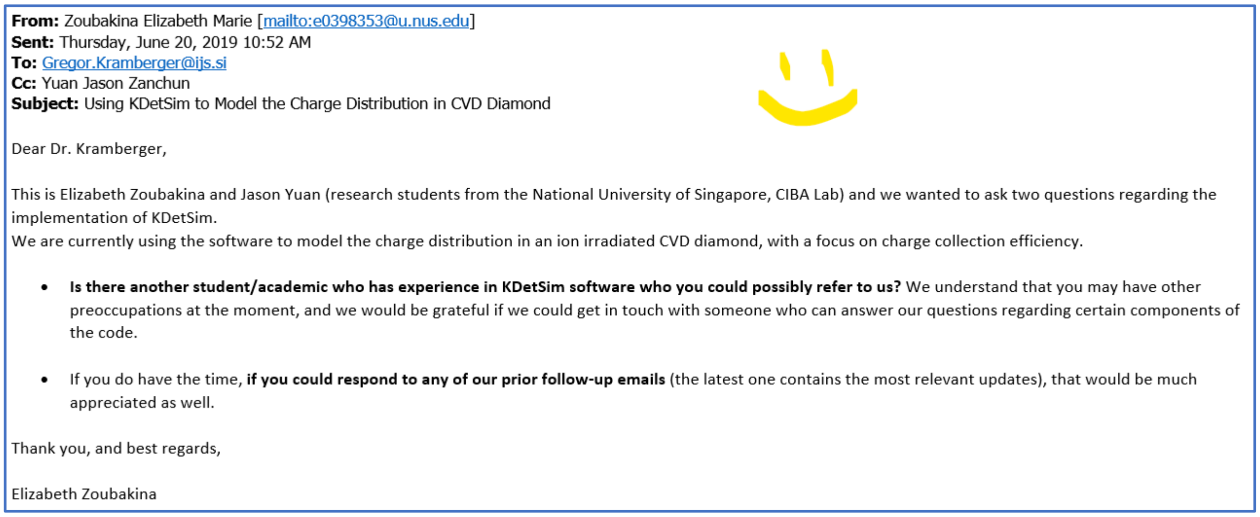
Figure 12: Example of Charge Multiplication. Multiplication occurs at voltage values above 350V. However, there is no multiplication occurring for electrons. This may be due to a mistake in programming the simulation (most likely due to boundary conditions or errors in electrode set-up).

We are planning to call either one of the KDetSim developers on Friday.

Otherwise, I began working through **LABView** Tutorials by watching videos and completing review tests. LABView is a graphic, high-level software that allows you to carry out signal processing and data analysis.

Week 7: June 24th to 28th, 2019

This has been a slow week. However, we have received a reply from Dr. Kramberger! After three weeks of tedious work and waiting, we have received the call to the great messiah. The developer of KDetSim. The key to receiving this email was to write a short, succinct email that asked two straightforward questions, and which emphasised with his position.



From: Zoubakina Elizabeth Marie [mailto:e0398353@u.nus.edu]
Sent: Thursday, June 20, 2019 10:52 AM
To: Gregor.Kramberger@ijs.si
Cc: Yuan Jason Zanchun
Subject: Using KDetSim to Model the Charge Distribution in CVD Diamond

Dear Dr. Kramberger,

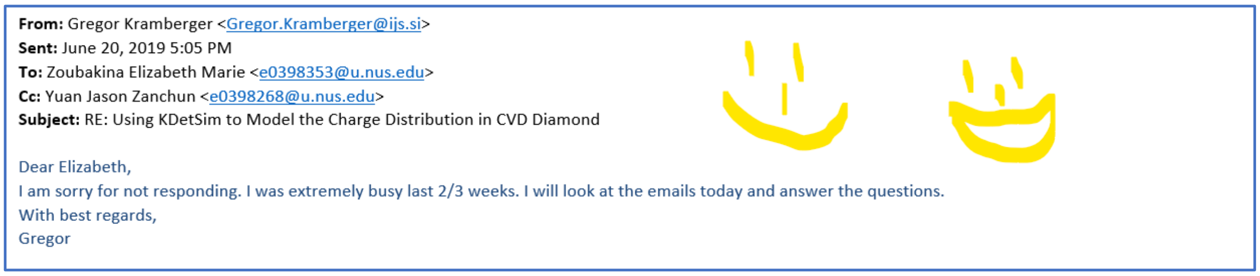
This is Elizabeth Zoubakina and Jason Yuan (research students from the National University of Singapore, CIBA Lab) and we wanted to ask two questions regarding the implementation of KDetSim.
We are currently using the software to model the charge distribution in an ion irradiated CVD diamond, with a focus on charge collection efficiency.

- **Is there another student/academic who has experience in KDetSim software who you could possibly refer to us?** We understand that you may have other preoccupations at the moment, and we would be grateful if we could get in touch with someone who can answer our questions regarding certain components of the code.
- If you do have the time, **if you could respond to any of our prior follow-up emails** (the latest one contains the most relevant updates), that would be much appreciated as well.

Thank you, and best regards,

Elizabeth Zoubakina

Figure 13: The email which prompted Dr. Kramberger to respond. I emailed this on Friday, near the mid-day.



From: Gregor Kramberger <Gregor.Kramberger@ijs.si>
Sent: June 20, 2019 5:05 PM
To: Zoubakina Elizabeth Marie <e0398353@u.nus.edu>
Cc: Yuan Jason Zanchun <e0398268@u.nus.edu>
Subject: RE: Using KDetSim to Model the Charge Distribution in CVD Diamond

Dear Elizabeth,
I am sorry for not responding. I was extremely busy last 2/3 weeks. I will look at the emails today and answer the questions.
With best regards,
Gregor

Figure 14: Response from Dr. Kramberger. He appeared to be very busy.

Otherwise, Jason and I have been focusing on different parts of our code. I started combining two aspects of my code, the QvsXY charge collection plane and the QvsV graph. By looping through an array of voltages, I constructed a 1D histogram (TH1I) which outputs the max CCE and graphs it against its respective voltage.

We have also received a detailed email from Dr. Kramberger. Unfortunately, it was not as helpful as we expected it to be. He has written and sent us two new functions:

- UserIonization();
- ShowUserIonization();

And these functions calculate the ionization values generated from ion irradiation. However, this is essentially what SRIM calculations have already provided us.

As well, Kramberger's code only functions in accordance with the Kramberger version of KDetSim software. Throughout this time, we have been using Daniel Pitzl's GitHub version of KDetSim.

Therefore, I sent Daniel Pitzl a follow-up email, which essentially contains a snippet of my code and a few questions:

I have been modifying the following Pixel files: Pixel_9.C, vert.C files, and ccevsf.C.

- **Q,** to my understanding, is the induced charge on the electrodes
  (double q = det->sum->Integral)
- **Ndiv** is the number of buckets created in MipIR, which determines the charge generated (e-h pairs) from ion irradiation
- I want to calculate **CCE**
  (total induced charge **Q** divided by total charge generated by irradiation **q**)
- How do I find **q,** which is the total charge generated by irradiation?

```
  TCanvas cQ;
  cQ.SetTitle("Charge collected per x-y plane, default space charge, landau on");
//modelling collection of charge as the diamond detector is irradiated by many H+
ions
  TH1I hq( "q" , "charge; charge [ke]; events", 450, 0, 450) ;
  TProfile2D qvsxy( "qvsxy", "Q vs x-y;x [um];y [um];<Q>", 225, 0, 450, 200, 0, 400,
-0.5, 99.5);  // (xbins, xmin, xmax, ybins, ymin, ymax, zmin, zmax)

  det->SetAverage(1);

  for (double x = 1; x < 449; x += 1) {
//looping through x and y to traverse the plane, assumes ion enters through surface
and stops at 25um (halfway through detector thickness)
          det->enp[0] = x;
          det->exp[0] = x;
          for (double y = 1; y < 399; y += 1) {
                  det->enp[1] = y;
                  det->exp[1] = y;

                  det->MipIR(ndiv);
                  double q = det->sum->Integral();
                  q /= ndiv;                    // does this calculate CCE or just
normalizes q?
                  cout << x
                  << "  " << y
                  << ": " << q
                  << endl;
                  hq.Fill(q);
                  qvsxy.Fill(x, y, q);
                  }
          }

          qvsxy.Draw("COLZ");
```

Week 8: July 1st to July 5th, 2019

Note: Went to Thailand on the 4th to 7th.