Executive Summary

In order to design effective modern wireless systems, knowledge of the medium where signals propagate (the channel) is required. In particular, the statistical locations of radio wave reflectors (scatterers) in the environment is essential to evaluate and design systems. The acquisition of this data is done through devices known as channel sounders. Currently, channel sounders are large, laborious, and expensive (hundred thousand dollar range) devices.

The main goal of our project is to build and test a small and inexpensive (thousand dollar range) channel sounder that creates an image showing the exact locations of radio scatterers in the ISM band (903 – 928 MHz), so that accurate channel models may be constructed.

The sounder consists of a stationary transmitter and a vehicular receiver. The receiver, whose positions are known at all times, captures and logs the signal. In post-processing, received data is first transformed into a ranged-compressed history. Then, a synthetic aperture radar imaging algorithm known as Back Projection is employed to recover the scatterer locations from this range-compressed history. The receiver and transmitter consist of USRPs (programmable radio transceivers), Rb clocks (for synchronization), GPS receivers (for position and time synchronization), and mini-computers for control and data storage. The channel sounder's center frequency is 915 MHz, bandwidth is 25 MHz, theoretical spatial resolution is 12 m, and costs around $12000.

Several lab tests, including direct connection and antenna transmissions, were conducted to verify transmission and reception of signals. During static testing using antenna, the system was able to detect a 2 square-meters metal plate target at 150 m total range. However, tests also revealed drifting offset in GPS time synchronization and a bottleneck in receiver data storage. Finally, fields tests were done along a 200 m section of 16th avenue on the west of Pacific Spirit Park in Vancouver. The received data was processed to yield an image of the target environment that corresponded reasonably well to a map of the area.

Overall, we recommend that work continue on this project, with some modifications. Current directions of improvement are investigating better position tracking using vehicle speed sensors, finding a reliable way to synchronize the transmitter and receiver, and acquiring a faster data storage medium.

Contents

List Of Figures

List Of Tables

Glossary

Fermi energy	Fermi energy of the SEmaterials is the position of the highest occupied state relative to the bottom of the conduction band.
Back Projection	A time-domain algorithm takes advantage of the changing Doppler shift over time to map out relative locations of scatterers
Channel	In telecommunications, the physical medium used to transmit a signal
Channel Model	A description of a physical channel, including scatterers, distortion effects, and fading
DAC	Digital to Analog Converter
DRGPS	Dead Reckoning GPS, a device that combines GPS information and vehicle speed information to gauge position accurately
Fading	When 2 signals of different phase interfere, causing destructive interference
GPS	Global Positioning System, a network of satellites used for reckoning positioning and time.
GPSDO	GPS Disciplined Oscillator, a highly precise (and accurate) clock.
HSR	High speed rail
MIMO	Multiple input multiple output, a radio technique involving sending data along different paths than the main line of sight in order to increase capacity or to work around obstructions
PPS	Pulse-per-second signal, shared by GPSDOs acquired from the GPS system

PRBS	Pseudo-Random Binary Sequence. A binary sequence of numbers consisting strictly of 0 and 1
Processing Frame	A linear array that contains a pre-defined number of samples meant to be processed as a single unit.
SAR	Synthetic aperture radar, an imaging technique that uses moving antennas to construct a scene
Scatterer	An object that deflects a radio signal, causing multiple propagation paths
UPS	Uninterrupted Power Supply
USRP	Universal Software Radio Peripheral, a device with fast DACs and ADCs for radio signals

1.0 Introduction

Driven by Moore's Law, the size of transistors has shrunk to nanometer scale. The small size of the transistors posts significant challenges for semiconductor failure analysis as it becomes increasingly difficult to obtain clear images of the device.

The most common technique to acquire images of semiconductor is scanning electron microscopy (SEM). SEM scans the object being image and collects the low energy secondary electrons, which are electrons produced from the object itself by chain reaction due to the higher energy primary electron. Several problems are presented when using this techniques to acquire images. For example, at high magnification there are significantly more electrons produced at around the edge of object to to increased surface area, which results in the edges being brighter than the rest of the object. This "edge bloom" effect affects the measurement of critical dimensions as it is difficult for software to identify the edge during the post procesing step. On the other hand, the boundaries between materials are unclear sometimes. These problems call for a better method for measurement.

In recent years, machine learning has proven to be a powerful end–to–end solution in many area of data science, including natural language processing and image analysis. In particular, convolutional neural network (CNN) is a useful tool in image analysis. In order to utilize this tool, a large number of labelled data are required 'train' a neural network, which would perform a specific task based on the designed of the net.

A slightly worse model would out perform a more "advanced" model given the proper data [4]. However, acquiring the large number of data required for training neural network is a daunting task. Most semiconductor images are confidential properties, and annotating the images for training purpose requires tedious effects and is proned to human error. It would be convenient to be able to reliably syntehsize the images.

JMONSEL [11], developed by John Villarrubia and Nicholas Ritchie of NIST, is a single–threaded electron tracker written in Java. It simulates the physical interactions between electron and the surrounding materials as it traverses through the material. Using the simulation results from JMONSEL, it is possible to construct realistic electron microscope images. However, the program runs very slowly, partly due to the fact that it is written in Java. The program needs to be optmized in order to generate enough images in a feasible length of time for training neural network.

This report explains the concept, construction, verification and working details of a multi–threaded, CUDA–compliant C++ version of JMONSEL, CudaMONSEL, that is able to rapidly generate realistic SEM images. Initial tests shows that CUDAMONSEL is up to 5 times faster than JMONSEL when running on a single–thread, and the number of threads are only limited by the number hardware of the device running the simulator.

2.0 Concepts

2.1 Interaction Cross Section

When two particles interact, their mutual cross section is the area transverse to their relative motion within which they must meet in order to scatter from each other. Interaction cross section measures the interaction rates between particles. One can imagine a beam of particles of type a with flux ϕ_a crossing a region of space where there are n_b particles per unit volume of particle b. The interaction rate per target particle r_b will be proportional to the incident particle flux so that $r_b = \sigma\phi_a$. The proportionality constant σ, which has the dimension of area and therefore known as interaction cross section, contains the physics of the interaction.

To calculate the probability of interaction δP when particle a travelling in v_a passes through a volume with cross sectional area A filled with particle b traveling in v_b in time δt, we use

$$\begin{aligned} \delta P &= \frac{\delta N \sigma}{A} \\ &= \frac{n_b(v_a + v_b)A\sigma\delta t}{A} \\ &= n_b v \sigma \delta t \end{aligned}$$

Thus the rate r_a of interaction is given by $r_a = dP/dt = n_b v\sigma$. For a beam of particle of type a with density n_a confined to volume V, total rate is given by $r_a n_a V = \phi N_b \sigma$ (flux ϕ times number of target particles n_a times cross section σ) and the cross section is given by number of interactions per time per target particle divided by incident flux.

In general, the cross section can be expressed as the ratio between the "number of interest" per unit time per target particle and the incidence flux. In this case, the number of interest is the number of interactions.

2.1.1 Differential Cross Section

When the cross section σ depends on some final–state variable Q, such as angle of departure and energy at departure, it is known as differential cross section $d\sigma/dQ$. The quantity Q is usually sensitive to the underlying physics, meaning it changes drastically based on the type of interaction between the source particle and the target particle. Two of the most common 'Q's are final energy and angle of departure of the source particle.

Differential and total scattering cross sections are among the most important measurable quantities in nuclear, atomic, and particle physics [12].

2.2 Scattering Angle

3.0 Implementation

The goal of this project is to refactor the JMONSEL program in order to generate a large number of realistic SEM images in a reasonable length of time. Genereated images should contain landmark features common among SEM images, such as edge bloom, and features or defects resulting from microscope characteristics (eg. out of focus). To achieve this, it is necessary to simulate the trajectory of an electron in a computational efficient manner. This section contains the background and algorithm used by the simulation. The JMONSEL program is being developed jointly by John S. Villarrubia and Nicholas W. M. Ritchie of NIST.

The simulation is divided into two main packages, microanalysis and nanoanalysis. The microanalysis package, authored by Nicholas Ritchie, deals primarily with x–ray analysis, while the nanoanalysis package, authored by John Villarrubia, focuses on interactions between electron and the materials in the scene. While their applications are different, the nanoanalysis package expanded on the microanalysis package to allow secondary electron generation. We introduce both packages below, focusing on the connections between the two.

3.1 Material

Different scattering models require different material properties. Material is defined by its elemental composition and density. The exact composition of the material is used primarily to obtain

1. MeanIonizationPotential in JoyLuoNieminenCSD
2. RandomizedScatterFactory (NISTMottRS Factory for example) in SelectableElasticSM etc.

3.1.1 Secondary Eleectron Generating Material

Secondary Electron generating material (SEmaterial) is an extension of material. User is able to specify more material properties required by different scattering models to allow secondary electron generation.

To define an SE material, one may specify any number of the parameters below based on the model selected for the material

1. composition (what elements and in what proportions)

2. density

3. work function (optional)

4. work function (optional)

5. energy of its conduction band minimum (optional)

6. band gap (optional)

7. core energy levels (optional)

For some models, scattering cross sections depend upon the kinetic energy of the target electrons and not simply upon its total energy. To facilitate these models, kinetic energies are associated with each of the states. These energies may be supplied by the user (e.g., based upon literature values or the user's own model for the material). If they are not supplied by the user, default values are supplied. For electrons in the conduction band (energy above the conduction band minimum) the default algorithm assigns a kinetic energy equal to the distance of the state above the band minimum. For electrons at lower levels the default algorithm assigns a kinetic energy equal to the distance of the state below the vacuum level. The user should override the default values if more accurate informations are available.

3.2 Scattering Physics

Simulation physics are provided by subclasses of *AlgorithmUser* and *AlgorithmClass* abstract classes.

3.2.1 Algorithm Classes

All physics classes are derived from the abstract base class, *AlgorithmUser*. *AlgorithmUser* class provides the abstract method *initializeDefaultStrategy*, which initializes the physics for the class.

If a user wishes to use a different algorithm to implement a specific type of algorithm the user can supply a global override Strategy. This is enabled by the *Strategy* class.

3.3 Simulation

The simulation computes the movement of the electron in steps. First, it acquires the position of the active electron. From the position info, it calculates the current active region and the associated material scattering model. To obtain the position of electron at the next step, the simulation acquires a random mean path length of the electron based on the scattering model and computes the electron's position at next time step. With the new position, the simulation obtains the next region, moves the electron and decreases its energy. Finally, it performs a check to determine if the active electron's trajectory is completed. If the trajectory is not completed, the simulation performs one of three actions based on the next region. At each time step, the energy loss and position of the electron is updated. When a new electron is spawned, the state of the original electron is stored on a stack until the new electron is destroyed when its energy falls below the tracking threshold of the scattering mechanism. Refer to function *takeStep* in MonteCarloSS.cu for details.

3.3.1 Barrier Scattering Mechanism

Barrier scattering occurs when an electron encounters a boundary between two materials and deflects. Differences in the potential energy (ie. work function) in the two materials may cause a change in the electron's energy or trajectory. Note that it is possible for the barrier scatter mechanism to produce a secondary electron.

The barrier scatter mechanism follows the abstract class shown in 3.1.

Listing 3.1: BarrierScatterMechanism.cuh

```
class BarrierScatterMechanism
{
public:
  virtual ElectronT* barrierScatter(ElectronT* pe, const RegionBaseT* nextRegion) const
      = 0;
};
```

The only barrier scatter mechanisms included in JMONSEL is the default *Exponential Quantum*

Mechanical Barrier Scatter Mechanism.

Exponential Quantum Mechanical Barrier Scatter Mechanism

Quantum mechanical scattering from a finite–width barrier with “exponential” shape, meaning the form of the potential energy barrier is assumed to be (3.1).

$$U(x) = \frac{\Delta U}{1 + e^{-2x/\lambda}} \tag{3.1}$$

This barrier function represents a smooth ‘s–shaped’ transition from $U(x) = 0$ well to the left of the barrier to $U(x) = \Delta U$ well to the right. λ is a measure of the width of the barrier. Roughly half ∼46% of the transition occurs over a distance equal to λ (from $x = -\lambda/2$ to $x = \lambda/2$), 90% over 3λ.

In the limit that lambda is “large”, this probability approaches the classical result. In the limit that lambda goes to 0, it approaches the transmission probability for sharp barriers encountered in elementary quantum mechanics texts. For most purposes the useful range of λ values is from $0m$ to $1E-9m$. Anything larger than this is essentially classical.

This is therefore a general implementation of barrier scattering that includes classical and sharp–barrier quantum mechanical scattering as special cases. Two constructors are provided. Both constructors accept a material as input (from which the barrier height is determined). One constructor additionally allows specification of λ. The other does not. The constructor without λ specification implements classical barrier scattering (the large lambda limit). This method of implementing classical barrier scattering is preferred to giving an explict but large value of λ because it uses a simpler limiting–case formula, and it avoids possible numerical issues associated with large arguments. The constructor with an explicit λ specification implements quantum mechanical scattering for a barrier of the specified width. Equation 3.1 is an instance of the sigmoid function.

A barrier scattering mechanism is associated with each region of our sample, but barrier transmission is a pair-wise phenomenon. That is, the barrier height and width at an interface between materials A and B depends in principle upon the properties of both materials. If the electron starts inside material A, then it is material A’s barrier scatter mechanism that governs that particular scattering event. The present scatter mechanism determines the barrier height by comparing the potential energies in the materials on each side of the interface. It uses the barrier width associated with material A. This scattering mechanism uses the property energyCBbottom, which therefore need to be properly defined for the materials on both sides of the interface.

This class computes barrier transmission using the formula provided by [5]. Note that this is the only barrier SM available in the JMONSEL package.

3.3.2 Scatter Mechanism

A scatter mechanism governs the interaction of a electron with the environment. Each material scatter model can be associated with multiple scatter mechanisms, each with its own scatter effects and probability of occuring (eg. scattering rate). Its effects are a function of the medium/material and the electron's current energy. Effects can include changing the (primary) electron's direction and energy, and even generating a secondary electron with its own energy and direction through some inelastic scattering events.

The prototype of the class is shown below

```
class ScatterMechanism
{
public:
   /**
    * Returns the reciprocal of the mean free path.
    *
    * @param pe - the primary electron
    * @return double Reciprocal of mfp in inverse meters
    */
   virtual double scatterRate(Electron pe) = 0;

   /**
    * Updates properties of the primary electron based on results of scattering and
        returns either a secondary electron or null.
    * @param pe -- the primary electron
    * @return Electron -- the generated secondary electron or null
    */
   virtual Electron scatter(Electron pe) = 0;

   /**
    * Sets the material within which the electron scatters. This method typically
        precomputes and caches combinations of material properties required by the
        scattering model.
    * @param mat
    */
   virtual void setMaterial(Material mat) = 0;
}
```

Scatter mechanisms included in JMONSEL are:

1. BrowningMottElasticSM
2. FittedInelSM
3. GanachaudMokraniPhononInelasticSM
4. GanachaudMokraniPolaronTrapSM
5. KoteraPlasmonInelasticSM
6. MollerInelasticSM
7. SelectableElasticSM
8. TabulatedInelasticSM

A model usually requires two scattering mechanism, one for elastic scattering and one for inelastic / SE–generating scattering. For elastic scattering, SelectableElasticSM with NISTMottRS is a good choice. (cite) For inelastic scattering, choose TabulatedInelasticSM pair with ZeroCS) when when scattering tables are available, and use FittedInelasticSM (pair with JoyLuoNieminenCSD) otherwise. For most models, the governing equations are hardcoded into the class itself.

FittedInelSM

FittedInelSM requires a parameter, E_{av}, that represents the average energy required to make a secondary electron. It is based on an old semi–empirical model, described most recently (and perhaps in most refined form) by [7].

TabulatedInelasticSM

For TabulatedInelasticSM, most of the scattering properties are not hardcoded into the object itself, but read from tables associated with each supported material. For example, the inverse mean free path, interpolating an IIMFP table associated with the material, the distribution of energy losses and angle changes by the primary electron are similarly determined by tables.

The tables are served as strings of full paths to the files containing the physical data. Each string is input to NUTableInterpolation object. The NUTableInterpolation object expects the text files to be in the following format: first entry in the file should be number of dimensions of domain n. The next entry is the number of entry in the first dimension n_1, then followed by n_1 entries representing values in the first dimension, followed by the number n_2 etc. The numbers form a n dimensional mesh that represents the domain of interpolation. After repeating the pattern for n times, the file should contain $\prod_{i=1}^{n} n_i$ numbers representing the values of the data points.

The tables are:

1. tableIIMFP, IIMFP 1–d table (inverse inelastic mean free path vs. primary electron energy E_O)
2. tableReducedDeltaE, reduced deltaE 2–d table ($\delta E/E_0$ vs E_0 and r, with r a random number)
3. tableTheta, the theta 3–d table (scattering angle of PE vs. E_0, $\delta E/(E_0 - E_{Fermi})$, r)
4. tableSEE0, the 2–d table of SE initial energy vs. δE and r

For 3 of the tables, the first input parameter is the kinetic energy of the primary electron. By default, this kinetic energy is assumed to be the PE's energy with respect to the bottom of the conduction band. Sometimes, however, tables use a different convention. For example, in a semiconductor or insulator tables may be computed for energies measured with respect to the bottom of the valence band (the highest occupied band). This can be accommodated by providing the constructor with the energy offset between these two definitions of energy (ie., $E_{offset} =$ energy of conduction band bottom − the energy defined as the zero for purpose of the tables).

However, an exception to the above generalizations. There remains some uncertainty in the literature over the connection between primary electron energy loss/trajectory change on the one hand and secondary electron (SE) final energy/final trajectory on the other. Part of the reason for this difference lies in varying assumptions about the initial (pre-scattering) energy and trajectory of the SE. Some of the different methods of treating this connection have been implemented in this class. One of them must be selected to instantiate an object of this class. The selection is made by choosing a value for the methodSE parameter in the constructor. Three methods have been implemented to deal with this.

The tables are found under the directory "C:\Program Files\NIST\JMONSEL\ScatteringTables".

1. $methodSE = 1$ [1]: If the PE energy loss, δ E is greater than a core level binding energy, the SE final energy is $\delta E - E_{binding}$. Otherwise, it is $\delta E + E_{Fermi}$, where E_{Fermi} is the Fermienergy of the material. The final direction of the SE is determined from conservation of momentum with the assumption that the SE initial momentum was 0.

2. $methodSE = 2$ [2]: If δE is greater than a core level binding energy the treatment is the same as $methodSE = 1$. If not, the SE final energy is $\delta E + E'$. If E' were the Fermi energy this would be the same as $methodSE = 1$. However, E' lies in the range $max(0, E_{Fermi} - \delta E) <= E' <= E_{Fermi}$. The value of E' is determined probabilistically based upon the free electron densities of occupied and unoccupied states.

3. $methodSE = 3$ [8]: The scattering event is assigned as either an electron–electron event or an electron–plasmon event based upon the momentum transfer. Plasmon events are treated as in $methodSE = 2$, except that the SE direction is isotropic. Electron–electron events have energy and direction determined as described by [8].

To use this class, the user–provided tables must cover the full range of energies that the electron will encounter in the simulation. This generally means the tables that take PE energy as one of the inputs should cover energies from the material's Fermi energy up to at least the energy of electrons generated by the electron gun.

This class uses coreEnergy array, energyCBbottom, bandgap, and workfunction.

3.3.3 Randomized Scatter

The type of randomized scatter depends on the the energy of the electron. The main computations are performed by these two member functions below

```
__host__ __device__ virtual double randomScatteringAngle(const double energy) const = 0;
__host__ __device__ virtual double totalCrossSection(const double energy) const = 0;
```

The method *totalCrossSection* computes the total cross section for an electron of the specified energy.

The method *randomScatteringAngle* returns a randomized scattering angle in the range [0,PI] that comes from the distribution of scattering angles for an electron of specified energy on an atom of the element represented by the instance of the class.

There is also a method that tracks the type of element the class is applicable to. Below we introduce each type of randomized scatter in the simulation.

CzyzewskiMottScatteringAngle

CzyzewskiMottScatteringAngle is an extension of CzyzewskiMottCrossSection, which reads its scattering information from a table with domain spanning from $20eV$ to $30keV$. This class linearly interpolates the intermediate values. For energy below $20eV$, it uses the Browning model. For energy above $30keV$, it uses the Rutherford model.

BrowningEmpiricalCrossSection

Browning empirical cross section is applicable to low energy electrons (although not lower than $100eV$).

$$CS(Z,e) = \frac{3 \times 10^{-22} Z^{1.7}}{e + 0.005 Z^{1.7}\sqrt{e} + 0.0007 Z^2/\sqrt{e}}$$

where e is electron energy in keV, Z is the proton number of the element.

$$SA(Z,e) = \begin{cases} acos\left(1.0 - \frac{2.0\alpha r_2}{\alpha - r_2 + 1}\right), & \text{if } r_1 \leq \frac{r}{r+1} \\ acos(1 - 2.0 r_2), & \text{otherwise} \end{cases}$$

where $\alpha = 7.0 \times 10^{-3}/e$, r_1, r_2 are random numbers $\in [0,1]$, and $r = (300.0e/Z) + (Z^3/3.0 \times 10^5 e)$.

ScreenedRutherfordScatteringAngle

Screened Rutherford cross section is applicable to high energy ($\geq 20keV$) electrons.

$$CS(Z,e) = \frac{7.670843088080456 \times 10^{-38} zp(1.0 + z)}{e + 5.44967975966321 \times 10^{-19} zp^2}$$

where e is the electron energy in keV, Z is the proton number, and $zp = Z^{1.0/3.0}$.

$$SA(Z,e) = acos\frac{1 - 2.0 * alpha * r}{1 + alpha - r}$$

where $\alpha = 5.44968 \times 10^{-19} Z^{2.0/3.0}/e$, $r$is a random numbrs $\in [0,1]$.

NISTMottScatteringAngle

NISTMott

NISTMott randomized scatter tries to provide a comprehensive model for all energy ranges. It is a function of electron energy and material/element the electron is colliding with. For energy between $100eV$ and $20keV$, it calculates using the the NIST SRD 64 method by interpolating from a table of cross sections with Lagrange interpolation. For energies lower than $100eV$, it uses one of three methods.

1. Cutoff energy is $50eV$, interpolation Browning,

2. Cutoff energy is $100eV$, interpolation Browning,

3. Cutoff energy is $100eV$, linear interpolation

In all three methods, the cross section is zero when the energy is zero, and the cross section matches the tabulated value when energy is at some minimum cutoff value, which may or may not be the minimum tabulated value at $50eV$. Findally, for energies above the tabulated range, computations use Screened Rutherford Scattering Angle.

NISTMottScatteringAngle uses 2nd order Lagrange interpolation in $log(E)$ for the totalCrossSection calculation. For randomScatteringAngle it uses 0^{th} order interpolation (i.e., it chooses the nearest tabulated value) in $log(E)$ and 1st order in the random number. In contrast the present class uses 3rd order for all interpolations. Orders can be set by the constants *qINTERPOLATIONORDER* and *sigmaINTERPOLATIONORDER*. The interpolation tables are given by

1. *mSpwem* is a 1–d table of total cross sections, uniform in $log(E)$ from $log(50.)$ to $log(20000)$
2. $mX1[j][i]$ is a 2–d table of q vs $Log(E)$ and random #. j indexes $log(E)$. i indexes r. q is related to $cos(\theta)$ by $1 - 2q^2 = cos(theta)$

$$SA(Z,e) = \begin{cases} acos\left(1.0 - \frac{2.0\alpha r_2}{\alpha - r_2 + 1}\right), & \text{if } r_1 \leq \frac{r}{r+1} \\ acos(1 - 2.0r_2), & \text{otherwise} \end{cases}$$

where $\alpha = 7.0 \times 10^{-3}/e$, $r_1, r_2$are random numbrs $\in [0,1]$, and $r = (300.0e/Z) + (Z^3/3.0 \times 10^5 e)$.

3.3.4 Slowing Down Algorithm

Slowing down algorithms compute energy loss of the electron as a function of initial energy and distance moved. The loss depends on properties of the medium through which the electron travels.

```
public interface SlowingDownAlg {
   /**
    * Sets the material for which the energy loss is to be computed
    *
    * @param mat
    */
   void setMaterial(SEmaterial mat);
```

```
  /**
   * compute - Computes the energy change for an electron with initial energy
   * eK traversing distance d. The return value is negative if the electron
   * loses energy.
   *
   * @param d double -- the distance moved by the electron
   * @param pe Electron, the primary electron
   * @return double -- the energy change
   */
  double compute(double d, Electron pe);
}
```

Joy–Lou–Nieminen

Implements a continuous energy loss formula modified for low energies. Above a cutoff energy it uses the Bethe energy loss as modified by Joy and Luo. The Joy/Luo modification improves the accuracy at low energies. Below the cutoff energy it approximates the energy loss as proportional to the electron's energy to the 5/2 power, as described by [9]. The dividing line between these two approximations is a parameter that must be supplied to the constructor when this slowing down algorithm is instantiated. The proportionality constant in Nieminen's low energy form is then chosen to enforce equality of the two forms at the cutoff energy.

This scattering mechanism uses the following material properties, which therefore need to be properly defined: workfunction (unless supplied in the constructor), the elemental composition, density, and weight fractions.

The JoyLuoNieminenCSD stopping power requires a parameter, *breakE*, that specifies where the stopping power switches from the low energy Nieminen expression to the higher energy Joy/Luo one.

3.4 Events

The event listener 'hooks' onto the simulation and provides a way to transfer information out of the simulation loop. This means that event listeners are by design passive and unable to affect the outcome of the simulations. The simulation will work even when all event–related code are removed, in which case some other mean of acquiring simulation state needs to be implemented if one wishes to retrieve any information about the on–going simulation.

The following events are defined inside $MonteCarloSS$ class:

1. ScatterEvent
2. NonScatterEvent
3. BackscatterEvent
4. ExitMaterialEvent
5. TrajectoryStartEvent
6. TrajectoryEndEvent
7. LastTrajectoryEvent
8. FirstTrajectoryEvent
9. StartSecondaryEvent
10. EndSecondaryEvent
11. PostScatterEvent
12. BeamEnergyChanged

3.4.1 Material Scatter Model

A material scatter model consists of three parts:

1. A list of scattering mechanisms (e.g., Mott elastic scattering, Moller SE production, Plasmon SE production,...) that operate in the material. These scattering mechanisms may change the primary electron energy and direction and they may create secondary electrons.

2. A single barrier scattering function. This method models scattering at the interface between two materials.

3. A single continuous slowing down function. Determines the energy loss of the primary electron within the material.

Each material in the sample must be associated with a material scattering model, which determines the overall scattering behavior by calculating the mean free path, scattering angle and secondary electron generation.

3.5 Detector

The most important events of are events relating to detecting backscattered electrons. Detectors are devices that collect the information about electrons for later processing. In the simulation, electrons that are characterized as secondary electrons are backscattered electrons with energy below $50eV$. To simulate the detector, backscatter events are used. Backscatter events are fired only when the electron hits the chamber wall (one of the three normal end states of a electron). If required, we may also obtain forward–scattered secondary electrons for TEM image simulations.

Concretely, the event listener must be added to the simulation class (ie $MonteCarloSS$), before the start of the simulation, and have member function $actionPerformed$ of the listener classes added to the simulation deal with the appropriate events. Note that, the simulation broadcasts the event triggered to all listener classes by calling the member function $actionPerformed$ of every listener class added to the simulation in the function $fireEvent$.

3.6 Shape

3.6.1 NormalShape

3.7 Amphibian – custom C++ library

Since one of the goals of the project is to make MONSEL run on both CPU and GPU, we built a custom library, Amphibian, to eliminate any dependence on the C++ standard template library (STL). The following data structures are included,

1. linkedlist (single– and double–linked)
2. vector
3. stack
4. unordered set
5. unordered map
6. (mutable) string

Algorithms like quick sort, binary search are also included. There is also a random class responsible for generating random numbers reliably.

4.0 System Verification

The program uses a series of smoke tests to ensure that most systems are functioning properly. Smoke tests include tests for library, imported data and computed results. Test sets obey the topological order (ie proper functionality of one test set relies on that of the previous test set). Similarly, tests within each test set obeys the topological order.

First, *testLibrary* function runs library tests to ensure that container classes in Amphibian library work as they should. *initSim* loads the data required (eg caches and tables, excluding scattering tables) for simulation. *testSim* runs the tests implemented for JMONSEL classes included in the original JMONSEL code. Some of these tests require the data loaded from the previous step, *initSim*. *loadNUTable* loads the scattering tables required for TabulatedInelasticSM class. Finally, *initRange* defines the locations of the raster scan and therefore the resolution of the final image.

5.0 Working Details

The main simulation loop is defined in the *MonteCarloSST* class. To run the simulation, user has to provide the electron gun and the chamber as parameters to the simulation. The electron gun provides information of the initial electron where as the chamber contains the scene to be simulated. At any step of the simulation, only a single electron is being tracked as it traverses through the regions. In the case where secondary electron is spawned, the simulation saves the state of the primary electron on a stack and allow the secondary electron to take over until the secondary electron's trajectory is completed before it resumes simulating the primary electron.

As soon as the electron leaves the electron gun, the simulation determines the next position of the electron by calculating the mean free path of the electron given the current state of the electron inside the surrounding material. As the electron moves, it loses energy. Once the electron's energy falls under the tracking energy of the material, the electron's trajectory is ended. Otherwise, the electron can be found in one of the following situations.

1. The electron stays in the same region as the region before the step is taken
2. The electron crosses a barrier from one region to another

If either of the above applies, the electron's trajectory is ended. Note that in either situations, it is possible for a secondary electron to spawn. These are the only way a secondary electron can possibly be spawn. The probability of spawning a secondary electron is determined by the material scatter model of the material. For detail see *takeStep* method in *MonteCarloSS* class.

6.0 Analysis

6.1 Backscattered Electron Energy Distribution

In SEM imaging, we are interested in only the low energy, backscattered electrons. A large percentage of these electrons are secondary/Auger electrons spawned by the higher energy primary electron. As a result, It would be useful to study the energy distribution of the backscattered electrons to better defined the energy threshold for these secondary electrons. In this section, we examine the backscattered electron energy distribution used to produced the image on figure 6.1. Also note that since the exact electron generation may be beyond 'secondary', it is common for total electron yield to be more than 200%.

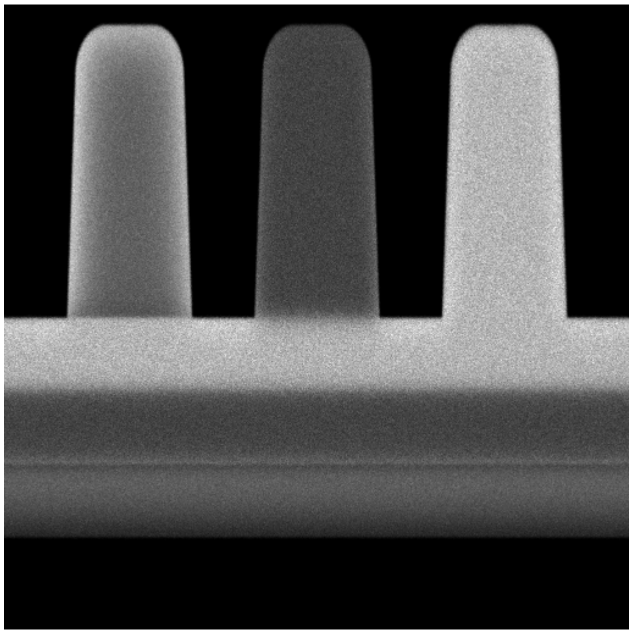


Figure 6.1: SEM image of a feature with 3 lines

The image on figure 6.1 is generated by with primary electron energy of $1000eV$, beam radius of $0.5nm$ and 200 electron per location using PMMA, Si and SiO2 as feature materials (left to right) while assume secondary electron energy being below $50eV$. Figure 6.2 below shows the energy distribution of backscattered electron at a particular, non–vacuum location on the image. Most of the backscattered electrons detected were low energy electrons, with a small number of of higher energy electrons detected beyond $50eV$. As a result, we conclude that it is safe to categorize electrons with energy lower than $50eV$ as secondary electrons for SEM imaging.

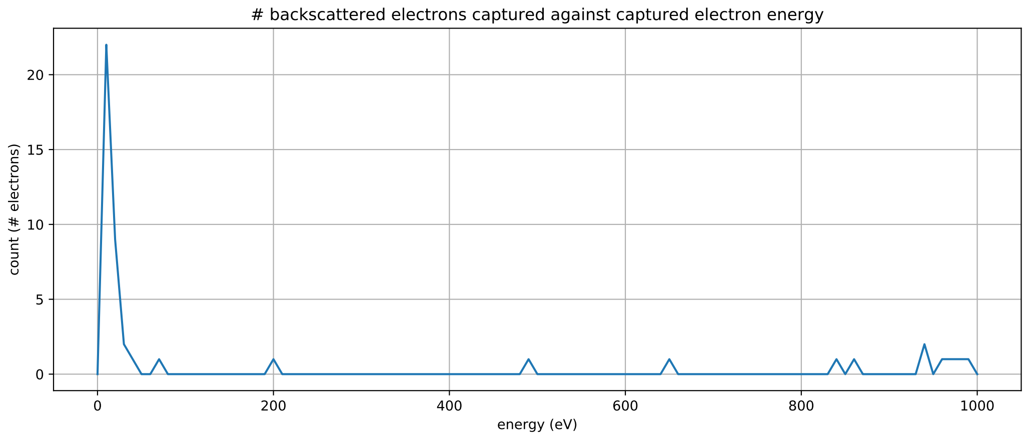


Figure 6.2: Backscattered electron energy distribution on a random, non–vaccum location of the image

To show that figure 6.2 is a representative energy distribution plot at any non–vacuum location, we produced a heat map of electron energies by acquiring a line scan of non–vacuum locations in figure 6.3. The top and bottom rows of figure 6.3 corresponds to row 16, column 75 and 132 (left and right edges of the feature) on the image 6.1. Since similar distributions are observed on different non–vacuum parts of the image, we will assume the distribution in 6.2 is a reasonable estimate of the energy distribution at any non–vacuum location.

6.2 Primary Electron Energy

In the previous section, we examined the energy distribution of detected electrons and discovered that most detected electrons are SEs with low energy $< 50eV$. It would be interesting to find out the effects of different primary electron energy on the SE energy distribution. In this section, we study how the primary electron energy affects the SE yield by comparing the relative quantity of SEs detected when similar electron beams are targetd on three different materials: SiO2, Si, and PMMA.

In this experiment, we used 250 primary electrons with beam size $.5nm$ and energy of $500eV$ at each location for the first image, and increased the PE energy by $500eV$ for each subsequent

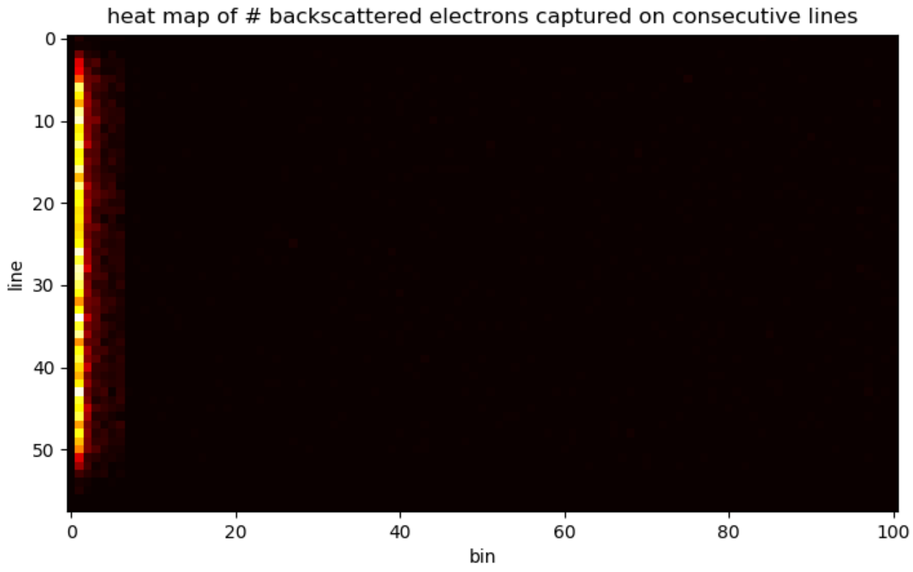


Figure 6.3: Heatmap of backscattered electron energy distribution on a random, non–vacuum line of the image

image. On every image, there are 3 line features made from PMMA, Si, and SiO2 (left to right), and 3 strips made from SiO2, Si, PMMA (top to down).

From the results 6.5, we can clearly see that brightness of all three materials decreased with higher energy. In particular, the relative brightness of Si decreased significantly relative to the other 2 materials, followed by PMMA while that of SiO2 remained approximately the same. The decrease in brightness in all 3 materials is consistent with the "universal law of SE yield", given in [6] by the equation 6.1 and figure 6.4

$$\frac{\delta}{\delta^m} = 1.28 \left(\frac{E_{PE}}{E_{PE}^m}\right)^{-0.67} \left(1 - exp\left(-1.614\left(\frac{E_{PE}}{E_{PE}^m}\right)^{1.67}\right)\right) \tag{6.1}$$

where δ^m is the maximum yield, which we assumed to be 1 and $E_{PE}^m \approx 1keV$ is the primary energy of the electron where the maximum is obtained. In particular for Si, the change in brightness shows that its SE yield reaches a maximum at a few hundred electron volts and decreases monotonically at approximately $1/E_{PE}$[6] as the energy increases [10][3].

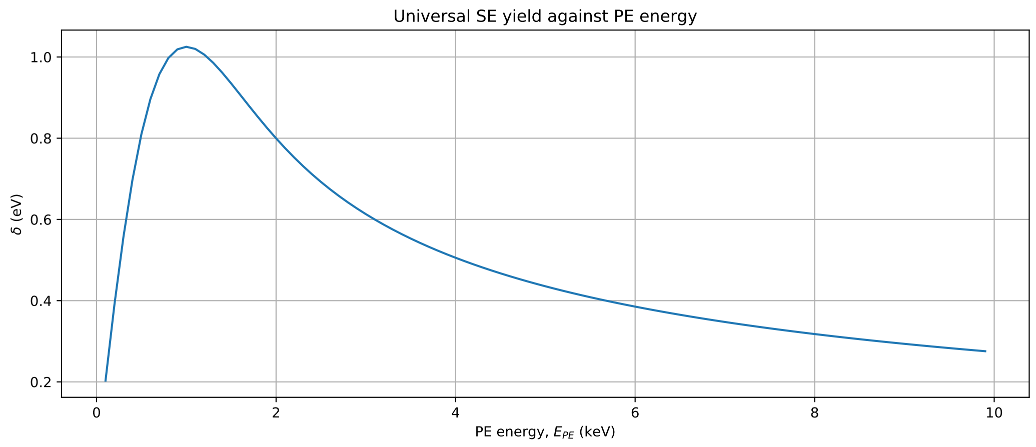
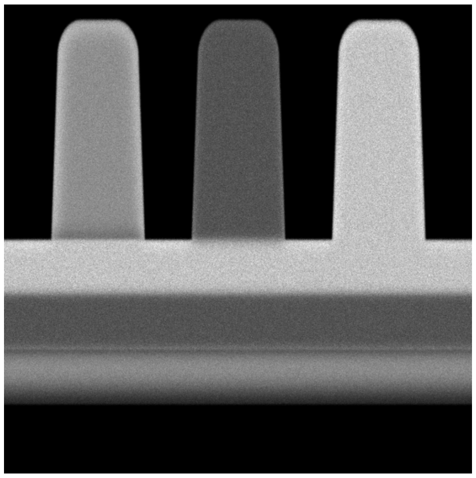


Figure 6.4: Universal law of SE yield against PE energy[6]

6.3 Gaussian Beam Radius

6.4 Number of Electron Trajectory per Location


(a) 500eV beam energy

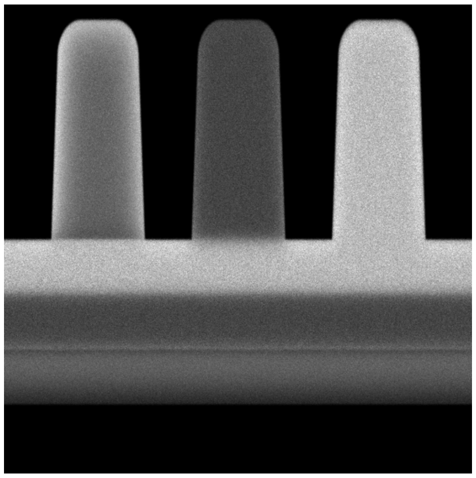

(b) 1000eV beam energy

Figure 6.5: 2 images with same setup using different beam energy

7.0 Conclusion

Overall, a working prototype of a spatial channel sounding device consisting of USRPs, mini-computers, GPS receivers, and atomic clocks was constructed. It used bi-static SAR algorithms to reconstruct the locations of scatterers in a field test. It has a spatial resolution of 12 m, and costs much less than current channel sounding solutions (around $12000 in total, see Table 8.1, versus the $100,000+ devices currently used). It is not without operational issues and difficulties however, so future work should continue on this project to make it commercially viable.

8.0 Deliverables

The main deliverable is a working spatial channel sounding prototype, already in the Radio Science Lab. It consists of the following components.

8.1 Hardware

8.1.1 Transmitter

- Mini-PC
- USRP
- Rubidium Frequency Standard
- GPS Receiver
- UPS
- Antenna
- Antenna Mast
- Amplifier

8.1.2 Receiver

- Mini-PC
- USRP
- Rubidium Frequency Standard
- GPS Receiver
- UPS
- Antenna
- Antenna Mount for Dr. Michelson's car

8.2 Software

8.2.1 Transmitter

- `transmitter_matlab.m`, Transmitter signal senerator (MATLAB)
- `transmitter.sln`, USRP transmitter software (C++)

8.2.2 Receiver

- `verifySignal.m`, for verifying signal validity (MATLAB)
- `receiver.sln`, USRP receiver (C++)

8.2.3 Post-Processing and Image Reconstruction

- `prepSignalForSim.m`, for range-compressing received signals (MATLAB)
- `jeppler_hsrsar_backproj.m`, Simulation and image reconstruction algorithm received from MDA.
- `backscatter.sln`, Fast image reconstruction algorithm re-written in C++ using DirectX and GPU.

8.3 Documentation

- GPU programming concepts and documentation of our reconstruction algorithm.
- Proposal
- Final Report

8.4 Estimated Raw Cost

Since all of these components were already available in the RSL at the time of project commencement, there are no current project costs. Table 8.1 shows the estimated total cost of all parts for both the transmitter and receiver.

Item	Cost
2x Lenovo M93p	2x $800
2x USRP N210	2x $1700
2x Rubidium Frequency Standard	2x $1500
2x UPS	2x $100
2x Timing GPS	2x $200
2x Panel Antenna	2x $200
30W Amplifier	$3000
DRGPS	$400
Total Cost	$12400

Table 8.1: Estimated total cost of prototype system

9.0 Recommendations

Another way would be to refactor the simulation enough so that they can be ran on GPU using CUDA. The advantage would be that we would be able to speed up the simulation significantly with our existing setup. The difficulty is that due to single core optimization implemented by some of the classes, it is required for every thread to have its own copy of the entire setup. In the CUDA environment where on–board memory comes at a premium, this would require some major code refactoring and testing before any development can take place, and it may not be as cost–efficient as getting a new CPU which may also provide significant speed improvement. Having multiple GPUs enabled also proved to destabilize the Windows 10 system on our current setup, causing the system to crash by DPC_WATCHDOG_VIOLATION. Further, the code is written in a completely CUDA–compliant manner, and a blueprint of the working principles is already in place. Barring any hardware upgrade or new insights in working details of CUDA API, the development of a working CUDA implementation of the simulator would be indefinitely postponed.

Appendices

A.0 Lagrange Interpolation

Given $n+1$ points (x_0, y_0), ... (x_n, y_n), the Lagrange Interpolation is given by

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

where $f(x_i) = y_i$ and $L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$.

References

[1] Z-J Ding and R Shimizu. A monte carlo modeling of electron interaction with solids including cascade secondary electron production. *Scanning: The Journal of Scanning Microscopies*, 18(2):92–113, 1996.

[2] ZJ Ding, XD Tang, and R Shimizu. Monte carlo study of secondary electron emission. *Journal of Applied Physics*, 89(1):718–726, 2001.

[3] C.G.H. Walker et al. The secondary electron emission yield for 24 solid elements excitedby primary electrons in the range 250–5000 ev: A theory/experimentcomparison. *Image Processing, IEEE Transactions on*, 30:365–380, 2008.

[4] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. 2009.

[5] Landau and Lifshitz. *Quantum Mechanics.* Prentice Hall, 1958.

[6] Yinghong Lin and David C. Joy. A new examination of secondary electron yield data. *Surface and Interface Analysis*, 37:895–900, 2005.

[7] Yinghong Lin and David C Joy. A new examination of secondary electron yield data. *Surface and Interface Analysis: An International Journal devoted to the development and application of techniques for the analysis of surfaces, interfaces and thin films*, 37(11):895–900, 2005.

[8] SF Mao, YG Li, RG Zeng, and ZJ Ding. Electron inelastic scattering and secondary electron emission calculated without the single pole approximation. *Journal of Applied Physics*, 104(11):114907, 2008.

[9] Nieminen. *Scanning Microsc. 2.* •, 1988.

[10] Francesc Salvat-Pujol and John S. Villarrubia. Conventional vs model-based measurement of patterned line widths from scanning electron microscope profiles. *Ultramicroscopy*, 206, 2019.

[11] John S Villarrubia, AE Vladár, Bin Ming, Regis J Kline, Daniel F Sunday, JS Chawla, and Scott List. Scanning electron microscope measurement of width and shape of 10 nm patterned lines using a jmonsel-modeled library. *Ultramicroscopy*, 154:15–28, 2015.

[12] Wikipedia. Cross Section. Retrived from https://en.wikipedia.org/wiki/Cross_section_(physics)#Differential_cross_section, 2019.