Electromagnetic Source Detection and Localization in a Room with Uncertain Dimensions
ECE 285 Spring 2012

Chris Tralie

April 22, 2012

1 Executive Summary

The purpose of this project was to assess the feasibility of a computational electromagnetic technique for determining the transfer function between a source and a receiver in a room environment. The idea of the technique is to attempt to estimate the scattering off the walls and objects of a room based on noisy 3D point cloud measurements taken of the room. Uncertainty in the position of walls and obstacles arises from the 3D measurement noise and can lead to false estimates of the EM field, especially if that uncertainty leads to constructive and destructive interference at the wrong places. Some believe that this makes the problem of estimating EM fields without perfect knowledge of an environment a false-starter, so it is important to do a rigorous experiment to verify this claim.

To do a rigorous analysis of the effects of uncertainty on this problem, thousands of experiments need to be run in a room on the order of 100 cubic meters. This means that a full wave proprietary EM solver is out of the question. Instead, a much more approximate simulator is implemented that only models specular reflection paths from source to receiver. It uses the well known geometric image sources and ray tracing technique to model perfect specular paths, and is able to approximately 22,000 experiments in 6 hours. The source code for this simulator can be found in the public git repository at the link https://github.com/ctralie/G-RFLCT. Although the simulator is approximate and the experiment is in a highly simplified environment, the hope is that the basic flavor of the problem is still captured and that some conclusions about the feasibility of this approach can still be drawn.

The official experiment models scattering from source to a receiver in a room of dimensions 5mx2.5mx5m, with uncertainty independently normally distributed with mean zero and standard deviation 3cm: a resonable error model for the PrimeSense Kinect 3D sensor. Performance is compared between a detector that models the measurement error and one that neglects it. An attempt is also made to determine which one of nine possible source locations an EM signal came from. Two main trends are observed. First of all, the detection performance penalty for neglecting measurement error is significantly higher for signals with wavelengths on the order of the measurement error. Even so, accounting properly for the measurement error does not degrade performance much beyond what it would be with perfect measurement, even if the error is on the order of a wavelength. This means that this technique may be feasible over a wide range of wavelengths if the prior over the measurement error can be estimated very accurately.

2 Introduction

The purpose of this project is to perform a feasibility analysis of a Ph.D. research project that is currently underway in my lab. We would like to estimate multipath parameters of a room from geometry inferred from a 3D optical scanner (like the PrimeSense Kinect). The hope is that scanning an environment with a 3D scanner can give enough information about boundary conditions in a room, such as obstacles and wall positions, to simulate electromagnetic propagation at any arbitrary wavelength in that room. This information can be used to come up with better statistical models for multipath in room, which can be used to improve source localization, determine potential dead spots, and steer propagation in a particular

direction, among many more applications. Figure 1 shows a cartoon picture of this process. The scanned geometry, including the walls and obstacles, is shown in black. A source is placed at the red circle, and a receiver is placed at the blue circle. We would like to get the transfer function between the source and receiver, a subset of which is shown by the purple reflection paths in that picture.

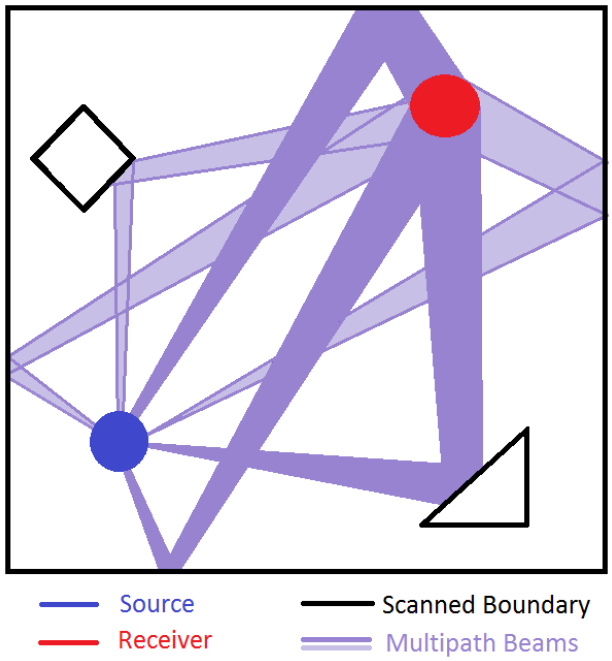


Figure 1: A multipath propagation scenario from scanned geometry

If we knew all of the geometry of an environment exactly with its reflection coefficients, then a simulation of propagation between the source and receiver should be a perfect match to reality, assuming fine enough meshing of the simulator. Unfortunately, a lot of uncertainty arises from imperfections in the 3D optical scanner. Figure 2 shows an example of geometry extracted from a kinect scan of part of our lab. Noise is clearly visible in this mesh. At typical ranges, it has been found that the location of 3D points returned by the Kinect can have an error of several centimeters. This error may be small compared to the dimensions of the room, and it may be perfectly acceptable for applications of the Kinect involving visualization, machine learning, and classification. However, for this particular application of performing an EM wave simulation, these small errors can have drastic effects because of the interference patterns that can occur when two wavefronts meet. If the error of the geometry is off by a few centimeters and this happens to be on the order of a wavelength (which will happen for Ghz frequencies), then the simulator may mistake a constructive interference pattern for a destructive one.

Given the knowledge of the problem, the purpose of this project will be to model uncertainty in geometry placement in a room and to analyze the effect on the transfer function between source an receiver. A basic attempt at EM source localization will be attempted under these different scenarios, and comparisons will be made at two different wavelengths.

3 Simulation Environment

Ideally a full wave electromagnetic wave solver would be used to model the transfer function once the geometry was imported from the Kinect. I attempted to use the proprietary solver known as "CST Microwave Studio" to simulate multipath from a source to a receiver placed in a room. However, I found that for a room of typical dimensions at wavelengths of a around 1Ghz, a single simulation would take days to complete because of the number of volumetric elements needed in the mesh. Since I needed to run tens of thousands of simulations for this project with different uncertainties and different source positions, this was not an option.

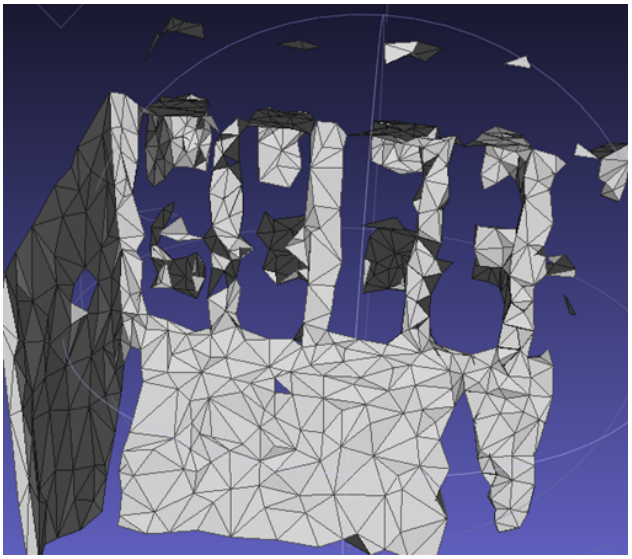


Figure 2: An example of a 3D surface inferred from Kinect scans of a part of my lab office

Instead, I decided to write my own simulator that was much more approximate. Instead of capturing the full complexity of wave interaction in a medium, my simulator simply attempts to enumerate all specular reflection paths (where angle in = angle out) from a source to a receiver placed in some surfaces. This means that it neglects more complicated wave effects such as diffraction and transmission, but the hope is that the important reflection paths will still be enough to inform us about the feasibility of this design.

3.1 The Image Sources Method

To find all of the perfect specular reflection paths, where the angle of incidence of a wavefront at a boundary is equal to the angle of reflection, I used the well known virtual image sources method. The basic idea is to reflect the original source across every planar face in a scene to create "virtual sources." These virtual sources can themselves be treated as electromagnetic sources in addition to the original source, and a path from them to any receiver in the scene can be calculated and treated as an actual propagation path. Figure 3 shows an example of a simple scene with four planar faces and the four virtual images associated with those faces.

It can be shown that drawing paths in this manner will always give rise to paths where the incident and reflected angles are equal. The code that I wrote to calculate the reflections is shown below:

```
def buildVirtualSourceTreeRecurse(self, currNode, level, maxLevel):
  self.vSources.append(currNode)
  if level == maxLevel: #The maximum number of bounces allowed has been reached
    return
  #Try to mirror this source around every face in the scene
  for m in self.meshes:
    EMNode = m.EMNode
    for f in m.faces:
      if f != currNode.meshFace:
        fP0 = currNode.pos
        facePoints = [P.pos for P in f.getVertices()]
        if level > 0: #Do pruning test to avoid making
          #virtual sources that cannot be reached from this one
          #meshFaceInFrustum(fP0, frustPoints, facePoints)
          frustPoints = [P.pos for P in currNode.meshFace.getVertices()]
          if not meshFaceInFrustum(fP0, frustPoints, facePoints):
```

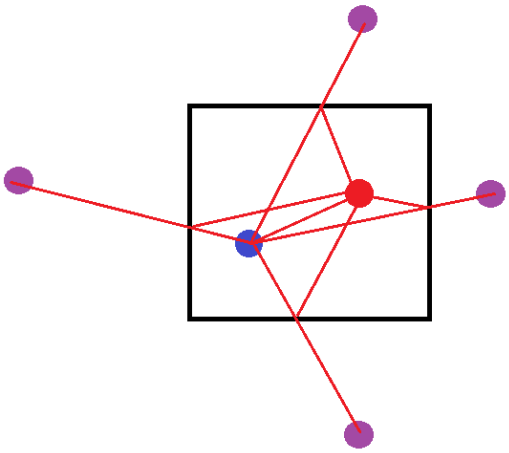


Figure 3: An example scene with all of the first order virtual images created. The original source is shown in red, the receiver is shown in blue, and the virtual images are shown in purple

```
        continue
    #Tests passed: need to make a virtual image here
    #First mirror the current virtual source point
    #across the plane of the new face
    faceNormal = getFaceNormal(facePoints)
    dV = fP0 - facePoints[0]
    perpFaceV = faceNormal.proj(dV)
    parFaceV = faceNormal.projPerp(dV)
    mirrorP0 = facePoints[0] + parFaceV - perpFaceV
    newSourceNode = EMVSourceNode(mirrorP0, currNode, f, m.EMNode)
    currNode.children.append(newSourceNode)
    #Now recursively make the virtual images for this new image
    self.buildVirtualSourceTreeRecurse(newSourceNode, level+1, maxLevel)
```

The code is recursive, because virtual images can be created from virtual images. A virtual image of a virtual image models paths that result from two bounces, and the virtual image of that virtual image models 3 bounces, etc. One should note, however, that the number of virtual images grows exponentially in the number of bounces modeled, with the base of the exponent equal to the number of faces in the scene. A section of my code attempts to prune the tree by eliminating mirrors across faces that a source's radiation could never reach. This is absolutely essential for scenes with high polygon counts, to reduce the effects of exponential blowup. The technique is similar to hidden surface removal in computer graphics, and it is shown pictorally in 4

Once the positions of the virtual sources have been calculated and a receiver position has been determined, an actual path can be extracted from source to receiver. Figure 5 shows an example of extracting a path from an image that has resulted from three recursive reflections across surfaces. First, a ray is cast from the receiver to the virtual source position (this is where actual ray tracing math occurs in the program). If nothing is in the way (it intersects the correct face corresponding to that virtual source), store that intersection point away, and call it P_0. Now cast a ray from P_0 to the virtual source's parent. If nothing is in the way, store that intersection point away. Repeat the process recursively until the path reaches the source. If any intersection is encountered along the way, the path is invalid and is discarded.

Luckily, in my relatively simple scenes of just boxes, the geometry is simple and the number of obstacles in the way of virtual sources is relatively low, so the ray tracing can take place quickly.

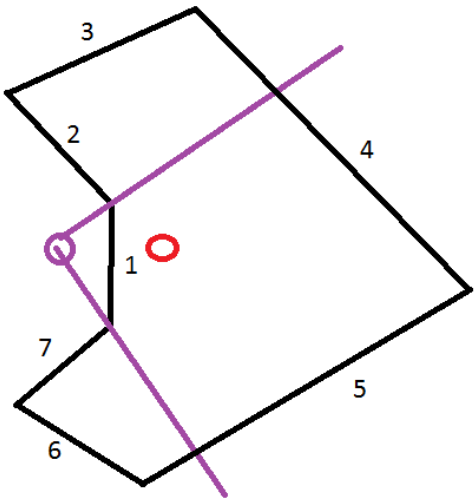


Figure 4: An example of a scenario where the image sources tree can be pruned. The virtual source under consideration (purple), which is associated with face 1, should only be mirrored across faces 4 and 5, because no path from that source could ever reach sources 2, 3, 6, or 7

3.2 Capabilities of My Simulator

I wrote am image sources simulator from scratch in Python and it is about 2000 lines of code, roughly 600 of which are devoted to the EM propagation (the rest is used for visualization and geometric primitive operations). The simulator can generate all virtual sources up to a specified order for a given "scene specification." A scene consists of a source position, a receiver, and 3D geometry which is stored along with reflection coefficients for that geometry. The simulator currently supports geometry that consists of convex planar patches, and it can handle any arbitrary affine transformation of geometry. Scenes and transformations are specified in an XML file, which is parsed by the simulator. An example file is shown below:

```
<EMScene>
  <EMMaterials>
    <material name = "reflective" R = "0.9" T = "0.0"/>
    <material name = "absorptive" R = "0.0" T = "0.0"/>
  </EMMaterials>
  <Source pos = "0 0 -2"/>
  <Receiver pos = "0 0 2"/>
  <box length = "5" width = "5" height = "2.5" center = "0 0 0" em = "reflective">
    1 1 0 0
    1 0 0 0
    0 0 1 0
    0 0 0 1
  </box>
</EMScene>
```

This file shows an example of specifying source and receiver positions in the world, as well as defining a simple scene with one box (six planar faces). This file also shows how to specify some rudimentary electromagnetic properties of surfaces (reflection and transmission coefficients), and it shows a specification of an affine transformation of geometry by placing a 4x4 matrix in a box tag (the matrix in this example is a shear matrix, but any affine transformation is supported). Once a scene is specified and the virtual sources are generated, the simulator is able to enumerate all specular reflection paths up to a certain order. Figure 6 shows a screenshot from a visualization program I wrote for my simulator.

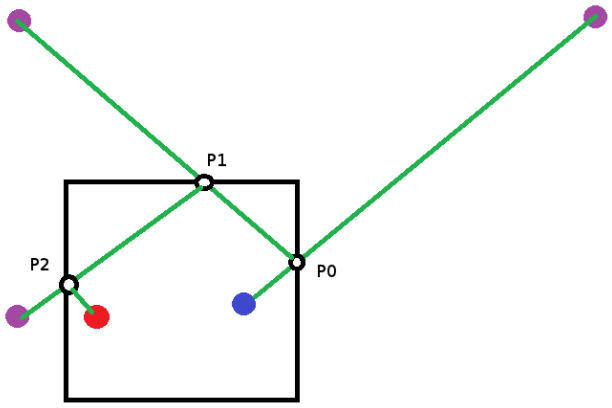


Figure 5: An example of tracing a path from a virtual source at depth 3 in the virtual sources tree (the result of 3 mirrorings), which gives rise to three bounces

3.3 Generating an Impulse Response

Once all reflection paths up to a certain order are found from source to receiver, this information needs to somehow be converted into an impulse response. Each path is accompanied by some attenuation, due both to path loss and reflections, and also by a phase delay that is determined by the length of the path and the speed of light. I modeled path loss with the well-known Friis Free Space Path Loss Model for antennas between a source and receiver:

$$PL = (\frac{\lambda}{4\pi d})^2 = (\frac{c}{4\pi f d})^2$$

where d is the distance from source to receiver, c is the speed of light, and f is the frequency in hz of the transmitted wave. Specifying the frequency in Mhz as f_{Mhz}, this can be rewritten as

$$PL = (\frac{300}{4\pi f_{\text{Mhz}} d})^2$$

The delay in seconds, τ, is simply d/c. Figure 7 shows an impulse response that is generated from paths using these equations in a simple room. The direct path from source to receiver (no bounces) is visible as the first impulse in time, and later impulses correspond to bounces from reflection off of boundaries. The attenuation of later bounces naturally arises from path loss and bouncing off of walls with reflection coefficients less than 1.

Once the impulse response is calculated, a steady state sinusoid with a particular amplitude and phase can be calculated as the sum of attenuated and phase delayed sinusoids from each path. The following equation does this by taking the following sum over each of the i paths:

$$s(t) = \sum_i A_i cos(2\pi f(t - \tau_i))$$

where A_i is the attenuation along the i^{th} path due to path loss and reflection losses, and τ_i is the phase delay due to propagation length. The rest of the project deals with the detection and classification of signals in this form: the steady-state sinusoids resulting from an impulse response from source to receiver

4 Experimental Setup

4.1 Overview

For the purposes of this course I considered a simple rectangular room with dimensions measured as [5mx2.5mx5m] (ceiling is 2.5 meters and the room cross sections are square). I placed the receiver at

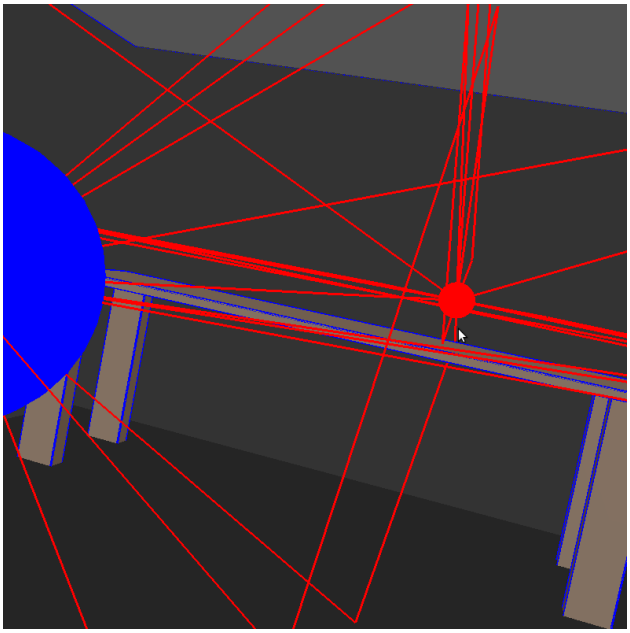


Figure 6: A screenshot from my simulator which was run on a simple scene consisting of a rectangular room with a desk in it. Source is shown as a red sphere and receiver is shown as a blue sphere. Red lines show specular paths from source to receiver.

the location (0.3, -1.2, 2.0), and I placed 9 sources in a 3x3 grid in front of the receiver (at the other side of the room in z). The sources are at positions ([-1, 0, 1], [-1, 0, 1], -2), where the square brackets indicate the possible values of the x and y position on the 3x3 grid. I fixed the reflection coefficients of each wall at 0.9. Note that the receiver's XY coordinates were intentionally chosen to be slightly offset from the centroid of the sources so that there would be no symmetry from two different sources (and so that no two received signals would be the same from two different sources)

4.2 Modeling Measurement Error

I also allowed for some measurement error in the dimensions of the walls, where the error in each dimension (x, y, z) is assumed to be independent Gaussian distributed with mean zero and standard deviation 3cm. This choice was motivated by experiences I have had with the Kinect fitting points to spheres and planes. For the purposes of this experiment, I actually discretized the possible error values uniformly across the interval from (-5cm to 5cm), for a total of 11 possible errors in every dimension. Note that the source positions are assumed to be uniformly distributed (every source position is equally likely) and independent of the error in the dimensions. This means that with the error incorporated in this discretized way and with the presence of 9 possible sources, this is an 11x11x11x9 = 11979 composite signal hypothesis problem. I ran my simulator for each of these 11979 possibilties and stored away the steady state sinusoid at the receiver that is associated with each one of them. This information was used later to generate the ROC curves.

4.3 Operating Frequencies

I chose two different operating frequencies of the transmitted sine wave: 915Mhz and 15Ghz. The wavelength of a 915Mhz wave is 33cm, while the wavelength of the 15Ghz wave is 2cm. In other words, the room dimension errors modeled are on the order of $1/10^{th}$ of a wavelength for the 915Mhz sine, and on the order of a wavelength for the 15Ghz wave. For a direct path, this means that small errors matter much more for a 15Ghz signal than they do for a 915Mhz signal, so one would expect detection performance to be poorer for the 15Ghz signal in this experiment. As a cursory look into this phenomenon, Table 1 shows the result of error on received sinusoids for a particular source/receiver configuration with no noise added. A relatively

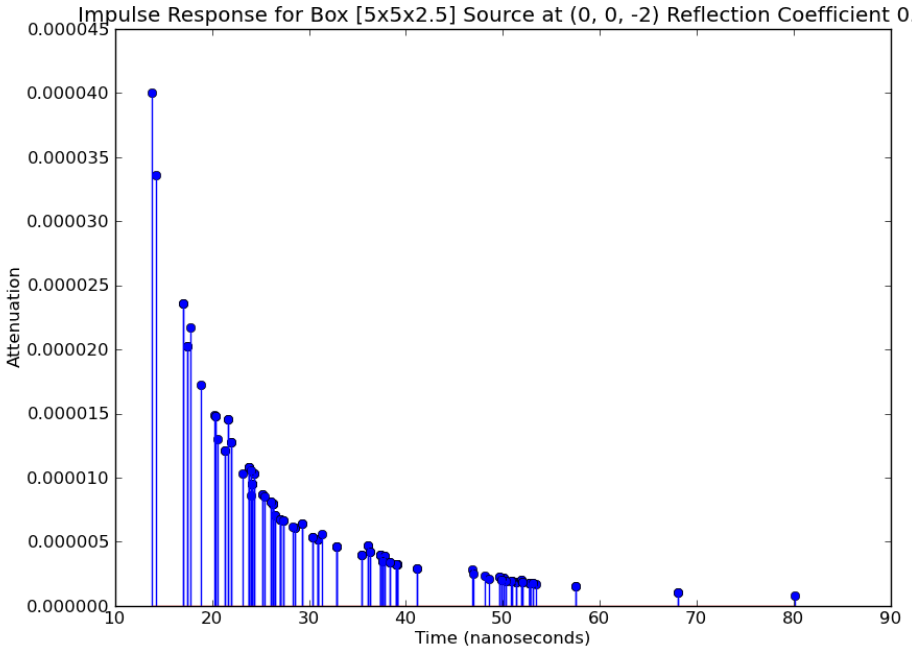


Figure 7: An example of an impulse response from source to receiver in a simple box room

small error of 2cm shifts the 915Mhz signal only slightly, while the same error shifts the 15Ghz sinusoid by half of a wavelength. The larger error of 10cm shifts the 915Mhz signal more, but has an extremely drastic change on the 15Ghz signal where some paths that were constructive actually began to destructively interfere and change the amplitude in addition to the phase. More rigorous experiments will be done to assess the holistic effects at both operating frequencies, but this at least leads to a hypothesis that a failure to model uncertainty properly will be worse for a higher frequency signal.

4.4 Generating ROC Curves

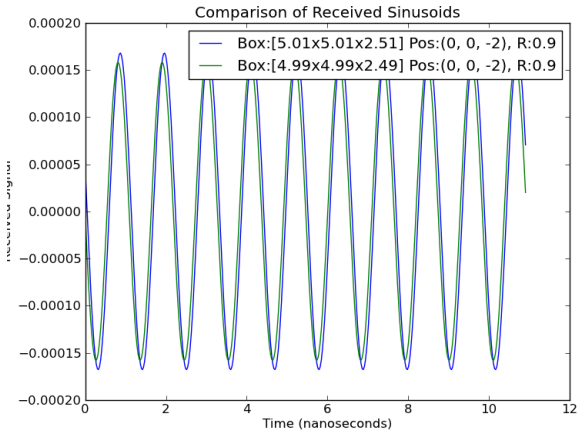
Once all of the 11979 possible signals are generated at the two operating frequencies, white Gaussian noise is added at the receiver and ROC curves are generated. There are three types of ROC curves considered. The first is from the scenario where the box is fixed at [5m x 2.5m x 5m] (i.e. perfect measurement) but the source positions can still be anywhere in the 3x3 grid. The optimal detector in this case is

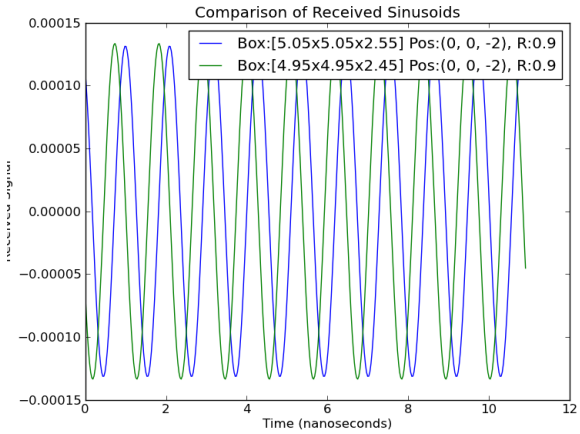
$$\lambda(\vec{R_k}) = \frac{1}{9}\sum_{i=1}^{9}\exp(\frac{E_s(i)}{2\sigma_n^2} + \frac{1}{\sigma_n^2}\sum_{j=1}^{k} r_j s_j(i))$$

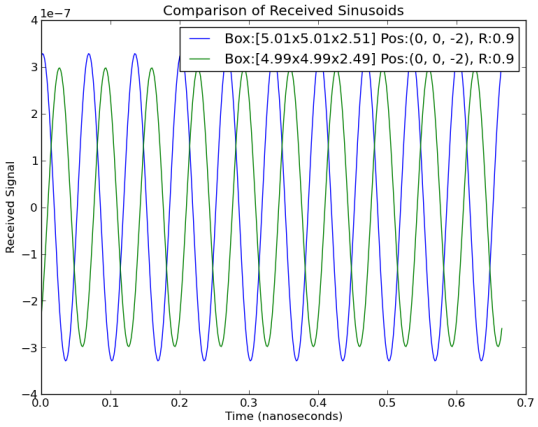
Where $s(i)$ is the i^{th} out of the 9 possible received signals, $E_s(i)$ is the energy of that signal, and k is the length of each signal (I took 10 cycles with 40 samples per cycle for $k = 400$). Note that the energy of each signal must be explicitly specified, because a source's position affects the length and attenuation of paths, which affects their amplitude and the constructive/destructive interference at the receiver. Hence, the energies of received signals from two different source positions can be vastly different. Figure 8 shows just how different the sinusoids can be from the 9 possible source positions. To generate the ROC curve for this scenario, I ran 100 experiments of signal + noise for every one of the 9 signals, and the same number of experiments for noise alone, for 300 different points on an ROC curve. The ROC points were obtained by calculating $\lambda(\vec{R_k})$ using the optimal detector, and counting true positives and false positives.

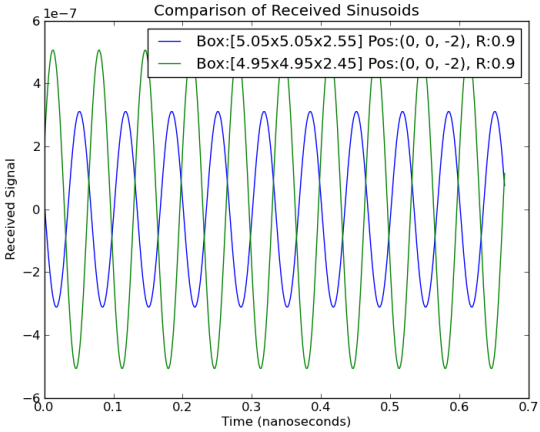
The second type of ROC curve considered is the case where the optimal detector with no uncertainty is used as defined before, but the uncertainty in room dimensions is actually present. The form of the detector

Table 1: The effect of errors in room dimension on the phase and amplitude of received sinusoids at 915Mhz (top) and 15Ghz (bottom) at 2cm error (left) and 10cm error (right)









is exactly the same, but now there are 11979 possible received signals. To generate this curve I randomly sampled 1000 possible room dimension choices and source positions, drawing from the Gaussian distribution for measurement uncertainty and from the uniform distribution for source position(cycling through every possibility would take far too long).

The third and final type of ROC curve is that for an optimal detector which incorporates measurement uncertainty. That detector is as follows:

$$\lambda(\vec{R_k}) = \sum_{i=1}^{9} \sum_{W=4.95}^{5.05} \sum_{H=2.45}^{2.55} \sum_{L=4.95}^{5.05} p(i,W,H,L)\exp\left(\frac{E_s(i,W,H,L)}{2\sigma_n^2} + \frac{1}{\sigma_n^2}\sum_{j=1}^{k} r_j s_j(i,W,H,L)\right)$$

Where W, H, and L are discretized in 11 points as described before, $E_s(i,W,H,L)$ and $s(i,W,H,L)$ are received signals for actual room dimension WxHxL coming from signal source i, and $p(i,W,H,L)$ is the probability of room dimensions WxHxL and source i (drawn from the independent Gaussian and uniform distributions discussed before; where the Gaussian distribution has been discretized to 11 points and re-normalized).

4.5 Generating Posteriors and MAP Estimation

I attempeted rudimentary source localization for the second and third cases (uncertainty not modeled and uncertainty modeled in optimal detector) used to generate the ROC curves. I used the sequential likelihood

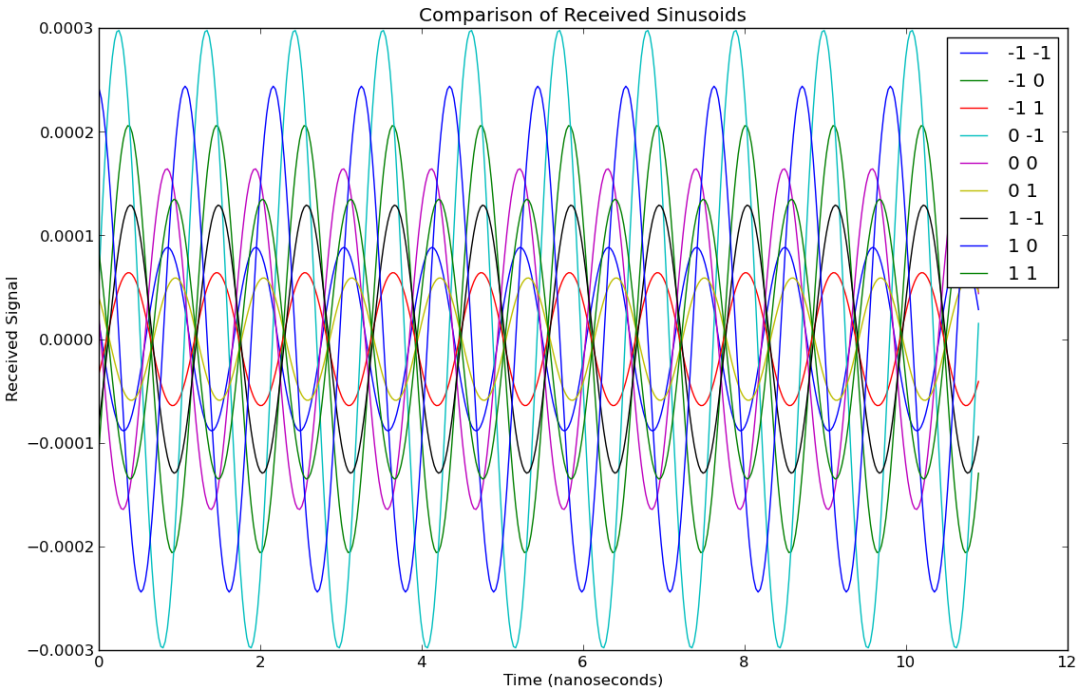


Figure 8: A comparison of all possible received signals from the 9 different source positions when the box's dimensions are known perfectly

ratio implementation in each case to generate posteriors, and estimated the source to be the one which had the maximum posterior value at the end. The equations for the first case, where the optimal detector assumes perfect room measurements, are as follows:

$$\lambda(r_k|\vec{R_{k-1}}) = \sum_{i=1}^{9} \lambda(r_k|s_i(k))p(s_i|\vec{R_{k-1}}, H_1)$$

$$\lambda(\vec{R_k}) = \lambda(\vec{R_{k-1}})\lambda(r_k|\vec{R_{k-1}})$$

$$p(s_i|\vec{R_k}, H_1) = \frac{\lambda(r_k|s_i)p(s_i|\vec{R_{k-1}}, H_1)}{\lambda(r_k|\vec{R_{k-1}}}$$

The prior starts out at $\frac{1}{9}$ for every signal

The update equations for the second case (with measurement uncertainty accounted for) are very similar, except the sums are taken over all 11979 possibilities (instead of just 9), which can be done because it is assumed the the source position and error in each dimension are independent. And the priors are set up as the $p(i, W, H, L)$ defined in the ROC section. Note that in this case we end up with posteriors over all 11979 signals, but we want to get posteriors for just the 9 source locations. Because of independence of error in width, height and length, the final posterior of each source is

$$p(s_i|\vec{R_{k-1}}, H_1) = \sum_{W=4.95}^{5.05} \sum_{H=2.45}^{2.55} \sum_{L=4.95}^{5.05} p(s_i|W, H, L, \vec{R_{k-1}}, H_1)$$

5 Results and Conclusions

5.1 ROC Curves

Plots of the ROC curves for the 3 different scenarios are shown in Figure 9. I chose a mid-range SNR that would give a high enough contrast between curves, and I set it up such that the average SNR over all of the possible 915Mhz signals was the same of that for the 15Ghz signals (it ended up being about 18). This ensured that the ROC curves would occupy similar regions in both cases. Also, the Pd axis is cropped to the range [0.5-1.0] so more contrast is visible.

A few clear trends are visible. First of all, for both frequencies, performance is only slightly worse when measurement error exists but is taken into consideration (green curve) than it is when perfect measurements are taken (blue curve). However, when it is assumed that measurement is perfect but uncertainty does exist (red curve), the results are degraded much more. Especially in the case of a 15Ghz signal, performance is significantly degraded when measurement error exists but is not modeled. Looking back at the sinusoids in Table 1 could explain the phenomenon behind this; the error in received signals is much greater in the 15Ghz case and will throw the optimal detector off much more than it will for the 915Mhz case. I was pleased to see, however, that once the uncertainty was modeled properly, the detector still performed fairly close to how it would with perfect measurements even in the 15Ghz case. This means that this technique of measuring room dimensions with a 3D scanner and modeling EM propagation may be feasible still at smaller wavelengths, as long as I do enough rigorous tests to come up with extremely accurate priors on the measurement error.

5.2 Posterior Results and Classification

I calculated the posteriors over the 9 signal locations for the case where measurement error exists but is modeled properly. The estimate of the source position is the MAP in each case. (I wanted to do the classification for case where it exists but isn't modeled properly, but the one remaining software bug I had yet to sort out resides in that script).

In the case of 915Mhz, taking the maximum posterior classified 6/9 correctly when measurement error was taken into consideration. The posteriors of one of the correctly classified signals are shown in Figure 10

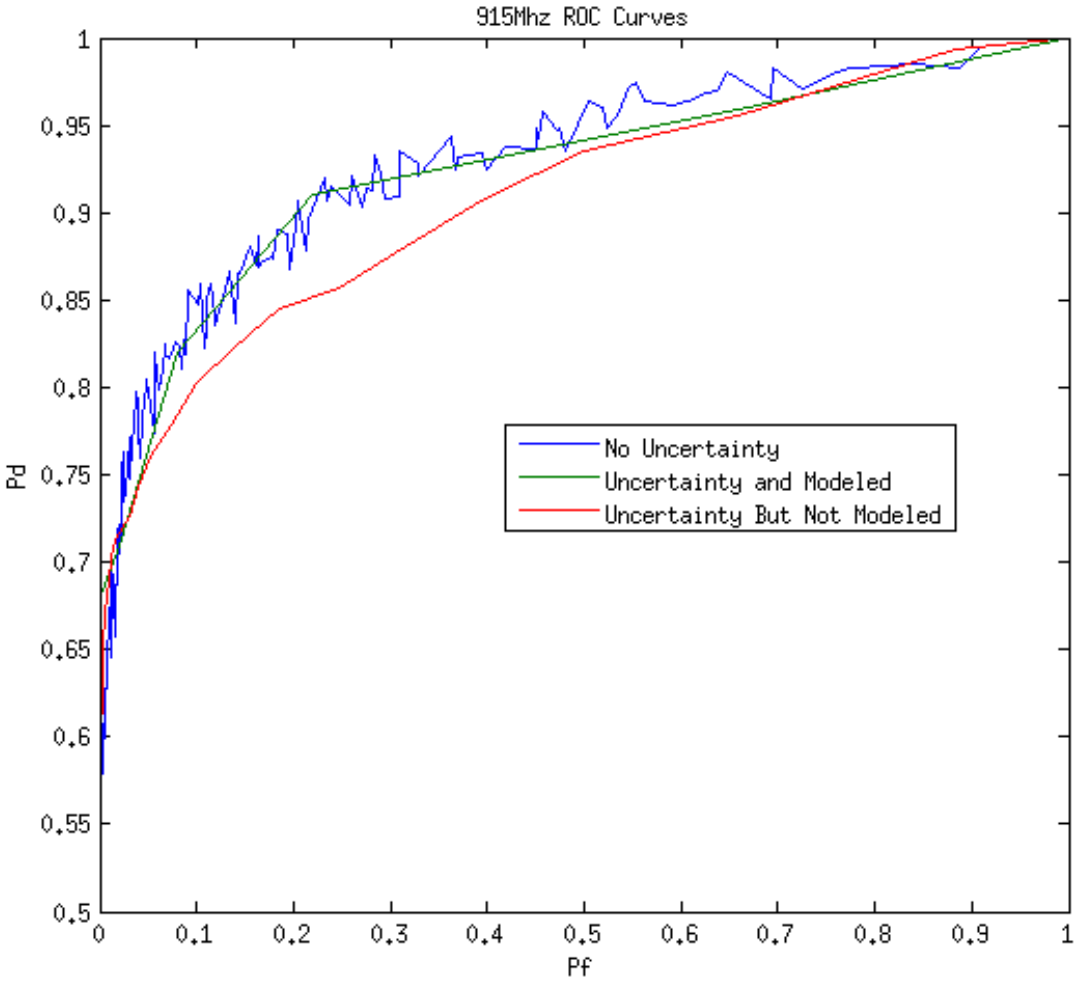
In the case of 15Ghz, taking the measurement into consideration only classified 2/9 correct, and failing to take it into consideration did not classify any correctly. The posteriors for one of the correctly classified ones in the case where measurement was modeled are shown in Figure 11. Though I need to look into this further, I believe the reason classification was worse here is because there is a much higher variation in the signal energies in the 15Ghz case than there was in the 915mhz case, which would make some far below average and more difficult to detect. So this is one caveat, that the classification performance may be more sensitive to the range of possible received signal energies than the detection performance is.

5.3 Extensions and Further Discussion

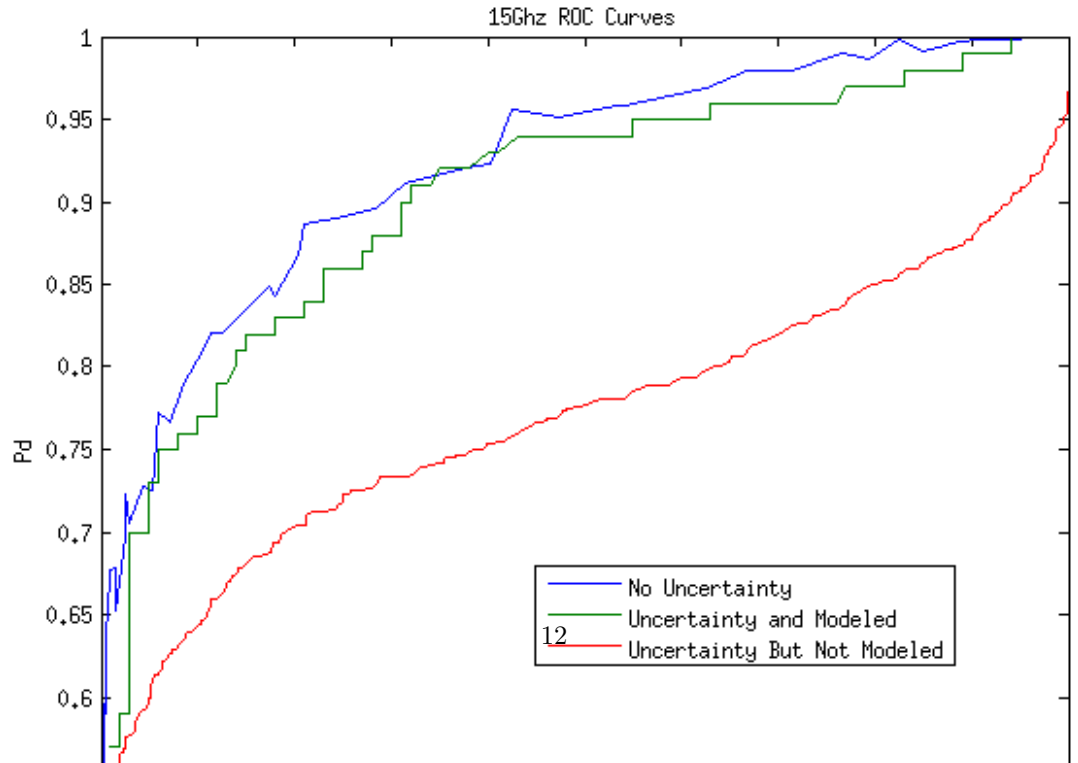
Due to time constraints, the only experiment I had a chance to implement was the one that modeled uncertainty over room size. However there are many straightforward extensions of this idea that my simulator can already handle. For instance, one question we have in lab is how will uncertainty of objects cluttering in space (such as moving people) affect the error in a received signal? I could test the feasibility of this idea by moving a person-shaped object around some radius of its measured position and doing the same tests. One caveat here is that my simulator does not model diffraction, which may become much more important if obstacles have geometric features on the order of a wavelength. This problem will certainly have to be dealt with.

Another extension is to vary the reflection coefficients of the room walls or objects in the scene, and to have some distribution over the uncertainty of those coefficients. This is an important question because estimating the reflection and absorption of objects in a scene is an extremely difficult problem. My simulator also currently supports specification of reflection coefficients, I simply fixed them all at 0.9 for the experiments in this paper.

One should also note that the SNR considered here is rather arbitrary, and was chosen simply to give a good contrast in the ROC plots. In practice, since the received sinusoid is of constant amplitude and phase, we should be able to make the SNR fairly high simply by taking more samples of the signal. Our only limit



(a) ROC Curves for 915Mhz signal



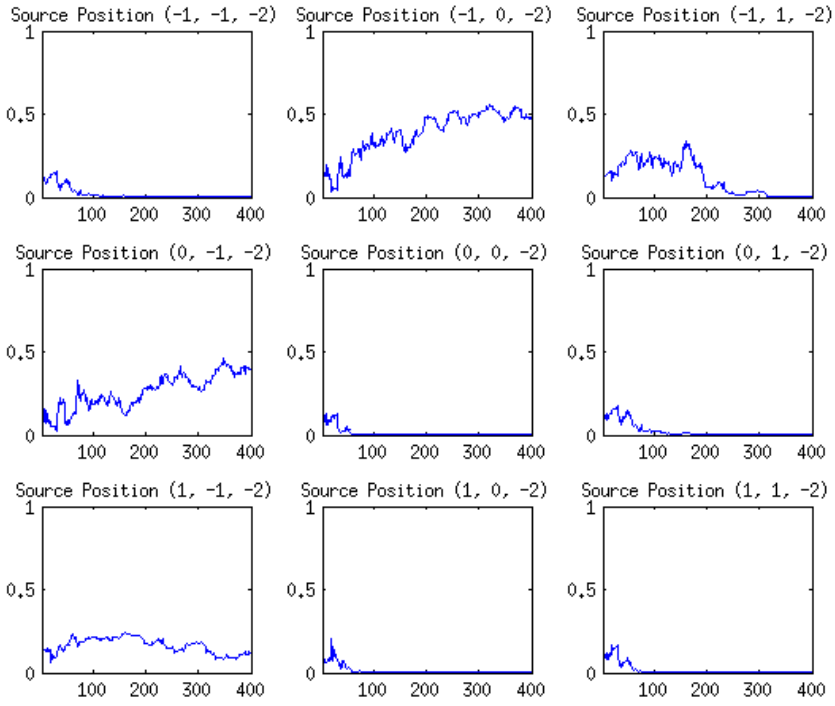


Figure 10: Posterior distributions that correctly identified a source at position (-1, 0, 2) at 915 Mhz when measurement error was taken into consideration

there is the amount of time the scene stays stationary, which depends on how quickly things are moving in the scene that would affect multipath. I have not had time to estimate these parameters properly in this project, but they are important and will need to be considered. Once a reasonable estimate of the maximum SNR given stationarity assumptions and receiver noise floor can be made, I would like to repeat this box experiment with many more sources and see how well localization can work, since I only took a very cursory look at localization in this paper

6 Source Code

The source code for this project can be found at the following public repository on Github

https://github.com/ctralie/G-RFLCT

This includes everything from geometric primitives, to visualization scripts, to the actual EM simulation. The most important files are EMScene.py, which implements the image sources method, and sceneView.py, which is the visualizer I made for this class. The apps/ subdirectory contains the files that I used to run the simulations in a systematic way for this project and generate my ROC curves and posterior distributions. The rest of the code in this repository deals with geometric primitive operations such as vector transformations and polygon model geometry and topology maintenance.

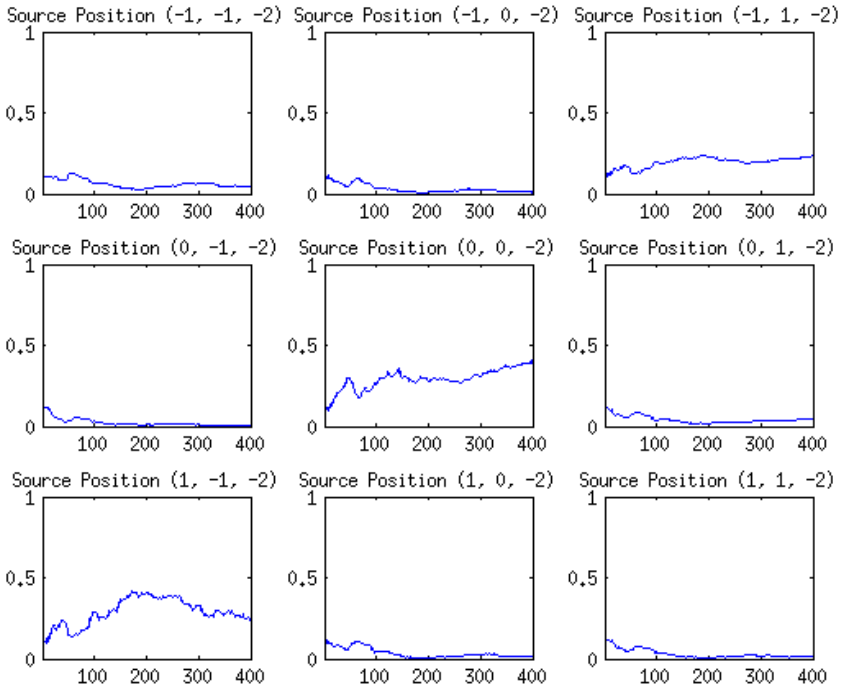


Figure 11: Posterior distributions that correctly identified a source at position (0, 0, -2) at 15Ghz