Classifying NLOS Propagation Channels by their Material Barrier

Nathan Brei (n_brei@mit.edu)

Robert Han (rhan@mit.edu)

16.62x Advisor:
Moe Win (moewin@mit.edu)

Submitted: December 8, 2010

Abstract

This paper explores using machine learning methods to identify and understand Non-Line-of-Sight situations through studying barrier's physical properties. Multiclass Support Vector Machine learning algorithms were used to identify material barriers based on the UWB radios channel pulse response. UWB waveforms were collected for 11 barriers at 5 locations in 5 different environments, representative of typical urban environments, and processed to extract a set of features such as rise time and delay spread.

A multi-class support vector machine (SVM) was used to identify the barrier materials. Although a direct analysis (ignoring environmental differences) yields an overall successful identification rate of only 31%, results can be significantly improved (upward to 70%) by constraining the environments. By applying the classification to a single environment, the success rate classification about doubled. This method also failed to account for the thickness of the material. Finally, looking into the feature space allowed for possible adjustments in the set of features used to improve classification results.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Significance

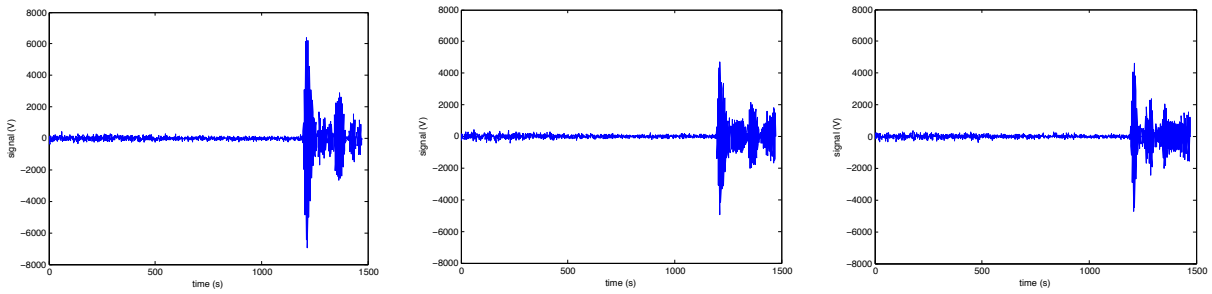
Localization, determining one's position, through Global System for Mobile Communications (GSM) and Global Positioning System (GPS) has been around since the early 1990s. As the technology has matured, localization products have taken on increasing prominent roles in both recreational applications such as geo-tagging and life-death situations such as rescues and military applications. However, both the commonly used GPS and GMS localization services perform poorly in dense non-line-of-sight (NLOS) urban environments due to environmental interference. One way to resolve the existing limitation is to supplement the existing long range oriented GPS and GSM localization services with short range Ultra-wideband(UWB) radio based localization systems that provides high resolution small scale localization. Using low-powered-UWB-Impulse radios waves for localization, UWB impulse radio is well suited for NLOS environments due to its multi-path channel propagation properties. Unlike the GPS or GSM localization techniques that triangulate using external high-powered static reference points, UWB localization employs nearby low-powered reference points that are less prone to interference. Furthermore, UWB impulse radio takes advantage of time of arrival (TOA) data points that allow for multipath propagation, enhancing its reliability and resolution.

1.2 Overview

Like other forms of localization services, UWB is also vulnerable to NLOS distortion. Numerous previous studies have proposed various statistical methods to compensate for the overall NLOS distortion. However, the previous works have grouped various NLOS barriers into general groups due to location or use such as parameters as office building or farmland. Article [8] by Win and Scholtz takes a new approach to NLOS channel interference detection by combining machine learning methods with observed impulse responses to classify obstructions through its physical characteristics. Thus this research lay the foundation for future research in applying machine learning and dynamic environmental awareness to improving accuracy of NLOS distance measurements. To test whether it is possible to identify a barrier's physical composition through distortions within UWB pulse responses, we designed a two stage experiment. The first stage is to measure the effect of various material barriers have on the channel propagation's statistical features such as rise time and kurtosis. Then armed with data from stage 1, we will implement machine learning technique to classify materials based on NLOS pulse response's distortion.

Figure 1.2 shows some sample waveforms. They visually differ according to the material barrier when the other variables are fixed. In a general setting, however, the waveform is strongly a affected by several kinds of interference, notably multipath propagation, diffraction, and noise.

This experiment shall determine whether or not this 'signature' can be detected even when the



(a) A line-of-sight waveform (b) Obstructed by a piece of wood (c) Obstructed by a piece of metal

Figure 1: Examples of line-of-sight and non-line-of-sight waveforms for different material barriers

location and environment varies. A number of different barriers, made of different construction materials, shall be tested. Testing entails sending a UWB pulse through the barrier and capturing the pulse response. For each barrier, a number of waveforms shall be recorded at different locations and a small set of identifying features shall be extracted from each. A machine-learning algorithm shall attempt to identify the barrier material from the waveform features.

2 Hypothesis, Objective, and Success Criteria

2.1 Definitions

Propagation channel A linear system describing how an ultra-wideband signal is distorted by the environment during its transit.

NLOS Non-line-of-sight. Any propagation channel corresponding to the situation where the direct path between the transmitter and receiver is blocked by a barrier.

Material barrier A medium between the transmitter and receivier. Changing the material barrier changes the propagation channel.

Statistical classification Given a set of discrete categories, in this case 'barrier materials', identify which category the item belongs to.

2.2 Hypothesis

It is possible to classify different NLOS propagation channels, from their pulse response, according to the material of the obstructing barrier, with a rate of accuracy ($\geq 60\%$.).

2.3 Objectives

Devise, fabricate, and use an experimental setup to capture the pulse response of a channel obstructed by a material barrier. Implement a machine-learning algorithm to classify the propagation

channel, according to the barrier material, from the channel pulse response.

2.4 Success criteria

Evaluate the probability of correct classification using test data.

3 Literature Review

The underlying theory of impulse radio is detailed in several papers by Win and Scholtz in 1998 [1] [2]. Much of this work focuses on fundamentals– particularly emphasizing multiple access issues and the design of Rake receivers. Another 1998 paper by Win and Scholtz, [3], discusses the robustness of UWB signals and suggests that it is advantageous for low-power, indoor communications. Though it is stated in [1] that UWB technology has existed for decades, it is clear that it is just now starting to 'take off'.

Much of the further literature is devoted to finding a *channel model*– basically a mathematical model which replicates the distortions a real-world signal would experience. This is of paramount importance for systems designers, but the wealth of literature on the matter suggests that a satisfactory model was difficult to achieve. Win and Scholtz discuss this in a 2002 paper. [4] They state that existing, narrowband channel models proved inadequate. A key problem is multipath– instead of blending, the different multipath components remain distinct at higher bandwidth. Approaches to establish a general indoor channel involve performing exhaustive experiments, performing statistical analyses, and running simulations. After many publications, including another 2002 paper by Cassioli, Win, and Molisch [5], a standard channel model was established by an IEEE task group in 2003. [6]

Meanwhile, UWB technology had been widely recognized [3], [7] as being well suited for radar and *location-aware* networks owing to the lack of multipath fading and the low power spectral density. A broad overview of the basics of UWB localization can be found in Gezici et al.'s "Localization via Ultra-Wideband Radios" [7]. The authors describe multiple localization strategies with regard to their pros and cons: localizing based on angle-of-attack measurements, signal strength, and time-of-arrival are all considered. Time-of-arrival was determined to be the superior method; later papers [8] suggest that it is used almost exclusively. Three sources of error are presented as particularly problematic in [7]: multipath propagation (where multiple reflected copies of the same signal interfere), multiple access interference (where signals from different nodes interfere), and non-line-of-sight propagation (where a barrier distorts or blocks the main signal)

Of these, it is the NLOS problem which is of highest relevance to this experiment. When the radio wave encounters a barrier, part of it is reflected, part is diffracted, part is scattered, and part penetrates, but moves at a slower speed. Thus, the signal does not reach the barrier until some time later, depending on the properties of the barrier and environment. The time delay effects

a positive distance measurement error. The authors argue that the solution to the problem is twofold: Firstly, use information from *all* of the other nodes in order to identify those that are most erroneous. Secondly, analyze certain waveform properties to statistically determine whether a signal is LOS or not.

Later authors offer more comprehensive overviews of different strategies. The NLOS problem is broken into two components: identification and mitigation. Schroeder et al., 2007, [9] provide an overview of identification techniques. Algorithms considered look at range variance estimates, features of channel impulse responses, and position (map-based) estimates. A 2009 "Survey of NLOS identification and error mitigation" [8] goes several steps further, creating charts directly comparing different identification techniques. The consensus in both papers is that channel pulse response techniques are superior to the range or position techniques owing to their lack of latency and computational cost. The channel pulse response techniques rely on different features, but generally involve thresholding and/or hypothesis testing.

Of the techniques covered, the majority of them assume the existence of precisely two possible channels: LOS and NLOS. When a greater granularity is needed, the techniques use highly generalized channels like 'modern office'. [5] An exception is a 2007 paper by Mucchi and Marcocci [10], who found that by analyzing the 'kurtosis' of a waveform, they were able to differentiate the channel *quality* of different environments, namely: LOS, quasi-LOS, NLOS, and extreme-NLOS.

An extremely recent development involves using machine learning techniques for NLOS identification instead of traditional statistical/probabilistic ones. Maran and Gifford [11], [12] demonstrated that the use of a support vector machine (SVM), which bypassed the traditional channel model completely, "outperformed previous parametric techniques."

The experiment proposed here seeks to build upon this body of research. Its chief contributions are as follows:

- It attempts a much more granular channel classification.
- It implements an SVM, building upon the work of Maran and Gifford.
- It doesn't explicitly assume a channel model
- It implicitly assumes that the channel is the superposition of an LTI 'barrier' system with an LTI 'multipath and noise' system.

4 Experiment Description

4.1 Overview

In the most basic sense, our experiment involved recording and analyzing the distortions in UWB pulse response through an obstruction. From that data, we applied machine learning techniques to

classify the barrier material. Our experimental setup consisted of two sub-components, the fixed apparatus setup and the variable material barrier.

4.2 Apparatus

To ensure a controlled data collection, we fabricated a radio setup consisting of an UWB radio transmitter on one end, a barrier in the middle, and a UWB receiver on the other. The radios were mounted on either side of a platform so that the material barrier bisected and was perpendicular to the shortest line between the two radios. The distance between two radios was fixed. A computer sent an initial impulse to the transmitter and collected the resultant waveform from the receiver.

In order to ensure portability across different environments and reduce the effect of background noise, we conducted our data gathering using a mobile fixed apparatus in 5 different environments, each representative of a potential operating environment. Environments being tested were open space within modern buildings such as Stata Center, enclosed space within modern buildings, open space within classic building such as the Building 10 first floor lobby, enclosed space within classic buildings and spaces immediately outside of buildings. For the purpose of the project, we were defining classic building as building constructed before 1990s and modern buildings as building constructed after 1990s. The two categories reflected the transformation in building techniques over time.

4.3 Test Articles

In order to ensure our research results were applicable in an urban environment, we used commonly found construction materials for barrier materials. We set up these materials in the testing apparatus to identify basic parameters on NLOS channel propagation. Ultimately, we processed these channel propagation into feature sets such as rise time allowing for SVM methods to be used to identify the material type.

As referenced in our test matrix, Figure 6, we used plywood, drywall, steel, insulation foam and plexiglass as materials for the barrier. Each one of our barrier materials represented of one category of materials commonly found within construction as referenced by Table 2. Togather thickness per material data, we had our barrier setup with two thickness for each material.

The test articles consisted of a collection of planar construction materials, 2 ft by 2 ft. These materials were chosen with several considerations in mind.

- **Number.** More materials equated to more physical labor. Attempting to reduce complexities in switching out more than 250 times.
- **Prevalence.** Materials should be common construction materials, to increase the real-world applicability of the results. This also simplifies acquisition.

- **Characteristics.** If the materials are too alike, identification becomes harder. Materials should be fundamentally different. However, running multiple thicknesses of the same material would allow for some interesting analyses.

- **Cost.** Materials must remain within budget.

The test articles were square, planar samples of construction materials. They were of the same size, to reduce the diffraction error. Two thicknesses of each was tested.

Table 1: Material barrier design

Material	Size	Thickness	Representative Category
Plywood	2' by 2'	3/4"	Timber, organic composite of cellulose fibers
Steel	2' by 2'	0.12"	Metallic and conductive material
Drywall	2' by 2'	1/2"	Low density plaster
Foam	2' by 2'	1.5"	Synthetic low density insulation
Plexiglass	2' by 2'	1/4"	Plastics solid material

Figure 2: The test matrix.

		Plywood	**Plywood (+2)**	**Drywall**	**Drywall (+2)**	**Steel sheet**	**Steel sheet (+2)**	**Insulation Foam**	**Insulation Foam (+2)**	**Glass**	**Glass (+2)**
Modern building (Stata) open area	1 ⋮ 5										
Modern building (Stata) hallway	1 ⋮ 5										
Older building (Maclaurin) open area	1 ⋮ 5										
Older building (Maclaurin) hallway	1 ⋮ 5										
Outdoors	1 ⋮ 5										

4.4 Fabrication

The purpose of our apparatus was to provide reliable means of transporting our experiment setup between various locations while maintaining the radios in a consistent configuration. The measurement systems consisted of UWB radios for data collection, see Section 5.1.1. A rolling cart that served as a base, more details in Section 5.1.2. There will be a custom made collapsible radio mounting mechanism, as described in Section 5.1.3 and a custom made barrier holder that allowed for quick switch of barriers more details in Section 5.1.4. Finally, there will be additional electronics to operate the radios as specified in Section 5.1.5.

Figure 3: The apparatus.

4.4.1 Radios

A pair of Time Domain PulsON 220 UWB Radios was used for data collection. This radio is one of first UWB radios that abide by FCC's newly drafted operating standards for UWB radio. It has a compact footprint and provides high-resolution data representative of a UWB in commercial usage. Setup requirements are AC input of 120/240V and Ethernet connection for data transfer to PC. We employed a pair of them with one as client, requester, and another as server, responder. They will be powered by running an extension cord to the nearest power source.

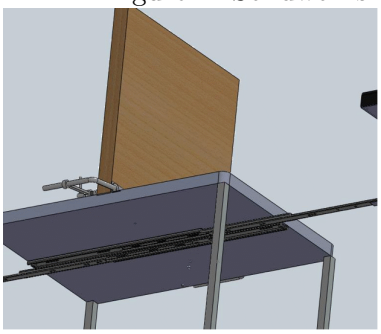
4.4.2 Cart

A 3" by 2" cart served as the foundation of our apparatus to which the radios, barrier and additional electronics were mounted to Additionally, it also acted as a mobile transport that enabled rapid movement between different environments without disturbing our radio setup. We used a plastic cart in the event that the metal from a cart interferences with radio transmissions.. As a base, we required a two level cart with the top level dedicated to the apparatus setup and a bottom shelf for storing additional electronic components for required for data collection. The plastic cart came with 2 shelves and the added bonus of a flush top shelf, allowing use to clamp a piece of plywood

with our experiment on top of it.

In regards to mobility, the light duty plastic cart was ideal due to its small 3" by 2"footprint, allowing it to maneuver easily through doorways and hallways. Additionally, it has sizable 5" wheels, allowing for easy movement through wider range of uneven terrains with delicate electronics onboard.

Figure 4: Solidworks rendering showing the underside of sliding drawer placement



4.4.3 Radio attachment

The goal of radio attachment was to keep the UWB radios, referenced in section 5.1.1, in a consistent fixed orientation during data collection and keep the UWB radios out of way during transport. For that end, each UWB radio is attached on aluminum mounting plate. This aluminum mounting plate was attached 1'by 1" by 1" hollow aluminum vertical extrusion that was attached to a 26" steel high precision drawer slides. The steel high precision drawer slides was mounted flush to a piece of plywood clamped to the cart's top shelf. The detailed construction ensured consistency throughout various environments. The high precision steel drawer slides enabled the radios to be retracted like the sliding drawers of a desk while not in use, allowing for reduced footprint while in motion.

The mounting plates was cut on the water jet with mounting holes that matched UWB radio's existing threaded mounting points for shock dampers. Each of the 1'by 1" by 1" hollow aluminum vertical extrusion had a L-bracket on each end, enabling the radio base plate and the steel high-precision drawer slide to be directly attached to the aluminum extrusion. The L-bracket on the aluminum vertical extrusion was secured through pre-drilled attachment holes.

4.4.4 Barrier Attachment

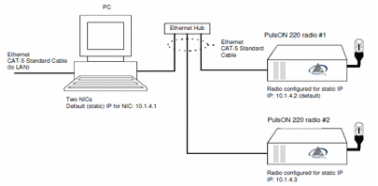
The barrier attachment was designed to securely attach various barrier materials securely to the top of the plywood, clamped to the cart. The design was composed of a 2' 90 deg angle perforated steel and two 6" medium-force bar clamps on each side. The 2' 90 deg angle perforated steel was securely bolted on to the plywood providing a rigid clamping point for the barrier. There was

a spring clamp on either side of the barrier to secure the barrier to the 90 deg angle steel bar. The spring clamp allowed for quick-release enabling for faster switching between various barrier materials.

4.4.5 Electronics

In addition to the items described the cart also had several additional electronics components for data collection and radio operation on the bottom shelf. For data collection, an IBM T43PThinkPad Laptop with an additional 3Com Ethernet PCI card will be used to interact with both radios and perform data collection. For power, a power strip with an extension cord was used to power the UWB radios, and the laptop.

Figure 5: Suggested radio setup by Time Domain



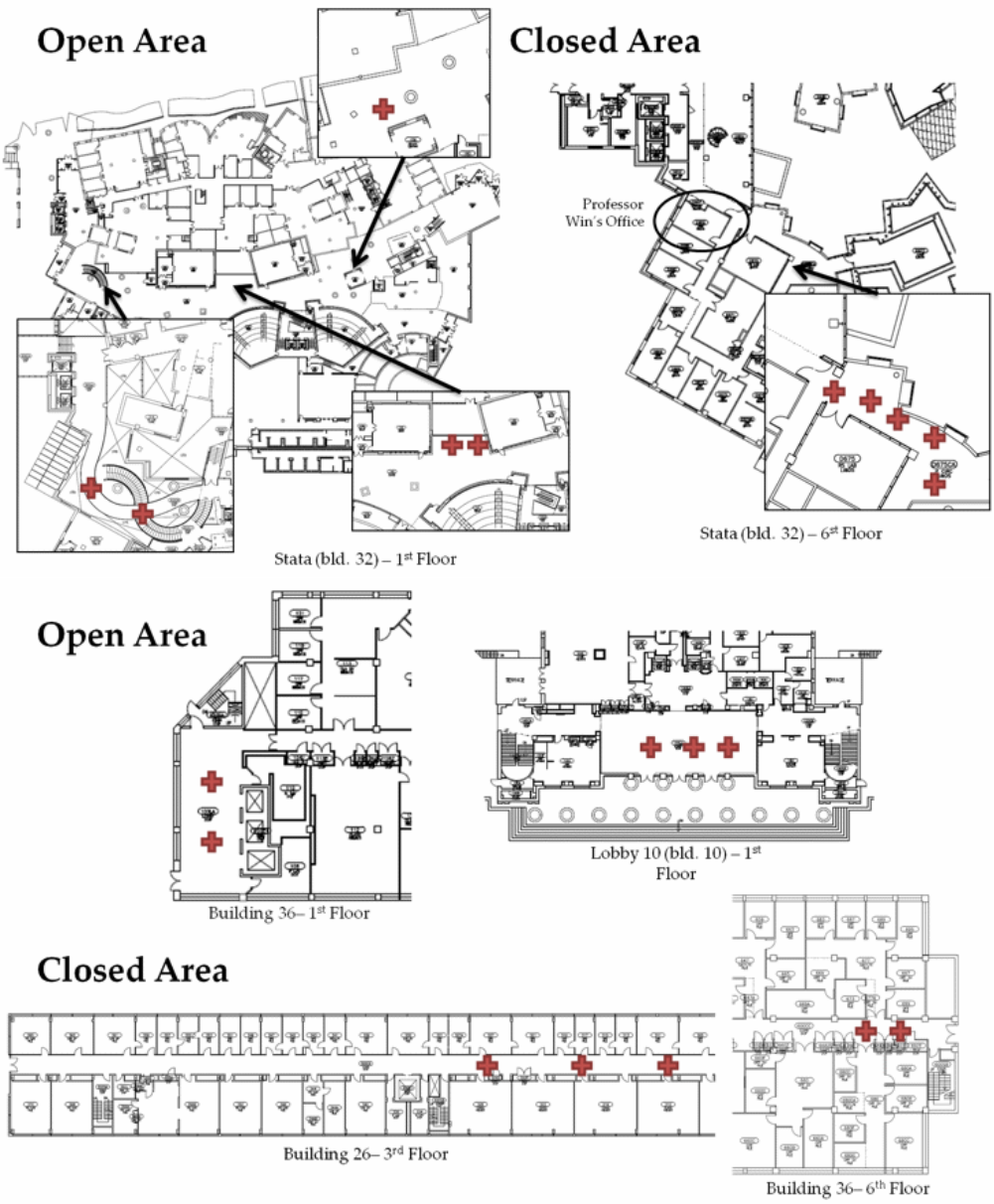
4.5 Data Collection

In terms of data collection, we collected waveforms of UWB channel propagation, r(t). The goal of data collection is to both provide a set of training data for support vector machine learning technique, further discussed in section 6.3, and also to provide data for back-testing and verification purposes further discussed in section 6.4. For each barrier material in Table 2 Test Matrix, we ran the our data collection program at 5 different location in each of our 5 testing environments. This generate a total of 25 location at which we collect data on each of 11 barriers including control, totaling 275 trials. Each execution is defined as using Time Domain's Ranging Analysis Module to collect 100 responder waveforms and saving these waveforms as .txt file.

During data gathering in order to minimize effects from background noise and other inconsistencies we done the following. We gathered data in areas devoid of people. We kept the apparatus stationary. We kept any physical objects aside from the barrier being tested away from the radios. We will keep the testing rig away from any source that may present interference such as a satellite antenna. In event of failure to do the above and external interfere with the data collection occurs, we will discard and replace significant out of sample waveforms, defined as waves forms more than 2 standard deviations away from the mean.

We collected data in various environments as shown by maps on the following page. We collected most of the data at night, during low traffic hours.

Figure 6: Collection locations



4.6 Data Processing

Moving on to data processing, the goal of data processing was to convert the waveforms in .txt format as originally outputted by to a Matlab friendly format. Once in Matlab format, the data can be further used either for training or back-testing. To achieve that end, we used a Matlab script to convert raw .txt file output into Matlab format. Ultimately, we obtained a series of received waveforms, r(t).

Figure 7: Formulae for metrics calculations

Table 4 Machine Learning Features

Features	Equation
Energy of the received signal:	$\Sigma_r = \int_{-\infty}^{\infty} \lvert r(t)\rvert^2 \, dt$
Maximum amplitude of the received signal:	$r_{\max} = \max\lvert r(t)\rvert$
Rise Time:	$t_{rise} = t_H - t_L$ where: $t_L = \min\{t : \lvert r(t)\rvert \geq \alpha \cdot \sigma_n\}$ $t_H = \min\{t : \lvert r(t)\rvert \geq \beta \cdot r_{\max}\}$ σ_n is standard deviation of thermal noise $\alpha > 0$ and $0 < \beta \leq 1$ are chosen empirically to capture rise time
Mean excess delay:	$\tau_{MED} = \int_{-\infty}^{\infty} \frac{t \cdot \lvert r(t)\rvert^2}{\Sigma_r} \, dt$
RMS delay spread:	$\tau_{RMS} = \int_{-\infty}^{\infty} \left(t - \tau_{MED}\right)^2 \cdot \frac{\lvert r(t)\rvert^2}{\Sigma_r} \, dt$
Kurtosis:	$\kappa = \frac{1}{\sigma_{\lvert r\rvert}^4 \cdot T} \int \left[r(t) - \mu_{\lvert r\rvert} \right]^4 dt$ where: $\mu_{\lvert r\rvert} = \frac{1}{T} \int \lvert r(t)\rvert \, dt$ $\sigma^2_{\lvert r\rvert} = \frac{1}{T} \int \left(\lvert r(t)\rvert - \mu_{\lvert r\rvert} \right)^2 dt$

4.6.1 Procedure

1. Import data into Matlab. The PulsOn software sequentially dumped each waveform to a numbered text file. The raw waveform data and the timestamp must be extracted and stored in an array in Matlab.

2. Correlate the waveforms with the respective materials. A Matlab script processes the environments, locations, and barriers associated with each sample. It then compares the timestamps to correlate the location and material IDs with each waveform. The respective location and material IDs were stored alongside each waveform. Waveforms captured while the barrier was in motion must be removed.

3. Calculate the waveform features. Each waveform had its features calculated and stored as a vector.

4. Input training data into the SVM. 1/10th of the waveforms are to be randomly selected. Their feature vectors are input as into the SVM's train() method along with their material ID.

5. Input test data into the SVM. Feature vectors for all of the waveforms were inputted into the SVM's test() method, without their material ID. The SVM returns the (predicted) material ID. Allowing for the success rates for plots to be calculated.

4.6.2 Machine Learning

The goal of the training was to use a subset of the data to calibrate Support Vector Machine(SVM) for classification of test data. We randomized the data set and subsequently took the first 200 or 10% samples for training. We evaluated the waveform features from this set of training data for a multiclass SVM. The waveform features were identified in document [4], a previous paper on the topic published by our advisor Professor Win, as ideal parameters for classifying NLOS data. We used Spider Matlab Machine Learning Toolbox for calculations. We picked the Spider Machine Learning Toolbox and Matlab because our familiarity with Matlab. The Spider Toolbox provided multiclass support and excellent user documentation.

Step 1, we calculated waveform features in accordance with the equations provided above in Table 4. r(t) equals the received waveform data as described in section 6.1. Step 2, we implemented Spider Toolbox's Multiclass machine learning algorithm to generate the machine learning predications. Finally, we performed back-testing to assess success rates.

4.6.3 Calibration and error mitigation

The multiclass support vector machine utilized an RBF kernel, chosen for simplicity and flexibility. The two parameters it took were γ, the RBF kernel hyperparameter, and C, the penalty for

overstepping decision boundaries. C and γ were initially determined via an automated grid search on the preliminary data set. C was left at infinity, and γ was manually iterated to be optimal for the entire dataset. All of the figures and numbers were calculated at $\gamma = 0.035$, except for the 'ignoring barrier thickness' bubble chart.

The experiment called for unorthodox error mitigation strategies. On one hand, there was zero uncertainty in the environment and barrier parameters. On the other, there was unbounded uncertainty with the waveform measurements. The choice of machine learning parameters was never intended to be optimal, but rather to act as a reasonable upper bound. The experiment was intended to see how much accuracy could be achieved, rather than to calculate a number to a desired level of accuracy.

With more time, the results would have been cross-validated, where each machine learning experiment would be repeated with n sets of different training data. While this would have made the final results more rigorous, it would have been unlikely to change the results, considering that the data was already randomized and that there was 140x more testing data than training data.

5 Analysis

5.1 Mathematical background

A simplified LTI model of the channel can be derived from raytracing. Let $x[t]$ be the transmitted signal and $y[t]$ be the received signal. Then,

$$y[t] = x[t] * (b[t] + m[t] + d[t]) + n$$

where

$$b[t] = \alpha_0 \delta[t - \tau_0] \tag{1}$$

$$m[t] = \sum_{i=1} \alpha_i \delta[t - \tau_i] \tag{2}$$

In this case, $b[t]$ describes the impulse response of the NLOS 'transmission path', attenuated by some factor α_0 and time-delayed by some constant τ_0. Likewise, $m[t]$ describes the multipath (reflection path), where each reflected copy of the signal that propagates to the receiver has its own attenuated and delayed spike. $d[t]$ describes the impulse response of the path that diffracts around the edges of the barrier. In the context of this experiment, $b[t]$ with the barrier material, while $m[t]$ varies with the environment and location parameters. $d[t]$ is a function of the barrier geometry, which is held approximately constant. n represents additive white Gaussian noise.

It must be noted that there are several problems with this model. Most importantly, this model is deterministic, whereas the real-world behavior is more stochastic. The experimental data shows

that $y[t]$ is certainly not constant, even when no parameters are varied. Of 100+ samples taken for each barrier at each location, no two are alike. The most obvious improvement to this model would be to recast everything in terms of random variables. The model also ignores details such as the distortion from both antennae. If one were to try to de-convolute $y[t]$ to find $b[t]$, etc, this model would need considerable refinement. However, this model is used simply to provide qualitative insight, and should be treated as a best-case scenario.

From the model, several things become clear. For the machine learning to be able to identify barriers successfully regardless of location, the barrier $b[t]$ should affect the waveform $y[t]$ in ways that the multipath $m[t]$ doesn't. For example, if $\tau_0 < \tau_i, \text{where} i > 0$, $b[t]$'s spike would always appear before the multipath spikes appeared. Thus, metrics calculated from the first spike, such as rise time, would be a function of $b[t]$ and not $m[t]$. Another example: If $\alpha_0 > \alpha_i, \text{where} i > 0$, then the spike from $b[t]$ would have a greater amplitude than all of spikes from $m[t]$, and the 'max' metric would characterize the barrier well regardless of location.

On the other hand, if $b[t]$ and $m[t]$ are indistinguishable in the received waveform, then none of the metrics would be able to characterize the barrier independently from the environment/location. Machine learning could still achieve success, but it would be 'tethered' to locations that it had already been trained on.

5.2 Results

The experiment's success criterion was to 'evaluate the probability of correct classification using test data.' This was achieved as shown in Figure 8. Using every sample collected, with a training set of 200 samples, an RBF with $\gamma = 0.035$, and no noise reduction, **the correct classification probability was 31%. This result invalidates our hypothesis.** However, the dataset lends itself well to further analyses. Under certain conditions and with different independent variable choices, the hypothesis can in fact be met.

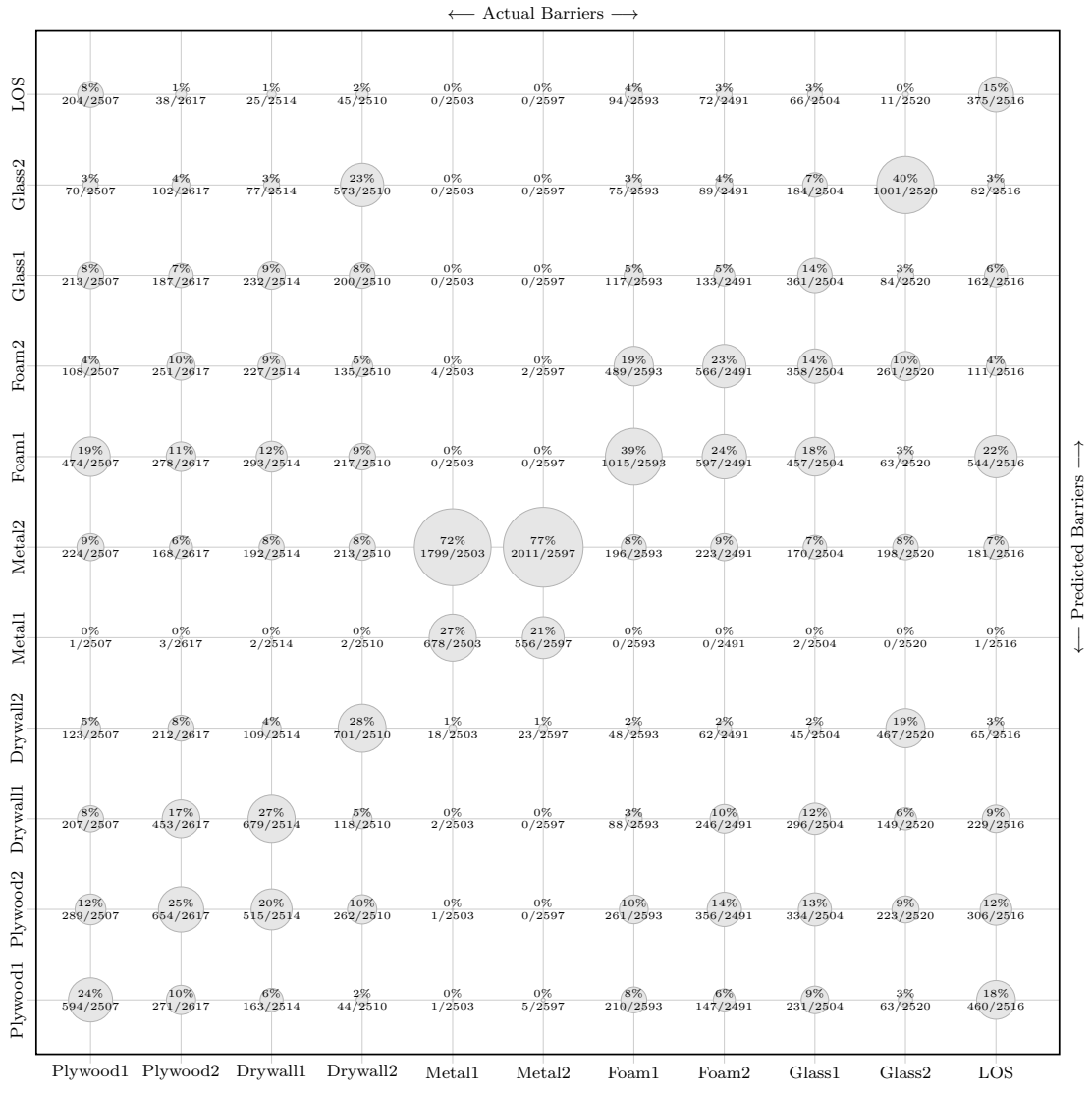
The following 'bubble chart' describes the key result of the experiment. The chart is to be read as follows: The x-axis indicates the actual barrier that was affixed to the apparatus, while the y-axis indicates the barrier that the machine learning predicted. The area of the bubble at each coordinate is proportional to the probability that this identification was made. For example, the bubble at $(metal1, metal2)$ tells us that the single-layer steel sheet was misidentified as double-layer steel sheet 72% of the time, or 1799 out of 2503 samples of that barrier. Thus, the 'correct' identifications are all along the diagonal $x = y$. A bubble chart with the largest bubbles along the diagonal shows considerably better results than a bubble chart where the bubbles are evenly distributed.

It is important to remember that the ordering of the barriers along the axis is not meaningful. The barriers are neither ordered according to physical characteristics nor grouped to show patterns in the misidentification. (This shall be justified shortly.) Also, the machine learning algorithm

chose the best possible classification from the *entire* set of barriers, i.e. completely independently from the ordering of the barriers along the axes.

Figure 8: Main results

All data [31 percent success]



The 30% figure, while not particularly good, is still considerably better than a random guess, which would give a 9% success rate. However, by slicing the data more finely and only looking at a particular subset of it, we can raise our success rate considerably. The most obvious choice is to only look at the five locations corresponding to a single environment. When only training on data within a single environment, we get a success rates of 57%, 41%, 65%, 50%, and 36%, for

environments 1 through 5, respectively. Figures 9 and 10 show the full bubble chart for two of these environments; the rest can be found in Appendix A.

This performance improvement can easily be explained. Firstly, by simply reducing the number of locations, we are reducing the variability in the data set. Secondly, by limiting ourselves to a single environment, we are ensuring that all the multipath, $m[t]$, correspond to straightforward patterns. (If the channel was properly LTI, there would be a unique multipath pattern for each location, however the real $m[t]$ behaves more like a random variable) Thirdly, limiting to a single environment can further reduce the set of different multipath components via symmetry. For example, environments 1, 3, and 4 were situated in lobbies and hallways where the walls were at right angles. Locations were parallel and reasonably close together. In contrast, environment 2 consisted of locations that were far apart and subject to walls with asymmetric geometries, while environment 5 was outdoors in locations subject to both odd geometries and vegetation. The symmetry of the former environments' geometry results in more homogeneous multipath, which leads to the significantly higher success rates.

Figure 9: Constant environment – Best

Constant environment = 3 [65 percent success]

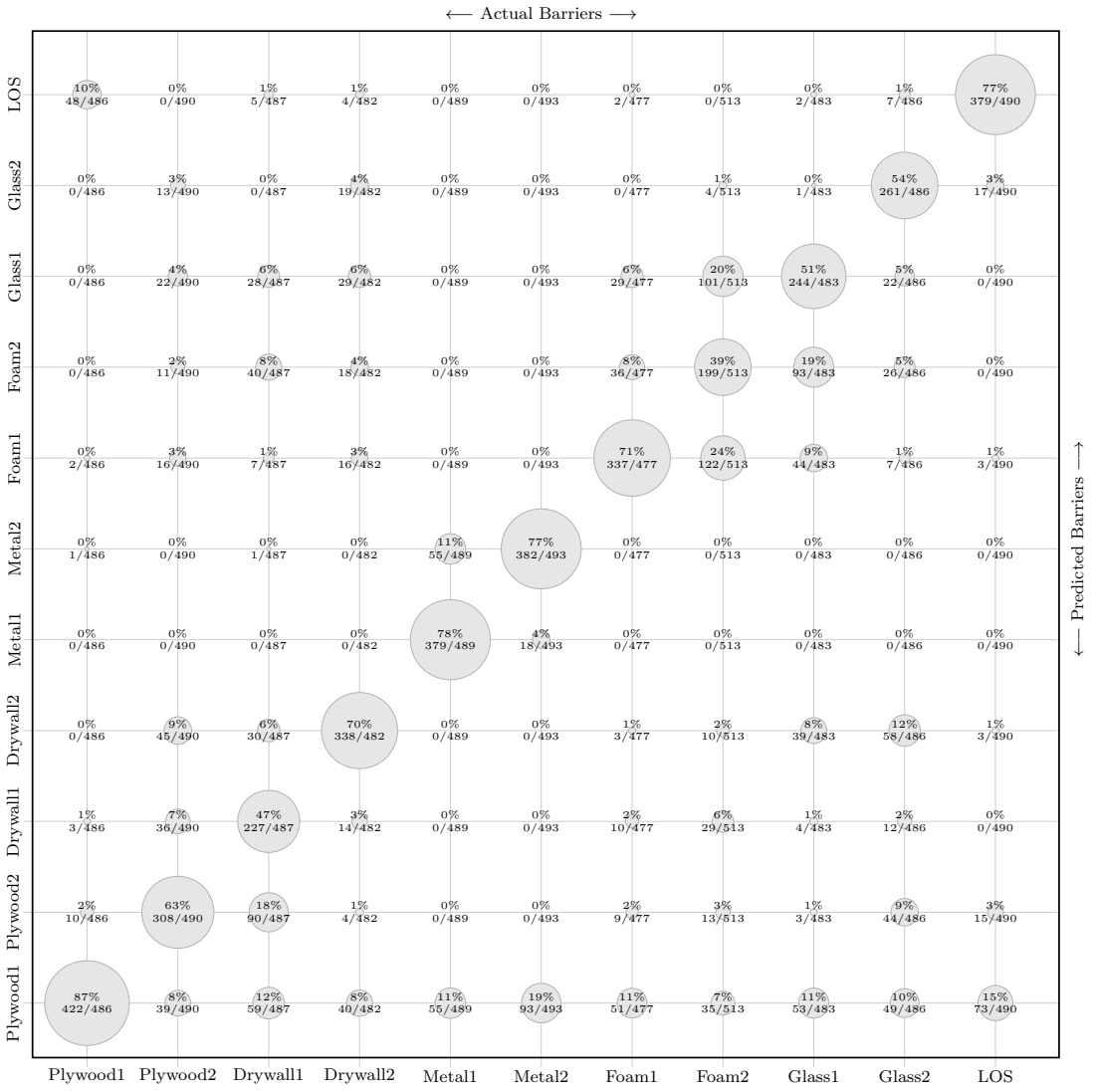
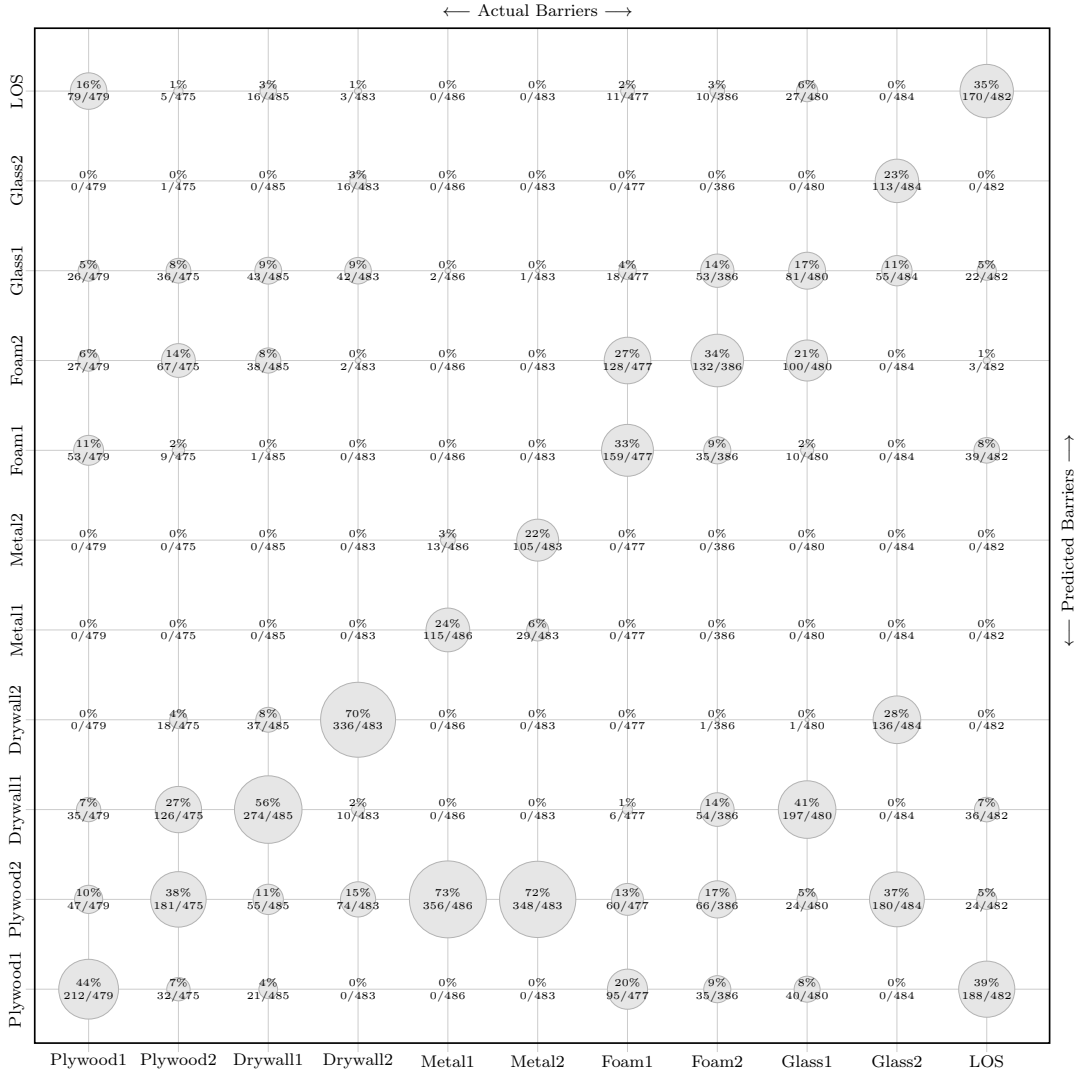


Figure 10: Constant environment – Worst

Constant environment = 5 [36 percent success]



The performance improvement that comes from restricting the data set has an upper limit. Since the channel is not deterministic, there is variation in the multipath even when both the environment and location are held constant. One example is shown in Figure 11. Although the success rate in this case is reasonably high at 59%, it is still far less than it would be if the channel were deterministic and, consequently, the multipath was constant. With the removal of the extra variation, however, some trends (already visible in the preceding figures) become very apparent.

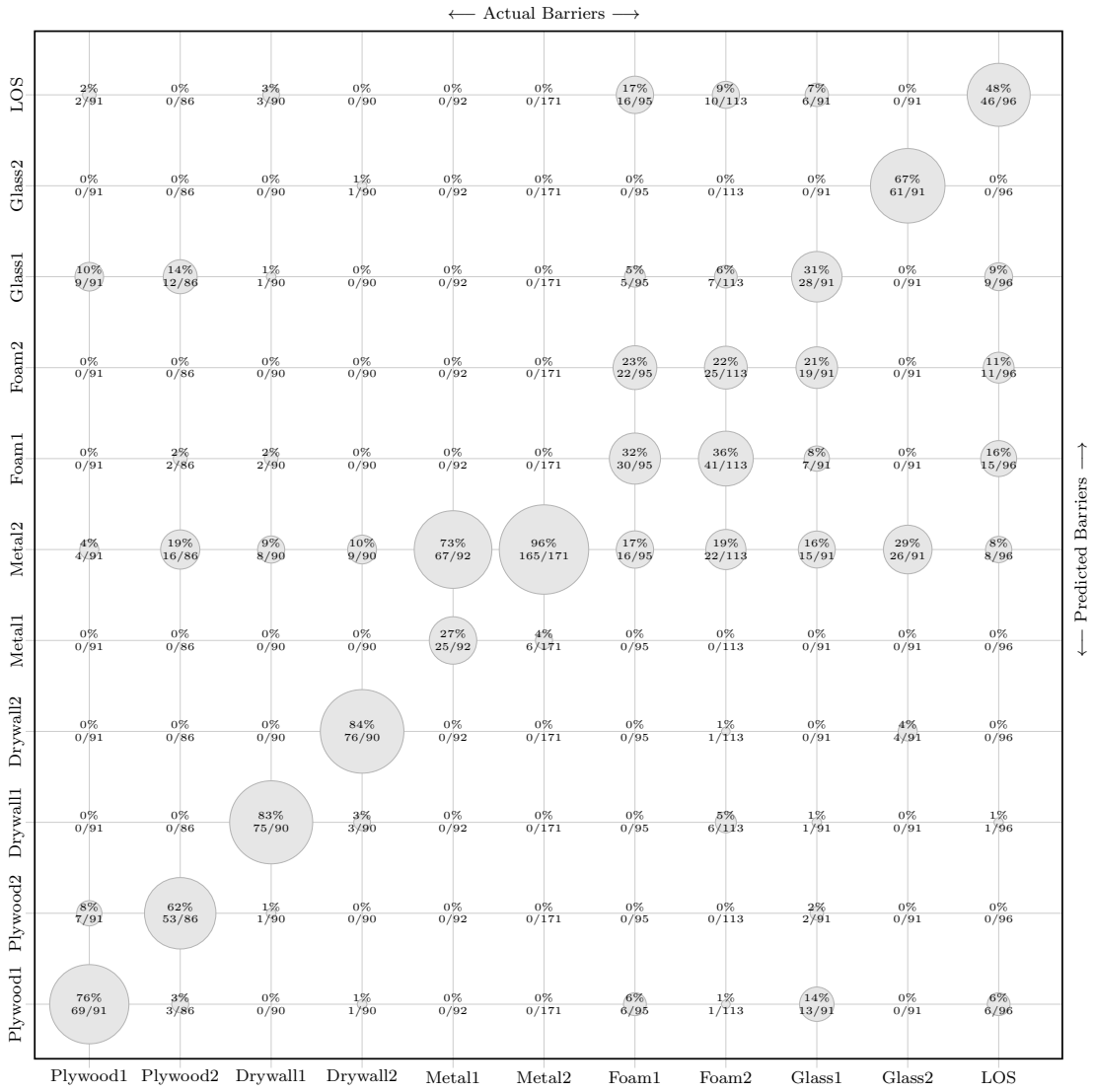
1. The algorithm is clearly not very good at differentiating thickness, particularly when it comes to metals and foams. Classifying barriers without regard to their thickness improves the success rate– an example is given in Appendix A

2. The vast bulk of the misclassified data forms a row– between 4 and 30 percent of all samples get misidentified as steel. In Figure 9, there is a similar row of error, but in this case it is identified as plywood. The mysterious row is caused by a significant number of samples, distributed surprisingly evenly between barriers, whose waveform metrics ('features', in machine-learning jargon) are unlike their brethren but quite like each other. This phenomenon can probably be attributed to 'junk' waveforms observed during data collection. It is believed that these are the result of a synchronization glitch between the radios.

3. When adjusting the RBF's γ parameter, it becomes immediately apparent that any misclassifications have less to do with physical similarity between barriers and more to do with the internals of the support vector machine. In order to understand the root cause of the misclassifications, it is necessary to look at the 6-dimensional space of features and the set of 'decision boundary' hyperplanes the SVM constructs.

Figure 11: Constant location

Constant location = 4.2 [59 percent success]



5.3 Feature space analysis

The SVM algorithm operates by calculating a set of hyperplanes, called 'decision boundaries'. They partition the vector space of input features. They are optimized so that (almost) all of the training data for each classification is on one side. The SVM then classifies testing data based on where it lies with respect to the decision boundary. In order to force the hyperplane to curve around a cluster of data points, the vector space undergoes a nonlinear transformation into a higher dimensional space.

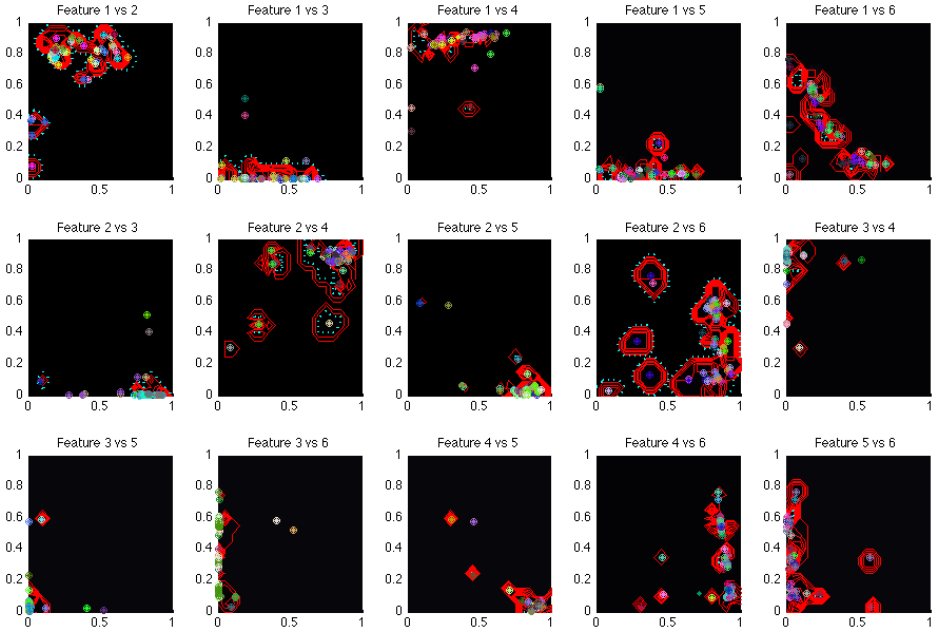
Figure 12 shows the internal state of the SVM after being trained on the entire dataset. Since six dimensions are hard to visualize, the space has been sliced to reveal every combination of 2D feature space. The red lines denote the decision boundaries, and the multicolored points denote the training data points. Points corresponding to a particular barrier are all given the same color. Several observations are immediate:

1. The classifications are not contiguous– the points corresponding to each barrier appear about as likely to be scattered as to be clustered together. The data does not fall into nice regions with this particular feature space.

2. Many unrelated points cluster together in the corners, making the decision boundary certainly complicated and very probably ineffective.

3. Each barrier only has a few points. Perhaps eleven classifications is too many when the SVM can only accept 200 training points?

Figure 12: Every 2D slice of the 6D feature space



Consider the histograms in Figures 13– 15(b). Each shows the distribution of all of the features, with colors corresponding to different barrier classes.

Figure 13: Stacked histogram for all data

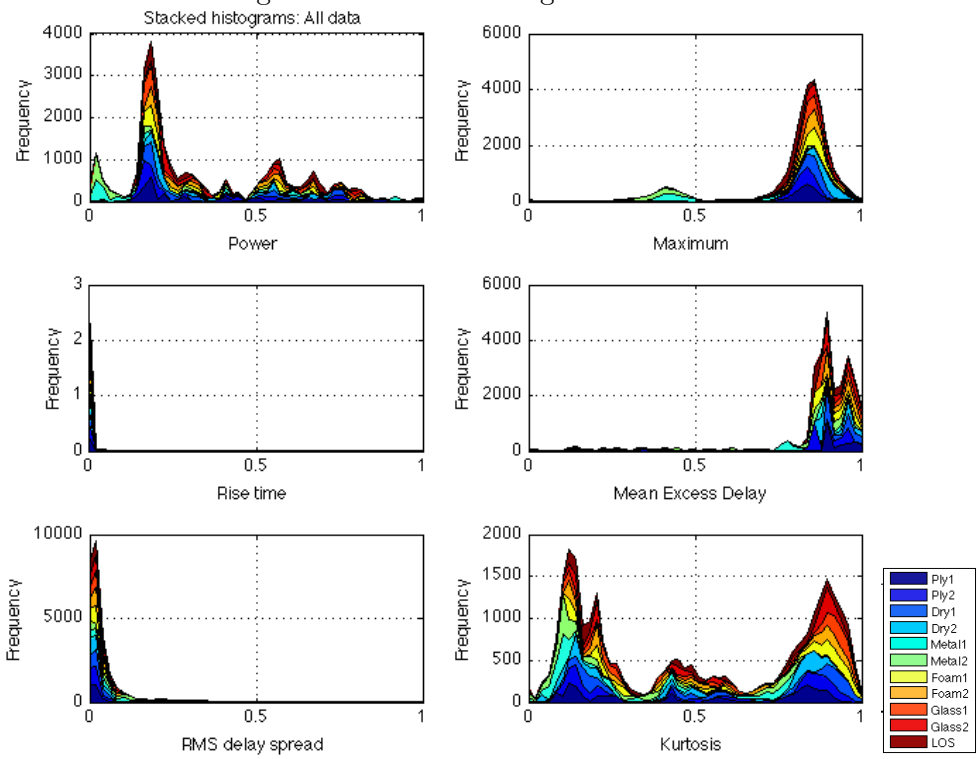


Figure 13 shows the distribution of the entire dataset with coloring to indicate the corresponding barriers. Some patterns emerge:

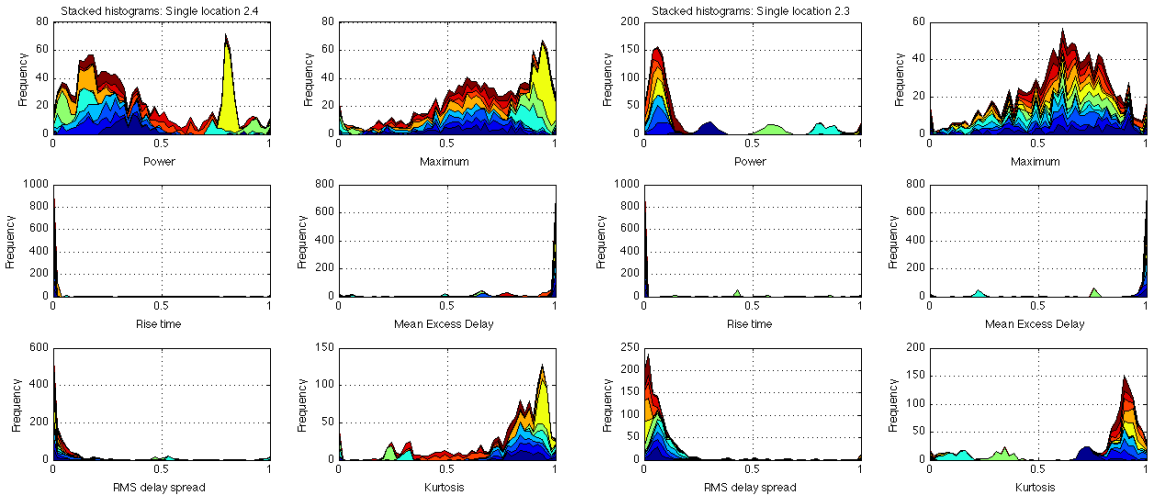
1. The rise time metric is distributed almost entirely at zero, making its contribution practically null. The RMS delay spread metric is slightly better, but still suboptimal.

2. The power and mean excess delay metrics have a number of peaks, but most of the barriers appear to be evenly distributed at each peak. These metrics appear to be much more sensitive to multipath than to the barrier material. However, there is a glaring exception– sheet steel, which is overwhelmingly concentrated in its own region. These features must be responsible for the SVM's noticeably better performance at identifying steel, compared to other barriers.

3. The 'maximum' metric appears to be insensitive to all barriers except steel, as we witness a very clean Gaussian distribution for each barrier centered at 0.8. Steel, on the other hand, is distinctly centered at 0.45. At first glance, this feature does not appear to be sensitive to steel thickness, or to multipath.

4. Kurtosis experiences a number of peaks, but all barriers are evenly distributed between them.

Figure 14: Stacked histograms for two locations



(a) Location 2.4 (b) Location 2.3

When we look at the distributions of all samples taken at a constant location, the picture changes. The multipath is ideally constant at each location, but between these two locations it varies wildly.

1. Although the two samples were taken mere feet from each other, they exhibit markedly different power distributions. In the first case, the power distributions for each barrier are roughly contiguous, but experience multiple peaks. In the second case, plywood and steel are not only contiguous but also well separated, while all other barriers hover at around 0.1.

2. The multipath seems to have caused the power distribution for foam to shift from about 0.8 to 0.95.

3. Rise time, RMS delay spread, and mean excess delay continue to do a poor job. However, with the smaller sample size, the outliers are now visible. It remains to be seen whether these outliers can be treated as noise and averaged away or removed entirely, or if they are significant. If they are significant, a possible technique to increase the usefulness of these features is to scale them logarithmically.

4. Kurtosis becomes visibly sensitive to steel, as long as the multipath stays quasi-constant. However, multipath does cause the distributions of steel and LOS to shift.

The analysis in feature space reveals that the features chosen to characterize the waveform have a limited retention of information about the barrier. Two problems arise: Firstly, several features

are too concentrated at their maximum or minimum to provide much information at all. Secondly, others can characterize the barrier nicely, e.g. power, but the information gets indeterminably skewed by the multipath. The only barrier which is visibly distinguishable from the others in feature space is steel. Considering its electrical properties, this is hardly surprising.

6 Conclusion

6.1 Summary of Findings

The experiment showed that machine learning can be used to identify a material barrier from a UWB pulse response, but that this ability breaks down unless steps are taken to mitigate multipath. The machine learning software functioned exactly as intended, and achieved a level of success in spite of the suboptimal set of features it was input. The features themselves proved to be the fundamental problem, as those that retained reasonable amounts of information about the barrier were also highly susceptible to interference from multipath. The only recurring exception was steel, which could be with 98% success even in the most general case.

6.2 Assessment of Hypothesis

The hypothesis was demonstrated to be **invalid**– 60% was predicted, 31% was achieved.

6.3 Future Work

Our research provided the basis for a wide variety of follow-on research in short term that can improve the accuracy of material barrier characterization and ultimately for lead to alternative applications of UWB technology. As direct follow ups, we can improve the results of success rate by taking a deeper look in the feature space to improve the accuracy of machine learning, account for echo within our surroundings and expand our materials to cover hybrid materials. In the long term, we can eventually move from predicting barriers to actively compensation and even potentially, adapting UWB radio for material sensing .

There exists many potential points of improvement in our machine learning assisted characterization. The success rates of characterization can be enhanced through examining the feature space. As observed from our analysis section, some of our current features are not utilized efficiently, replacing them or better management of the outliers should yield significant improvement. Additionally, we can be more sensitive to echo within our environment, given it can present an intermediate effect to the channel propagation readings. Potentially, we can reduce the potential error by better integrating the control LOS data into our calculations to normalize against echo. Finally, we can also introduce hybrid materials barriers, which makes the experiment more realistic since it's extremely unlikely for buildings to be made up on one primary material.

In regards to long term impact, the ability to accurately predicting material property can not only allow for compensation in NLOS distance finding but also for UWB to be potentially used as a sensor to gather information of its surroundings. Within UWB localization, the next logical step is to use material data to compensation for the deviations that caused NLOS. This will greatly improve the accuracy of UWB readings for determining locations. Beyond location finding, UWB radios can potentially be used for material sensing, such as using the channel propagation data to determine the material of its surroundings. Given UWB's broad frequency coverage, it can provide immense amounts of information as it operates at a much higher and broader bandwidth than our biological eyes.

Although we had to reject our original hypothesis, our failure illustrated many areas of possible future research that can drastically improve future success rate. Our research suggested there is much promise in using machine learning and UWB channel propagation for material analysis. It demonstrated the importance of environment variables in terms of machine learning and also showed that material thickness is not the best predictor. More importantly, we realize there is potentially large amount of information in the feature analysis space.

References

[1] M. Z. Win and R. A. Scholtz, "Impulse radio: How it works," *IEEE Commun. Lett.*, vol. 2, no. 2, pp. 36–38, Feb. 1998.

[2] ——, "On the energy capture of ultra-wide bandwidth signals in dense multipath environments," *IEEE Commun. Lett.*, vol. 2, no. 9, pp. 245–247, Sep. 1998.

[3] ——, "On the robustness of ultra-wide bandwidth signals in dense multipath environments," *IEEE Commun. Lett.*, pp. 51–54, February 1998.

[4] ——, "Characterization of ultra-wide bandwidth wireless indoor communications channel: A communication theoretic view," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 9, pp. 1613–1627, Dec. 2002.

[5] D. Cassioli, M. Z. Win, and A. F. Molisch, "The ultra-wide bandwidth indoor channel: from statistical model to simulations," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 6, pp. 1247–1257, Aug. 2002.

[6] A. Molisch, J. Foerster, and M. Pendergrass, "Channel models for ultrawideband personal area networks," *IEEE wireless communications*, vol. 10, no. 6, pp. 14–21, 2003.

[7] S. Gezici, Z. Tian, G. B. Biannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra- wideband radios," *IEEE Signal Process. Mag.*, 2005.

[8] J. Khodjaev, Y. Park, and A. S. Malik, "Survey of NLOS identification and error mitigation problems in uwb-based positioning algorithms for dense environments," *Ann. Telecommun.*, vol. 11, January 2009.

[9] J. Schroeder, S. Galler, K. Kyamakya, and K. Jobmann, "NLOS detection algorithms for ultra-wideband localization," in *4th Workshop on Positioning, Navigation, and Communication*, Hannover, August 2007, p. 8.

[10] L. Mucchi and P. Marcocci, "A new UWB indoor channel identification method," in *Cognitive Radio Oriented Wireless Networks and Communications, 2007. CrownCom 2007. 2nd International Conference on*, 1-3 2007, pp. 58 –62.

[11] S. Maranò, W. M. Gifford, H. Wymeersch, and M. Z. Win, "NLOS identification and mitigation for localization based on UWB experimental data," *IEEE J. Sel. Areas Commun.*, 2010, accepted pending revision.

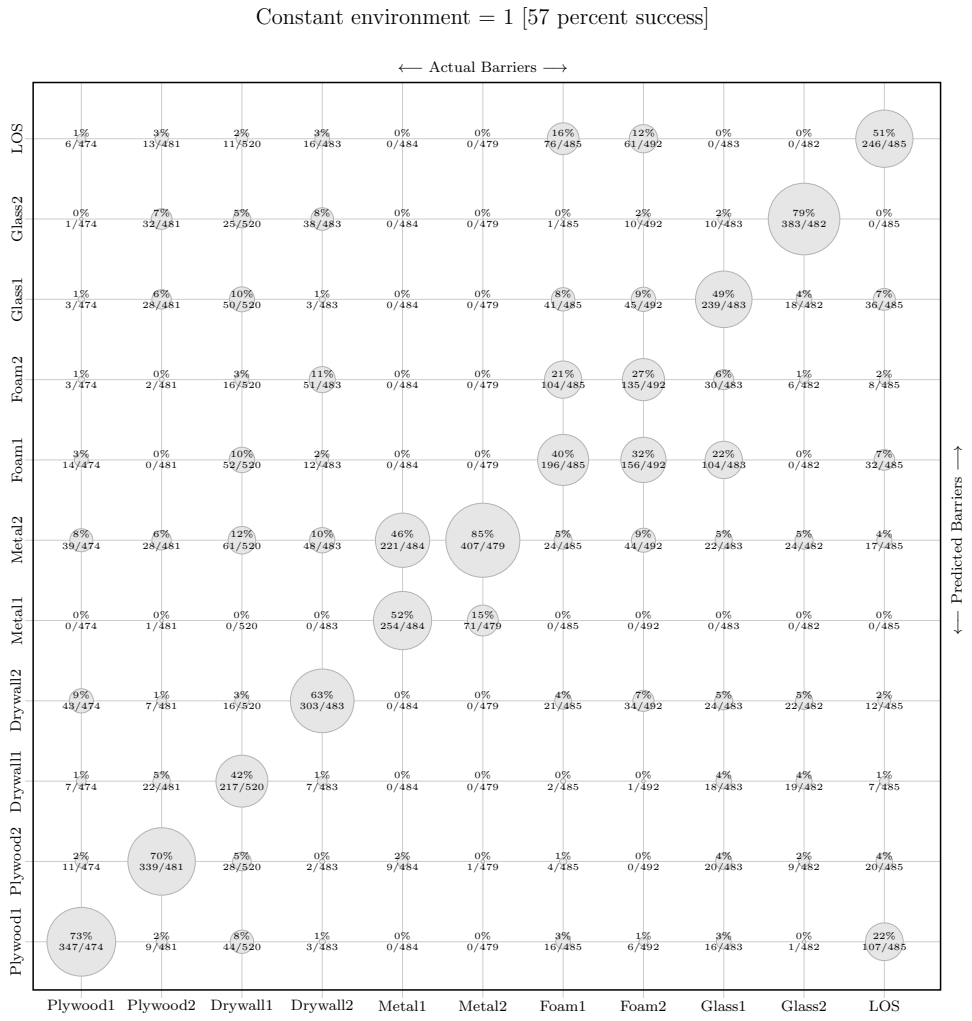
[12] ——, "Nonparametric obstruction detection for UWB localization," in *Proc. IEEE Global Telecomm. Conf.*, Honolulu, HI, Dec. 2009.

7 Acknowledgements

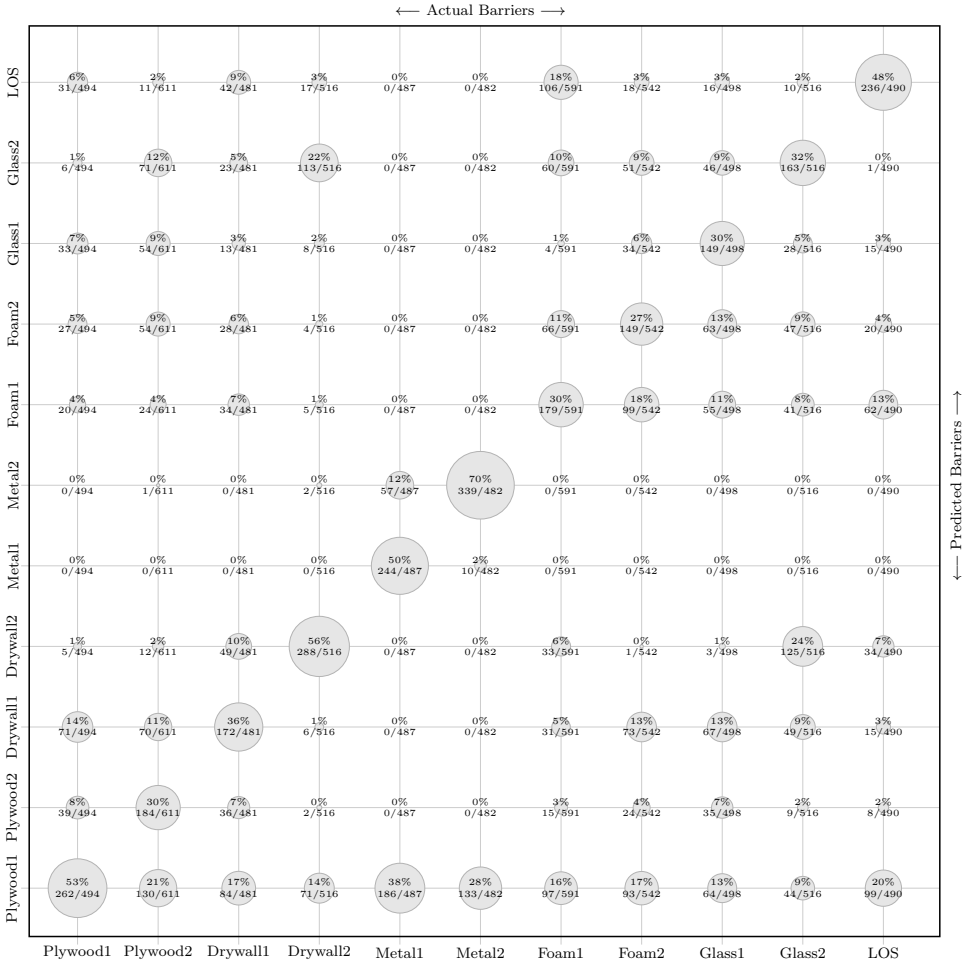
Thanks to Professor Moe Win and Santiago Mazuelas for your mentorship, your technical expertise and constant availability. Thanks to Jennifer Craig for your support and writing advice. Thanks to Todd and Dave for all the machining help and the last minute cart change. Thanks to the rest of the 16.62x support for all the help!

Appendices

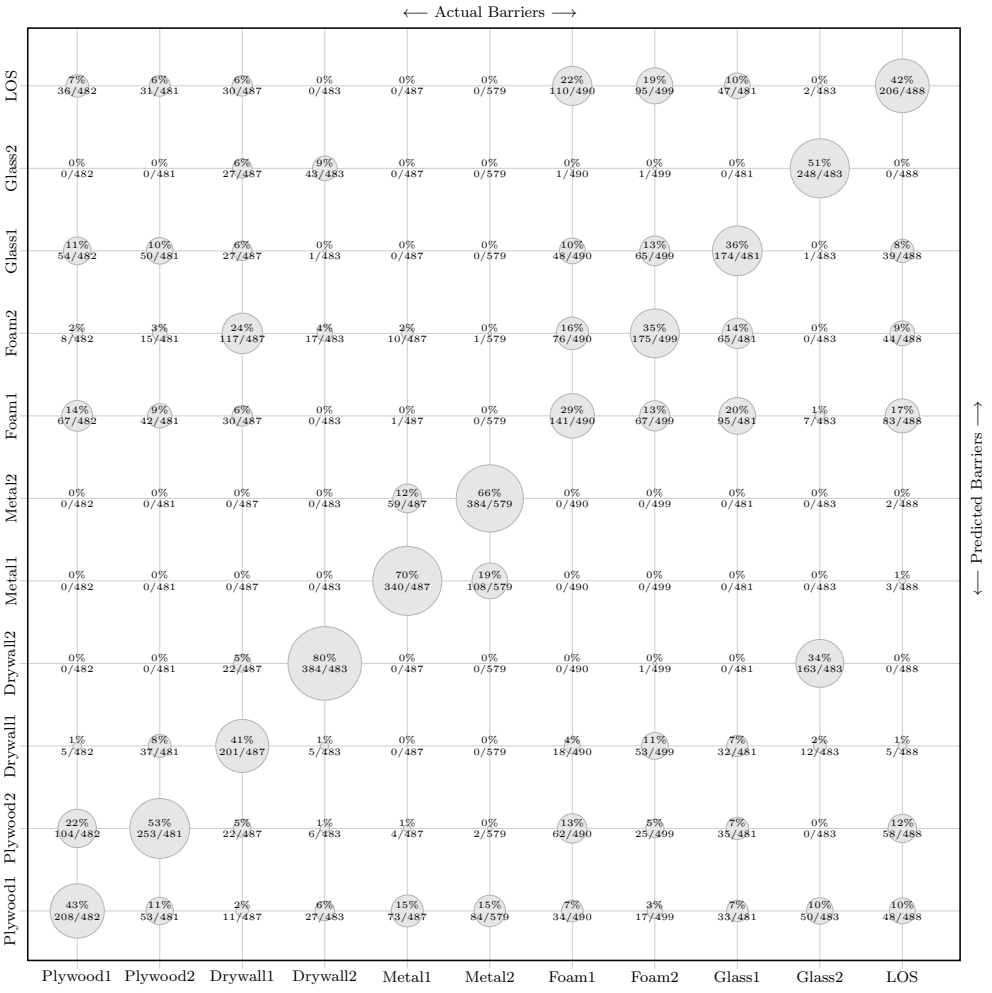
A Data Analysis– bubble diagrams



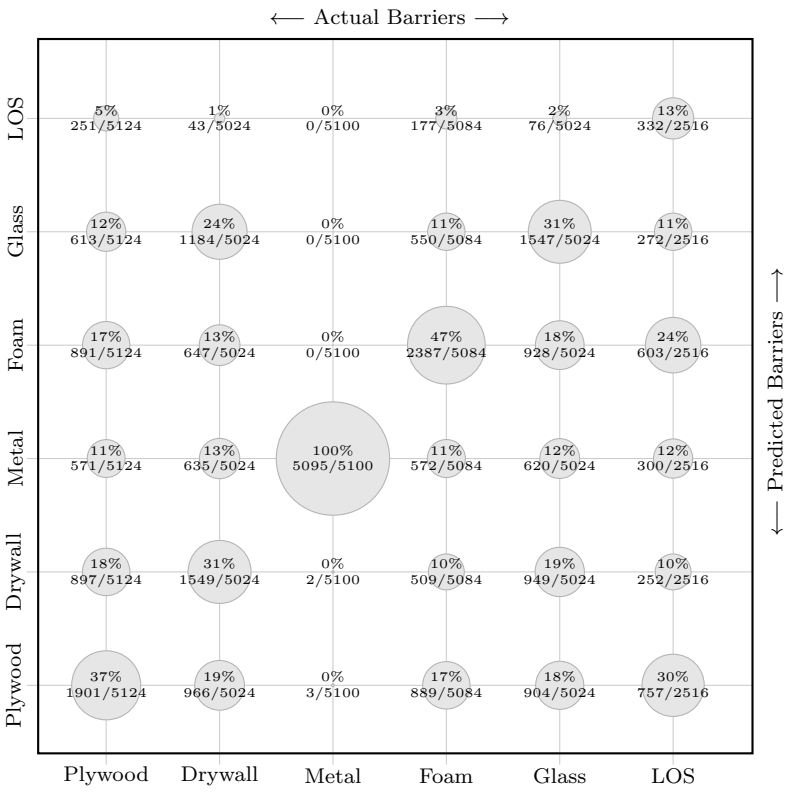
Constant environment = 2 [41 percent success]



Constant environment = 4 [50 percent success]



All data, barrier thickness agnostic. [46 percent success]



B MATLAB scripts

B.1 Data acquisition

B.1.1 Retrieve function

```
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % % % % % % %
%
% Waveform retrieval function
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % % % % % % %
%
% function waveform = retrieve(path)
%
% This function extracts a waveform [array of ints] from the text file
% that the PulsOn software generates.
% Inputs:    path = string giving location of waveform file
% Outputs:   waveform = array of ints

function waveform = retrieve(path)

data = 0;
num = 0;
counter = 0;

% Open file
fid = fopen(path);

% Create matrix. PulsOn will probably give us 1472 data points.
waveform = zeros(1472,1);
num = fgetl(fid);

while num ≠ -1

    % If we are reading the data, add the data to the output matrix
    if data
        counter = counter+1;
        waveform(counter,1) = str2num(num);

    % Ignore header info for now. Eventually put in a struct if needed.
    elseif (strcmp(num,'Data'))
        data = 1;
```

```
43      end
44
45      % Read next line
46      num = fgetl(fid);
47
48  end
49
50  fclose(fid);
51  %disp(sprintf('%s: Successfully read %d data points.',path,counter));
52  %plot(waveform);
53
54  end
```

B.1.2 Processing

```
% % % % % % % % % % % % % % % % % % % % % % % % %
%
% Waveform preprocessing function
% by Nathan Brei
% n_brei@mit.edu
%
% Adapted from extractFeatures.m, courtesy of Wesley Gifford
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % % % % % % % % % % % % % % % % %

% This function transforms a single waveform into a vector of metrics.
% Input:    waveform = array of floats as returned by retrieveWaveform
% Output:   metrics = [power, r_max, t_rise, t_med, t_rms, kurt]

function metrics = preprocess(waveform)

%% Time vector
time = linspace(0, 1, length(waveform));
% We are assuming (dangerously) that the waveform array captures 1s of
% radio activity.  Will this mess up the RMS-DS metric? Ask Santi/Wesley.

%% Energy metric
power = sum(abs(waveform).^2);

%% Max metric
r_max = max(abs(waveform));

%% Risetime metric
gamma_L = 6 * std(waveform(1:500));
gamma_H = 0.6 * r_max;

%ensure gamma_L > gamma_H
while (gamma_L > gamma_H)
    gamma_L = gamma_L/2;
end;

tStart = time( find(abs(waveform)≥gamma_L,1) );
tStop = time( find(abs(waveform)≥gamma_H,1) );

%t_rise = max((tStop-tStart), 1e-10);
% This doesn't make sense. tStop and tStart are scalars, considering we
% only took the _first_ index that satisfies abs(waveform)≥gamma_L
% Is this an artifact from older code? Instead we use:
t_rise = tStop - tStart;

%% Mean excess delay metric
```

```
51 t_med = sum(time' .* (abs(waveform).^2) ) / power;
52
53 %% RMS delay spread metric
54 t_rms =   sum( ((time' - t_med ).^2) .* (abs(waveform).^2) ) / power;
55
56 %% Kurtosis metric
57 kurt = kurtosis(abs(waveform));
58
59 %% Return
60 metrics = [power, r_max, t_rise, t_med, t_rms, kurt];
```

B.1.3 Bulk processing

```
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % %
%
% Bulk preprocessing script
% by Nathan Brei
% n_brei@mit.edu
%
% This script makes sure that all metrics have been calculated. If not, it
% calculates them. Data should exist in memory, as capture() and select()
% would leave it.nubivagant vicambulate tristificial vanmost woundikins schismarch
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % %

datadir = '/Users/nbrei/Desktop/62x/Data2/';

% Initialize data array
alldata(27500,1) = struct('env',[], 'loc',[],'bar',[], 'sam',[], 'met',[]);

disp 'Crunching... this might take a while...'

list=dir([datadir 'sample_*.txt'])
for i=1:length(list)
    name = list(i).name
    [env,loc,bar,sam] = strread(list(i).name, 'sample_%d_%d_%d_%d.txt');

    alldata(i).env = env;
    alldata(i).loc = loc;
    alldata(i).bar = bar;
    alldata(i).sam = sam;
    alldata(i).met = process(retrieve([datadir list(i).name]));
end

% Save results
save([datadir 'metrics-' date], 'alldata');
```

B.1.4 Organize

```
% % % % % % % % % % % % % % % % % % % % % % % % %
% Organizes collected waveforms for bulkpreprocess
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
% % % % % % % % % % % % % % % % % % % % % % % % %

function sample = organize(env, loc, bar, mask, capturedir, savedir)

    if (nargin<5)
        capturedir = '/Volumes/TravelDrive/data/';
        savedir = '/Users/nbrei/Desktop/62x/Data/';
    end

    % Do we have a capture directory? If not, yell at user
    if (exist(capturedir,'dir') ≠ 7)
        error([capturedir ' does not exist! Make it exist.']);
    end
    % Do we have a save directory? If not, yell at user
    if (exist(savedir,'dir') ≠ 7)
        error([savedir ' does not exist!']);
    end

    list=dir([capturedir mask]);
    for sample=1:length(list)

        inputfile = [capturedir list(sample).name];
        outputfilename = ['sample_'...
        num2str(env) '_'...
        num2str(loc) '_'...
        num2str(bar) '_'...
        num2str(sample) '.txt'];
        outputfile = [savedir outputfilename];

        % Copy raw waveform file to hard disk
        % Warning-- this will take ¬600MB
        copyfile(inputfile, outputfile);
    end
end
```

B.1.5 Bulk organize

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Waveform file management
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% This script captures and organizes large numbers of waveforms
% retrieved from a UWB radio via the PulsOn software. Before running:

% Move all *range files out of there
% cd capturedir
% !mkdir ranges
% !mv *range.txt ranges

% % % % % % % % % % % % % Knobs % % % % % % % % % % % % % % % % % % %
startenvironment = 5;               % Initial environment parameter
startlocation = 1;                  % Initial location parameter
minlocation = startlocation;
maxenvironment = 5; maxlocation=5; maxbarrier=11;
capturedir = '/Users/nbrei/Desktop/outside/';
savedir = '/Users/nbrei/Desktop/62x/Data2/';
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

close all;
clc;
disp ' '
disp '-----------------------------------------------------------------'
disp '                 16.62x Data Collection Time!'
disp '-----------------------------------------------------------------'
disp ' '

fprintf('startenvironment\t%d\n', startenvironment);
fprintf('startlocation\t\t%d\n', startlocation);
fprintf('capturedir\t\t%s\n', capturedir);
fprintf('savedir\t\t\t%s\n', savedir);
disp ' '
disp 'Press a key to continue'
disp ' '
pause

% Do we have a capture directory? If not, yell at user
if (exist(capturedir,'dir') ≠ 7)
    error([capturedir ' does not exist! Make it exist.']);
end

% Do we have a save directory? If not, yell at user
```

```
51 if (exist(savedir,'dir') ≠ 7)
52     error([savedir ' does not exist!']);
53 end
54 
55 % For each environment
56 for env = startenvironment:maxenvironment
57     for loc = minlocation:maxlocation
58         minlocation = 1;
59         for bar=1:maxbarrier
60 
61             filename = [int2str(env) '.' int2str(loc) '.' int2str(bar) '_*'];
62             organize(env, loc, bar, filename, capturedir, savedir)
63 
64         end % for each barrier
65     end % for each location
66 end % for each environment
```

B.2 Machine learning scripts

B.3 Key results, for multiple dataset slices

```
% % % % % % % % % % % % % % % % % % % % % % % % %
%
% Experiment script 1.1
% Visualizes raw results for multiple dataset slices
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % % % % % % % % % % % % % % % % %

clear all; close all; clc;
disp ' '
disp '1.1: Visualizing raw results for multiple dataset slices...'
disp '--------------------------------------------------------------'
disp ' '

% Load our complete data set
load /Users/nbrei/Desktop/62x/data-complete.mat
alldata=everything(1:28072);

% Dataset slices
clear selectFcn;
selectFcn{1} = @(sample) (1);
selectFcn{2} = @(sample) (sample.env==1);
selectFcn{3} = @(sample) (sample.env==2);
selectFcn{4} = @(sample) (sample.env==3);
selectFcn{5} = @(sample) (sample.env==4);
selectFcn{6} = @(sample) (sample.env==5);
selectFcn{7} = @(sample) (sample.env==1 && sample.loc==3);
selectFcn{8} = @(sample) (sample.env==1 && sample.loc==5);
selectFcn{9} = @(sample) (sample.env==4 && sample.loc==2);

% Dataset descriptions
titles{1} = 'All data';
titles{2} = 'Constant environment = 1';
titles{3} = 'Constant environment = 2';
titles{4} = 'Constant environment = 3';
titles{5} = 'Constant environment = 4';
titles{6} = 'Constant environment = 5';
titles{7} = 'Constant location = 1.3';
titles{8} = 'Constant location = 1.5';
titles{9} = 'Constant location = 4.2';

```

```
for i=1:length(titles)
    % Extract experiment set
    [X, Y, Z]=construct(select(alldata,selectFcn{i}));

    X = randintrlv(X,22);       % We shuffle the samples
    Y = randintrlv(Y,22);       % Seed -> Randintrlv shuffles in same order

    if length(X)<2000           % Number of training samples
        ts = floor(0.1*length(X));
    else
        ts = 200;
    end

    Xtrain = X(1:ts,:);         % Partition our dataset into training...
    Ytrain = Y(1:ts,1:11);
    Xtest = X(ts+1:end,:);      % ... and testing datasets.
    Ytest = Y(ts+1:end,1:11);

    % Train
    [tr a]=train(mc_svm(kernel('rbf',0.035)),data(Xtrain,Ytrain));

    % Test
    tst=test(a,data(Xtest,Ytest));

    % Tabulate results
    [successrate, resultstable] = results(tst);

    disp 'Generating plot...'
    % Generate plot
    fnamestr = ['1.1.' int2str(i) '.tex'];
    titlestr = [char(titles{i}) ' [' num2str(round(100*successrate)) ' percent success]'];
    axesstr = {'Plywood1','Plywood2','Drywall1','Drywall2','Metal1',...
    'Metal2','Foam1','Foam2','Glass1','Glass2','LOS'};
    visualize(resultstable, axesstr, fnamestr, titlestr, 1);

    disp(titlestr);
    resultstable
    disp '----------------------------------------------------------------'
end
```

B.3.1 Ignoring barrier thickness

```
% % % % % % % % % % % % % % % % % % % % % % % % % %
%
% Experiment script 1.2: Barrier thickness agnostic
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % % % % % % % % % % % % % % % % % %

clear all; close all; clc;
disp ' '
disp '1.2: Visualizing raw results, barrier thickness agnostic'
disp '---------------------------------------------------------'
disp ' '

% Load our complete data set
load /Users/nbrei/Desktop/62x/data-complete.mat
alldata=everything(1:28072);

% Merge different barrier thicknesses
alldata = merge(alldata, 2,1);
alldata = merge(alldata, 4,3);
alldata = merge(alldata, 6,5);
alldata = merge(alldata, 8,7);
alldata = merge(alldata, 10,9);

[X, Y, Z]=construct(alldata);
% Kill off empty classification columns
Y=Y(:,1:2:11);

X = randintrlv(X,22);       % We shuffle the samples
Y = randintrlv(Y,22);       % Seed -> Randintrlv shuffles in same order

Xtrain = X(1:200,:);         % Partition our dataset into training...
Ytrain = Y(1:200,:);
Xtest = X(201:end,:);        % ... and testing datasets.
Ytest = Y(201:end,:);

% Train
[tr a]=train(mc_svm(kernel('rbf',0.025)),data(Xtrain,Ytrain));

% Test
tst=test(a,data(Xtest,Ytest));

disp 'Done testing.'

% Tabulate results
```

```
51 [successrate, resultstable] = results(tst);
52
53 fnamestr = '1.2.tex';
54 titlestr = ['All data, barrier thickness agnostic. [' num2str(round(100*successrate)) ' percent success]'];
55 axesstr = {'Plywood','Drywall','Metal','Foam','Glass','LOS'};
56 visualize(resultstable, axesstr, fnamestr, titlestr, 1);
57
58 disp(titlestr);
59 resultstable
60 disp '----------------------------------------------------------------'
```

B.3.2 LOS vs NLOS

```
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % %
%
% Experiment script 1.3
% by Nathan Brei
% n_brei@mit.edu
%
% LOS vs NLOS identification from the entire dataset
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % %

clear all; close all; clc;
disp ' '
disp '1.3: Visualizing raw results, LOS vs NLOS identification'
disp '------------------------------------------------------'
disp ' '

% Load our complete data set
load /Users/nbrei/Desktop/62x/data-complete.mat
alldata=everything(1:28072);

% Extract experiment set
selectFcn = @(sample) (sample.bar==11);
[X1,Y1,Z1]=construct(select(alldata,selectFcn),11);
selectFcn = @(sample) (sample.bar≠11);
[X2,Y2,Z2]=construct(select(alldata,selectFcn),11);

X2 = randintrlv(X2,77);       % We shuffle the samples
Z2 = randintrlv(Z2,77);       % Seed -> Randintrlv shuffles in same order

X = [X1; X2(1:length(X1),:)];
Z = [Z1; Z2(1:length(Z1),:)];

X = randintrlv(X,22);       % We shuffle the samples
Z = randintrlv(Z,22);       % Seed -> Randintrlv shuffles in same order

Xtrain = X(1:200,:);         % Partition our dataset into training...
Ytrain = Z(1:200,:);
Xtest = X(201:end,:);        % ... and testing datasets.
Ytest = Z(201:end,:);

% Spider training method
[tr a]=train(svm,data(Xtrain,Ytrain));
tst=test(a,data(Xtest,Ytest));

% Tabulate results
[successrate, resultstable] = results(tst);
```

```
51
52 fnamestr = '1.3.tex';
53 titlestr = ['All data, LOS vs NLOS. [' num2str(round(100*successrate)) ' percent success]'];
54 axesstr = {'NLOS','LOS'};
55 visualize(resultstable, axesstr, fnamestr, titlestr);
56
57 disp(titlestr);
58 resultstable
59 disp '----------------------------------------------------------------'
```

B.4 Feature space

B.4.1 What the SVM actually sees

```
% % % % % % % % % % % % % % % % % % % % % % % % % %
%
% Experiment script 2.1: Slicing feature space
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % % % % % % % % % % % % % % % % % %

clear all; close all; clc;
disp ' '
disp ['2.1: 2D slices of feature space, with decision boundaries']
disp '------------------------------------------------------------'
disp ' '

% Load our complete data set
load /Users/nbrei/Desktop/62x/data-complete.mat
alldata=everything(1:28072);


% Extract experiment set
selectFcn = @(sample) (sample.env==1);
%selectFcn = @(sample) (1);
thisdata=select(alldata,selectFcn);
[X, Y,Z]=construct(thisdata);
Xbackup = X;

X = randintrlv(X,22);       % We shuffle the samples
Y = randintrlv(Y,22);       % Seed -> Randintrlv shuffles in same order

figure;

% Repeat for each combination of features
choices=nchoosek(1:6,2);
for k=1:15

    % Only use the two features from this time around
    Xtrim = [X(:,choices(k,1)) X(:,choices(k,2))];


    Xtrain = Xtrim(1:50,:);         % Partition our dataset into training...
    Ytrain = Y(1:50,1:11);      % Write another function to remove unused Y cols
    Xtest = Xtrim(50:end,:);        % ... and testing datasets.
```

```
47      Ytest = Y(50:end,1:11);
48
49      % Train
50      [tr a]=train(mc_svm(kernel('rbf',.035)),data(Xtrain,Ytrain));
51
52      % Test
53      tst=test(a,data(Xtest,Ytest));
54
55      % Tabulate results
56      [successrate, resultstable] = results(tst)
57
58      subplot(3,5,k);
59      plot(a);
60      title(['Feature ' int2str(choices(k,1)) ' vs ' int2str(choices(k,2))]);
61
62  end
63
64
65  featurespace(thisdata, Xbackup,500)
```

B.4.2 Stacked histograms

```
% % % % % % % % % % % % % % % % % % % % % % % % %
%
% Experiment script #28: Histograms of feature space
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % % % % % % % % % % % % % % % % %

disp ' '
disp '---------------------------------------------'
disp ' '
disp ' Experiment 28: Histograms of feature space'
disp ' '
disp '---------------------------------------------'
disp ' '

close all;
load /Users/nbrei/Desktop/62x/data-complete.mat
alldata=everything(1:28072);

% Dataset partitions
selectFcn{1} = @(sample) (1);
selectFcn{2} = @(sample) (sample.env==2);
selectFcn{3} = @(sample) (sample.env==2 && sample.loc == 1);
selectFcn{4} = @(sample) (sample.env==2 && sample.loc == 2);
selectFcn{5} = @(sample) (sample.env==2 && sample.loc == 3);
selectFcn{6} = @(sample) (sample.env==2 && sample.loc == 4);
selectFcn{7} = @(sample) (sample.bar<5);
selectFcn{8} = @(sample) (sample.bar==1 || sample.bar==11);

% Figure titles
titles = {'All data', 'Single environment', 'Single location 2.1','Single location 2.2','Single location 2.3','Si

for p=1:length(selectFcn)

    thisdata=select(alldata,selectFcn{p});
    Y = construct(thisdata);

    figure; set(gcf,'Color','w');

    for g=1:6
        subplot(230+g);
        featurehist(thisdata,g, Y);
    end
    legend({'Plywood 1','Plywood 2','Drywall 1','Drywall 2','Sheetmetal 1',...
             'Sheetmetal 2','Foam 1','Foam 2','Glass 1','Glass 2','LOS'});
```

```
51
52      subplot(232);
53      title(['Stacked histograms: ' char(titles{p})]);
54  end
```

B.5 Noise reduction

```
% % % % % % % % % % % % % % % % % % % % % % % % %
%
% Experiment script 3.1: Noise reduction on alldata, env1, loc2.2
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % % % % % % % % % % % % % % % % %

clear all; close all; clc;
disp ' '
disp '3.1: Noise reduction on entire dataset, one env and one loc'
disp '------------------------------------------------------------'
disp ' '

if (1)
    load noisereduced-06-Dec-2010.mat
    [X,Y,Z]=construct(outdata);
else
    % Load our complete data set
    load /Users/nbrei/Desktop/62x/data-complete.mat
    alldata=everything(1:28072);
    [X, Y, Z]=construct(removenoise(alldata,3));
end

X = randintrlv(X,22);       % We shuffle the samples
Y = randintrlv(Y,22);       % Seed -> Randintrlv shuffles in same order

Xtrain = X(1:200,:);         % Partition our dataset into training...
Ytrain = Y(1:200,:);        % Write another function to remove unused Y cols
Xtest = X(201:end,:);        % ... and testing datasets.
Ytest = Y(201:end,:);

% Train
[tr a]=train(mc_svm(kernel('rbf',0.55)),data(Xtrain,Ytrain));

% Test
tst=test(a,data(Xtest,Ytest));

% Tabulate results
[successrate, resultstable] = results(tst);

disp 'Generating plot...'
 % Generate plot
 fnamestr = ['3.1.1.tex'];
 titlestr = ['Entire dataset with noise reduction [' num2str(round(100*successrate)) ' percent success]'];
 axesstr = {'Plywood1','Plywood2','Drywall1','Drywall2','Metal1',...
```

```
51  'Metal2','Foam1','Foam2','Glass1','Glass2','LOS'};
52  visualize(resultstable, axesstr, fnamestr, titlestr, 1);
53
54  disp(titlestr);
55  resultstable
56  disp '----------------------------------------------------------------'
```

B.6 Tools

B.6.1 Visualization: MATLAB to LaTeX to TikZ

```
% % % % % % % % % %% % % % % % % % % % % % % % % % % % %
% Autogenerate a viewgraph and table depicting
% the results of an experiment with Spider.
% Generates TikZ inside LaTeX inside MATLAB. Fun!
%
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
% % % % % % % % % %% % % % % % % % % % % % % % % % % % %
%
% function visualize (results, labels, fname, titletext, silent)
%
% Inputs:  results    Matrix output of results.m
%          labels     Cell array of strings corresponding to SVM classes
%          fname      String for filename of LaTeX/TikZ source output
%          titletext  String for figure title
%          [silent]   Verbose pdflatex output? [Default=1]

function visualize (results, labels, fname, titletext, silent)
if(nargin==4), silent=1; end;

% get percentages
colsum = sum(results,1);
probs = results;
for n=1:size(results,2)
    probs(:,n) = probs(:,n) / colsum(n);
end

% Generate data string for LaTeX viewgraph, along with results table
fignum=['\\def \\nbclasses {' int2str(length(results)),'}\n'];
figdata='\\def \\nbviewgraphdata {';
for p=1:length(results)
    for q=1:length(results)
        percent = round(probs(p,q)*100);
        figdata = strcat(figdata, num2str(percent,3),'/',...
            num2str(sqrt(probs(p,q))),'/',int2str(q-1),'/',...
            int2str(p-1),'/',int2str(results(p,q)),'/', ...
            int2str(sum(results(:,q))),',');
    end
end
figdata = strcat(figdata(1:end-1), '}\n');


```

```
% Generate labels string for LaTeX viewgraph
figlabels='\\def \\nbviewgraphlabels {';
for m=1:length(labels)
    figlabels = strcat(figlabels, char(labels(m)), '/', int2str(m-1),',');
end
figlabels = strcat(figlabels(1:end-1), '}\n');

figtitle=['\\def \\titletext {' titletext '}\n'];

% LaTeX source for stub, so we can compile figure
stubtext={
'\\documentclass{article}'
'\\usepackage{tikz}'
'\\usepackage{fullpage}'
'\\usepackage{booktabs}'
'\\begin{document}'
'\\begin{center}'
'\\input{test.tex}'
'\\end{center}'
'\\end{document}'
};


% LaTeX source for viewgraph
figtext={ ...
'\\begin{tikzpicture}'
'   [scale=1.5, anno/.style={draw=none,fill=none,font=\\tiny},'
'   head/.style={draw=none,fill=none,font=\\footnotesize},'
'   title/.style={draw=none,fill=none,font=\\large},'
'   bubble/.style={draw=black!30, inner sep=0pt, fill=black!10, circle}]{'
'\\draw[step=1cm, black!20, thin] (-.7,-.7) grid (\\nbclasses-.3,\\nbclasses-.3);'
'\\draw[black,thick] (-.6,-.6) rectangle (\\nbclasses-.3,\\nbclasses-.3);'
''
'\\node[title] at (\\nbclasses/2-0.5, \\nbclasses+.5) {\\titletext};'
'\\node[head] at (\\nbclasses/2-0.5, \\nbclasses-.1) {$\\longleftarrow$\\ Actual Barriers\\ $\\longrightarrow$};'
'\\node[head, rotate=90] at (\\nbclasses-.1, \\nbclasses/2-0.5) {$\\longleftarrow$\\ Predicted Barriers\\ $\\long
''
'\\foreach \\bar / \\iter in \\nbviewgraphlabels {'
'   \\node[head] at (\\iter,-.8) {\\bar};'
'   \\node[head,rotate=90] at (-.8, \\iter) {\\bar}; }'
''
'\\foreach \\pct / \\rad / \\x / \\y / \\num / \\tot in \\nbviewgraphdata {'
'   \\node[bubble, minimum size=1.5*\\rad cm] at (\\x,\\y) {};'
'   \\node[anno] at (\\x,\\y+.075) {\\pct\\%%};'
'   \\node[anno] at (\\x,\\y-.075) {\\num/\\tot};}'
'};'
'\\end{tikzpicture}' ...
};

% Assemble LaTeX stub
if ¬exist('visuals','dir')
    mkdir visuals
end
```

```
100 fid=fopen(['visuals/stub' fname],'w');
101 stubtext(7)=cellstr(['\\input{' fname '}']);
102 for w=1:length(stubtext)
103     fprintf(fid,[char(stubtext(w)) '\n']);
104 end
105 
106 % Assemble viewgraph figure
107 fclose(fid);
108 fid = fopen(['visuals/' fname], 'w');
109 fprintf(fid,fignum);
110 fprintf(fid,figdata);
111 fprintf(fid,figlabels);
112 fprintf(fid,figtitle);
113 for w=1:length(figtext)
114     fprintf(fid,[char(figtext(w)) '\n']);
115 end
116 fclose(fid);
117 
118 % Compile and show
119 if silent
120     system(['cd visuals; /usr/texbin/pdflatex stub' fname ' >silent.txt']);
121     system(['open visuals/stub' fname(1:end-4) '.pdf']);
122 else
123     system(['cd visuals; /usr/texbin/pdflatex stub' fname]);
124     system(['open visuals/stub' fname(1:end-4) '.pdf']);
125 end
```

B.6.2 Select

```
% % % % % % % % % %% % % % % % % % % % % % % % % % % % %
% Script to select samples which satisfy a user-specified condition
%
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
% % % % % % % % % %% % % % % % % % % % % % % % % % % % %

% Inputs:  indata=       Array of struct containing waveform parameters and
%                       metric, should be created automatically by capture()
%          selectFcn=   Anonymous Boolean function that returns true if
%                       sample (which is an element of data) satisfies
%                       arbitrary conditions

function [outdata] = select(indata, selectFcn)

% Default/sample selectFcn
if (nargin<2)
    selectFcn = @(sample) (sample.env==1);
end

% Create output data array
outdata(27500,1) = struct('env',[], 'loc',[],'bar',[], 'sam',[], 'met',[]);
count=0;

% Run through each indata sample, add to outdata if it conforms to condition
for x=1:length(indata)
    if (selectFcn(indata(x)) & ¬(isnan(indata(x).met(6))))
        count = count+1;
        outdata(count) = indata(x);
    end
end

% Trim outdata
outdata=outdata(1:count,:);

fprintf( '%d samples selected.\n', count);
```

B.6.3 Results

```
% % % % % % % % % % % % % % % % % % % % % % % % %
%
% Results tabulator
% by Nathan Brei
% n_brei@mit.edu
%
% This script takes Spider's output object and turns it
% into something readable. Needs spider.
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % % % % % % % % % % % % % % % % %

function [successrate, resultstable] = results (testOutput)

X = testOutput.X;       % ACTUAL classifications... nah, this is fucked up
Y = testOutput.Y;       % PREDICTED classifications

% Give success rate
classloss = loss(testOutput,'class_loss');
successrate = 1-classloss.Y;

% Make compatible with single-class SVM results, too
if(size(X,2)==1)
    X=[X -X];
    Y=[Y -Y];
end

% Generate results matrix from Spider's output
resultstable = zeros(size(X,2));
for k=1:length(X)

    if(length(find(X(k,:)==1))>1)
        error(['ZOMG Spider FAILS at X(' int2str(k) ')']);
    end   % This little bit should satisfy Prof. Win

    x=find(X(k,:)==1,1);
    y=find(Y(k,:)==1,1);
    resultstable(x,y) = resultstable(x,y)+1;
end
```

B.6.4 Removenoise

```
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % %
% Function to remove noise by averaging samples
%
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % %

% Inputs:  indata=     Array of struct containing waveform parameters/metrics
%          n=          Number of samples per bucket
% Outputs: outdata=    Array just like indata, ready for construct.m

function [outdata] = removenoise(indata, n)
    if(nargin<2)
        n=5;
    end
    disp 'Removing noise...'
    count=1;
    for env=1:5
        for loc=1:5
            for bar=1:11
                disp([int2str(env) '.' int2str(loc) '.' int2str(bar)])
                selectFcn = @(sample)(sample.env==env&&sample.loc==loc && sample.bar==bar);

                samples = select(indata, selectFcn);
                for i=0:n:length(samples)-n
                    met = [0 0 0 0 0 0];
                    for j=1:n
                        met = met+samples(i+j).met;
                    end
                    outdata(count) = samples(1);
                    outdata(count).met = met/n;
                    count=count+1;
                end
            end
        end
    end

    save(['noisereduced-' date], 'outdata');
```

B.6.5 Merge

```
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % % % % % % %
% Function to merge two barrier types into a single class
% Note that you still need to manually remove the frombar column
% from construct.m's Y output
%
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
% % % % % % % % % %% % % % % % % % % % % % % % % % % % % % % % % % %

% Inputs:  indata=     Array of struct containing waveform parameters/metrics
%          frombar=    Barrier to be reassigned as
%          tobar
% Outputs: outdata=    Array just like indata, ready for construct.m

function [outdata] = merge(indata, frombar, tobar)

outdata = indata;
count = 0;

% Run through each sample, see if it conforms to specified condition
for i=1:length(outdata)
    if(outdata(i).bar==frombar)
        outdata(i).bar = tobar;
        count = count + 1;
    end
end

fprintf('%d samples merged.\n', count);
```

B.6.6 Feature histograms

```
1
2  % thisdata= array of struct as returned by select(), same order as...
3  % Y=        matrix of scaled metrics as returned by construct()
4  % n=        number of samples from each
5
6  function featurehist(thisdata,met,Y)
7
8      if(nargin<3)
9          Y=construct(thisdata);  % scale metrics
10     end
11
12     X={[],[],[],[],[],[],[],[],[],[],[]};
13     for i=1:length(thisdata)
14         X{thisdata(i).bar} = [X{thisdata(i).bar} Y(i,met)];
15     end
16
17     hold on; grid on; box on;
18     names = {'Power', 'Maximum', 'Rise time', ...
19     'Mean Excess Delay', 'RMS delay spread', 'Kurtosis'};
20     xlabel(names{met});
21     ylabel('Frequency');
22
23     for i=1:11
24         Z(:,i)=hist(X{i},50)';
25     end
26     area(linspace(0,1,50),Z);
27 %   alpha(0.25);
28 end
```

B.6.7 Construct

```
% % % % % % % % %% % % % % % % % % % % % % % % % % % %
% Script to convert a set of samples into a set of
% Spider classification matrices
%
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
% % % % % % % % %% % % % % % % % % % % % % % % % % % %

% Inputs:  data=       Array of struct containing waveform parameters/metrics
%          losbar=     Controls which bar value gets reported as 1 in Z
% Outputs: X=          Metrics array, digestible by spider
%          Y=          Multiclass labels array
%          Z=          LOSvsNLOS labels array

function [X, Y, Z] = construct(thisdata, losbar)

% Default losbar
if (nargin<2)
    losbar=11;
end

% Create (blank) spider input matrices
X=zeros(length(thisdata),6);
Y=zeros(length(thisdata),11);
Z=zeros(length(thisdata),1)-1;
barrierclass = 2*eye(11)-1;     % For generating Y

% Run through each sample, see if it conforms to specified condition
for i=1:length(thisdata)
    X(i,:) = thisdata(i).met;
    Y(i,:) = barrierclass(thisdata(i).bar,:);
    if(thisdata(i).bar==losbar)
        Z(i) = 1;
    end
end

% Scale all metrics between 0 and 1
metmax = max(X,[],1);
metmin = min(X,[],1);

for col=1:6
    X(:,col)=(X(:,col)-metmin(col))/(metmax(col)-metmin(col));
end

fprintf( '%d samples scaled.\n', length(X));
```

B.6.8 AutoRBF

```
% % % % % % % % % % % % % % % % % % % % % % % % %
%
% RBF optimization
% by Nathan Brei
% n_brei@mit.edu
%
% 16.62x Project
% Partner: Bo Han
% Advisor: Moe Win
%
% % % % % % % % % % % % % % % % % % % % % % % % %

function rbf=autorbf(X, Y)

disp ' '
disp '--------------------------------------------'
disp ' '
disp '    Finding RBF; this may take awhile...    '
disp ' '
disp '--------------------------------------------'
disp ' '

% What we used before:
%[tr a]=train(mc_svm(kernel('rbf',0.03)),data(Xtrain,Ytrain));
%tst=test(a,data(Xtest,Ytest));
%[successrate, resultstable] = results(tst);

d=data(X,Y);

s = mc_svm(kernel('rbf',0));
a1=param(s,'rbf',[0.59 0.58 0.57 0.56 0.55]);

a2=gridsel(a1);
% Gridsel will perform a n-fold for each object in a1 and choose the best performing according the cross-validati

[r,a]=train(a2,d);

loss(r)
a.best
r
```