Navigation Using VLF Environmental Features

Joseph Curro, John Raquet Air Force Institute of Technology

Biography

Capt Joseph Curro is an electrical engineering PhD student in the Department of Electrical Engineering at the Air Force Institute of Technology. He is performing research in the Autonomy and Navigation Technology (ANT) Center.

Dr. John Raquet is the Director of the Autonomy and Navigation Technology (ANT) Center at the Air Force Institute of Technology, where he is also an Associate Professor of Electrical Engineering. He has been involved in navigation-related research for over 25 years.

Abstract—This paper examines the use of Very Low Frequency (VLF) electromagnetic signals to perform location estimation without any knowledge of the source of the VLF signal or their interaction with the environment. Signals were collected from different locations in a variety of indoor environments. The signals are divided into equal time length segments and processed into feature sets. A logistic regression and an Artificial Neural Network are then both trained on the sample statistics to give a set of parameters to return likelihood of each locations with the correct location being determined as the position with the highest likelihood. This approach using either method achieved an error of less than 1% for 17 locations recorded for 10 minutes each.

Introduction

This paper focuses on using Very Low Frequency (VLF) signals to determine the location of the receive antenna. The VLF signals and transmitters are not known beforehand. Each recorded sample is labeled from the location it came from in order to characterize the location.

Most existing RF-based positioning estimates rely heavily on knowledge of the environment before navigation. For example the Global Positioning System (GPS) solution uses precise timing and known signals to compute a position solution anywhere on the earth. GPS was set up intentionally as a positioning system but other signals not intended for navigation can be used in a similar manner, an approach commonly related to is navigation using signals of opportunity. For example AM radio signals were used with a base station in time difference of arrival approach to determine position. In this instance the signal of the AM radio is unknown but the signal parameters such as frequency and transmit location are.

In this paper we propose the use of signals where the transmit location, frequency and signal shape are all unknown. This approach will be closer to a fingerprint approach that looks for the existence of signals. Biswas et. al used Wi-Fi signals from common wireless routers to navigate [3]. The Wi-Fi routers were identified by their Service Set Identifier (SSID) to uniquely identify routers. A signal strength measurement was made to give a rough idea of the distance to the known router. The location of each unique router was known and an estimate of position was derived using the set of routers identified and signal strength for each router. This was combined with other inertial sensors to develop a navigation solution as show by Vilaseca et. al in [9]. Faragher et. al [5] used the signal strength in specific bands such as VHF, cellular signals, and WiFi to discriminate locations when combined with inertial sensors. Discriminating by signal strength allows other bands other then WiFi to be included but only the signal strength can compare one band measurement to another.

In an approach to characterize different wireless routers Cobb et. al used statistics from each router to develop a unique fingerprint for each wireless device [4]. The statistics hone in on the imperfections during manufacture of the routers to uniquely identify a router even if the transmitted signal and SSID are the same. The device signals are pre-processed into statistical feature sets in order to compare signals. This allows signals that transmit the same logical information to be differentiated. Cobb et. al performed this in order to authenticate a known router but the process can also be used to identify and characterize an unknown router.

This paper focuses on using VLF signals to determine the location. This process combines the approach of navigation with router signal strengths in [3], [5] with the ability to uniquely identify a router in [4]. In this application the signal variations are caused by at different locations in-
stead of different devices. The big difference is that this approach now is agnostic of the transmit device and can use any signal, not just signals from wireless routers. Different locations are sampled beforehand to determine the characteristic features of the signals in that location, in effect generating a feature map. All of these location samples are then used to make an algorithm that can determine location based on the samples. Currently two different approaches, Logistic Regression and an Artificial Neural Network (ANN), are used to process the samples to make a function that can return likelihoods of each recorded location.

The rest of this paper is organized as follows. First a method of processing the recorded signals into feature sets is outlined. Next the Logistic Regression and Artificial Neural Network used to classify the feature sets into locations are described. Finally results from recorded signals are analyzed.

3 Algorithms

This section outlines the concepts to determine location from recorded VLF signals. First signals are recorded over a period of time at each location. The recorded signals are broken down into equal time samples and a feature set for each sample is calculated based on statistics of each sample. The samples are labeled by each location they were recorded at and used to train a Logistic Regression or an ANN which will discriminate differences between the locations based on their samples. When navigating a signal is recorded from an unknown location. The signal is broken into samples and processed into feature sets the same as the training samples were. The weights and biases determined in training the Logistic Regression or ANN are then applied to this feature set to give a likelihood of each of location. The location with the highest likelihood is declared the location of the signal.

3.1 Sample Statistics Generation

To determine the statistics for a recorded time the same process is done for any given sample as follows [7].

1. Determine the power spectral density of the recorded time and crop to desired time window and frequency band (Blue box in Figure 1).
2. Break each sample window into a grid of patches all with equal frequency band and time length (Green box in Figure 1).
3. Flatten the 2D patch into 1D and perform a batch of statistics on the patch.
4. Stack an array of all the statistics for each patch into a feature set for the given window.

This creates a feature set that can be used to train the logistic regression or ANN. These steps will now be described in more detail.

![Figure 1: Spectral density plot of a recorded signal. Blue box is a window that a sample will be taken from. Green box is a patch that statistics will be performed on to derive features for the sample window. A white hot heatmap is used where white is higher power and dark red is lower power.](image)

The power spectral density calculates the power in each frequency of the signal over time. This creates a 2D image with time on the horizontal axis and frequency on the vertical axis as shown in Figure 1. The value of each pixel in the image represents the power of that frequency at that time. The frequency content is more robust to changes in amplitude caused by antenna gain or propagation loss which make this a better starting point for a feature set compared to the raw amplitude measurements.

This power spectral density is done over the entire recorded time and with frequency band as large as the sampling of the signal allows. This arbitrary record time and frequency band are then partitioned into a window so that each final sample is from the same time period and frequency band. This window has a given time period $t$ and frequency band $f = (f_{low}, f_{high})$. Each of these windows are further divided into a grid of patches as shown in Figure 1. Each window has an equal number of patches determined by the partitions made in the time and frequency axis. The number of partitions made in the time axis is $N_T$ and in the frequency axis $N_F$. This now creates $N_{TF} = N_T N_F$ total patches for each window. For each patch a set of statistics is calculated on the flattened 2D matrix that represented the power spectral density in that patch window. A combination of statistics used can include mean, standard deviation, variance, skewness, and kurtosis. The total number of statistics chosen is $N_{Stat}$. Thus each patch has an array of
statistics.

\[ e_p = [s_1 \ldots s_{N_{Stat}}] \]

where \( e_p \) are the statistics for the given patch and \( s_n \) is a given calculated statistic for example \( s_1 \) could be variance and \( s_2 \) be kurtosis. After all the statistics are determined for each patch they are all concatenated together to make a feature array \( f_W \) of length \( N_{feat} = N_{Stat} N_{FF} \).

\[ f_W = [e_1 \ldots e_{N_{FF}}] \]

where \( f_W \) is the array of statistics for the given window \( W \). Each individual statistic is normalized over each window. For example all the variances from the \( e_p \) arrays are collected and normalized for a given \( f_W \). To generate more samples the window is moved over in time and new statistics are computed for the recorded signal. A feature array represents the given window and will be used to train an algorithm or get a likelihood from a pre-trained algorithm. All of the \( f_W \) arrays from all the windows of the recorded signal for all the locations are then stacked on top of each other to make the full feature set matrix \( F_W \) which is \( (N_S) \times (N_{feat}) \) where \( N_S \) is the number of sample windows.

\[ F_W = \begin{bmatrix} f_1 \\ \vdots \\ f_{N_S} \end{bmatrix} \]

An output matrix \( Y_W \) is also produced which is the output location of each of the samples in \( F_W \). For example if row \( i \) is location number \( y \) then the value of \( Y_W(i) = y \).

\[ Y_W = \begin{bmatrix} y_1 \\ \vdots \\ y_{N_S} \end{bmatrix} \]

### 3.2 Logistic Regression

The logistic regression used in this paper is a probabilistic, linear classifier as described by Bastien and Bergstra [1] [2]. It is parameterized with a weight matrix \( W \) and a bias vector \( b \). To characterize the output locations the input feature array \( f_W \) is multiplied by the weights and added to bias. This is then normalized with the softmax function to give a set of likelihoods as shown by.

\[ P(Y = i|f_W, W, b) = (\text{softmax}(Wf_W + b))(i) \quad (1) \]

where \( i \) is the index of the output location and \( f_W \) is the input feature array. The softmax function raises \( e \) to the power of each argument in the array \( Wf_W + b \) and then normalize each argument so the array sums to one. To determine \( W \) and \( b \) a cost function is minimized. The cost function is the negative log likelihood of the probability of the known location as shown by

\[ l(W, b) = -\sum_{i=0}^{N} \log(P(Y = i|f_W, W, b)) \quad (2) \]

where \( N \) is the number of locations in the output. This cost function is attempting to minimize the negative of the probabilities of the correct location for each output. This is the same as maximizing the probabilities of the correct location for each output. Making the cost function a probability causes the correct location compete with other locations because the probability must sum to one. This prevents the algorithm from being too greedy and increasing the weights to large numbers in order maximize a result.

This cost function is then minimized using a stochastic gradient descent approach, where the derivative of the cost function is taken with respect to each unknown in \( W \) and each unknown in \( b \). The derivative times a learning rate \( l_r \) is then used to update the values of \( W, b = (l_r)W, b \), and this process is repeated until a threshold of the magnitude of the change of number of steps is reached. The learning rate is chosen to allow the solution to converge more quickly or slowly compared to using the derivatives raw. LeCun et. al describe how to choose a learning rate in [6].

The stochastic gradient descent attempts to reach a minimum for the cost function which is the best fit of the model parameters given the sample data. However this minimum may not be the global minimum for the entire sample data space. Choosing appropriate values for the learning rate and initialization values for the weight and bias matrices can prevent the algorithm from becoming stuck in a local minimum.

### 3.3 Artificial Neural Network

The ANN uses the logistic regression describe above but with extra layers of processing as described by Bastien and Bergstra [1] [2]. The ANN is a modified logistic regression layer inserted on top of the original logistic regression. The ANN now receives the \( f_W \) vector and multiplies it by weights and adds a bias as the logistic regression as in section 3.2 but the modified layer performs an arc tangent function on the result instead of the softmax function. The arc tangent function acts like a switch driving the inputs high or low quickly.

\[ l_W = (\text{arctan}(Wf_W + b_l)) \]

where \( W_l, b_l \) are the weights and bias matrices for the ANN layer. The new vector \( l_W \) is now treated like the \( f_W \) vector from the logistic regression in order to develop likelihoods. The vector \( l_W \) also can have more or fewer entries then the input vector \( f_W \). The number of entries in the vector changes the weights the logistic regression will use in order to determine final likelihoods. The length of \( l_W \) is determined experimental with rules developed over time as explained by LeCun et. al in [6]. The final equation to determine the likelihoods with a two layer network is shown.
by

\[ P(Y = i|f_W, W_l, b_l) = \text{softmax}(W(l \arctan(W_l f_W + b_l)) + b_l(i)) \]

this probability now has many more degrees of freedom due to the added weights and biases from the ANN layer. This whole network is trained using a gradient descent as the logistic regression was but with more weights and biases. The new cost function from Equation 2 now has more degrees of freedom than the logistic regression was due to the added weights and biases from the ANN layer. This whole network is trained using a gradient descent as the logistic regression was but with more weights and biases. The new cost function from Equation 2 now has more degrees of freedom than the logistic regression was due to the added weights and biases from the ANN layer.

\[ I(W, b) = -\sum_{i=0}^{N} \log(P(Y = i|f_W, W_l, b_l)) + L_1 + L_2 \]

\[ L_1 = \sum(W_l) + \sum(W) \]

\[ L_2 = \sum(W_l^2) + \sum(W^2) \]

where \( L_1 \) is the sum of the entries of both weight matrixes and \( L_2 \) is the sum of the squares of both weight matrixes. The \( L_1 \) and \( L_2 \) add a penalty for the weights becoming too large, since the cost function is minimized with the stochastic gradient descent. This approach helps prevent over training to a specific set of samples which is more likely with the added degrees of freedom compared to the logistic regression. Each of the penalty parameters \( L_1 \) and \( L_2 \) are also multiplied by a scalars \( l_1 \) and \( l_2 \) which is tuned according to methods described in [6].

This method can be extended to multiple layers if \( W_l \) is used as the input to another layer with a new set of weights and biases. This adds even more degrees of freedom for the model to adapt to complicated phenomena. However, more degrees of freedom take longer to train and may not provide improved performance.

3.4 Training

Both the logistic regression and the ANN method require training in order to determine the weights and biases. Each method uses a stochastic gradient descent of a cost function in order to determine the next changes to be made to the weights and biases. This is also called back-propagation. The training is done with samples that are divided into three different sets called the training, validation and testing sets. This is done to remove any correlation in using the training samples to test. For example, the algorithm could actually just be memorizing each sample exactly and not determining a characteristic method to identify locations. The training set is used only to train the weights and biases of the model. The validation set is used to check the performance of the weights at set intervals during training. This allows a threshold for stopping to be not based on the training data. The training uses the improvement from one validation set to the next as a criteria for stopping. The final set is the test set and is used for final testing after the weights are determined. This final test shows how well the model can predict samples it has never seen before but are still within the known locations.

Training has two stopping conditions. One is an overall amount of change this step will make in updating the weights and biases. As the change becomes smaller the time required to make a change is more costly compared to the improvement it will add to the weights and biases. The other is an early stopping criteria based on the differences in validation sets. Validation sets are small batches of samples used to update the weights and biases. At the end of a batch the validation set is used to test performance. If the number of incorrect matches is lowered by a proportion of the best amount it is considered significant and the batches continue. If the improvement is not high enough the algorithm increases a patience counter. When the patience counter exceeds a threshold the algorithm runs out of patience and stops doing any more batches, because they are not reducing the number of incorrect matches. The training also has a minimum amount of patience. This is the minimum number of updates to perform regardless of improvement.

3.5 Threshold

After training the weights, a rejection threshold can be made to lower the false positive rate of the model while introducing a false negative and a true negative rate. The threshold is the ratio of the largest likelihood to the second largest likelihood

\[ t_r = \arg\max_0(P(Y = i|f_W, W, b))/\arg\max_1(P(Y = i|f_W, W, b)) \]

Where \( \arg\max_0 \) returns the \( x^{th} \) largest likelihood. This threshold allows the algorithm to be conservative and return positions only when the position is known with much higher certainty relative to the other positions. This returns fewer false positives than strictly returning the largest likelihood.

3.6 Setup Decisions

A loop antenna is used to record the VLF signals. Loop antennas have properties that are favorable for VLF signals. One property of a loop antenna is the magnetic fields have high gain compared to electric fields. This coupled with magnetic fields dissipating quickly through air over distance make the magnetic VLF signals a loop antenna receives limited to devices in the local area. That is magnetic fields with a source near the loop antenna are received much better then magnetic fields far away from the antenna. This property allows locations that would be considered very close in terms of the wave lengths involved to have very different signals. This is also useful because the low frequencies like VLF penetrate most ordinary objects.
This means that moving the antenna will have little effect on changing the types of signals that are high power and travel long distances. However, the local signals from devices near the antenna dominate while high power signals from far away sources have much lower gain. This will create location variations relative to the distance of local transmitters.

Loop antennas are also omnidirectional in the x-y plane which allows for the receive signals to be the same, regardless of the heading the antenna faces when the loop antenna is flat. Thus, pedestrian navigation which is mostly about knowing a 2D location will be unaffected by heading changes. These reasons made a loop antenna the preferred receive antenna for the VLF signals in this paper.

4 Results

This section analyzes the results from recording data from multiple locations inside one building and training a logistic regression and ANN on the sampled locations. The error in determining locations and possible sources of error are included at the end of the section.

4.1 Test Setup

A number of locations were recorded in the Air Force Institute of Technology (AFIT) in Dayton Ohio. The locations inside the building were located all over the building and were never in the same room twice. Classrooms, lobbies, and offices make up the bulk of locations. The classrooms and offices contained at least one computer as a potential transmitter while the lobbies were missing any obvious transmitter.

4.1.1 Loop Antenna

The device for our loop antenna was designed by Dr. James Plesa [8]. A loop antenna with 41 windings and diameter of 31.75cm was used to record the VLF signals. The loop antenna was attached to the microphone input for a 3.5mm Tip Ring Ring Sleeve (TRRS) jack for recording by an Apple iPhone. A resistor pot was put in series with the loop antenna to achieve the minimum resistance required for the iPhone to recognize a microphone input. The iPhone recorded the loop antenna and saved it as a .wav file for later processing.

4.2 Sample parameters

Samples were taken from 17 different locations within AFIT. Each location was recorded for at least 10 minutes. Locations were recorded over a two month window returning to certain spots periodically to observed changes in time. The sample window time of $w_t = 1$ second and frequency band $w_b = (0, 6000)$ Hz was used to make at least 600 samples for each location (collecting over 600 seconds). Any location with more samples then 600 was reduced back to 600 by randomly picking samples from the total samples available for that location. The power spectral density used a 10 point Fast Fourier Transform (FFT) to compute the samples. The sample windows were divided into $N_T = 15$ time sections and $N_F = 50$ frequency sections. The number of statistics calculated for each patch was $N_{Stat} = 4$ with mean, variance, skewness and kurtosis being the four used. In order to train the weights without overtraining the total samples were divided into 3 different sections. The training set, the validation set and the testing set which were $5/7$ $1/7$ and $1/7$ of the total samples respectively.

4.2.1 Logistic Regression Results

Using the above parameters the weights were trained with the samples for all the locations. The logistic regression had a learning rate of $l_r = .13$ and a max number of validations steps of 1000. Validation batches included 600 samples from the training set. A validation improvement of 99.5% was considered significant. So, if at least one of 100 incorrect samples was made correct by an update, the algorithm continues. The regression had a patience of two validation updates, meaning after failing to improve it will continue with two more validation steps regardless. The regression would also have patience for at least 5000 updates.

The final testing performance results are summarize in Table 1. The final testing performance had an error rate of under 1% as shown in table 1. That means one percent of all
Table 1: Logistic Regression Results

<table>
<thead>
<tr>
<th></th>
<th>true class</th>
<th>false class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy without threshold</td>
<td>0.992</td>
<td>0.008</td>
</tr>
<tr>
<td>Accuracy with threshold</td>
<td></td>
<td></td>
</tr>
<tr>
<td>above threshold</td>
<td>0.979</td>
<td>0.002</td>
</tr>
<tr>
<td>below threshold</td>
<td>0.013</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Table 2: Artificial Neural Network Results

<table>
<thead>
<tr>
<th></th>
<th>true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy without threshold</td>
<td>0.998</td>
<td>0.002</td>
</tr>
<tr>
<td>Accuracy with threshold</td>
<td></td>
<td></td>
</tr>
<tr>
<td>above threshold</td>
<td>0.996</td>
<td>0.001</td>
</tr>
<tr>
<td>below threshold</td>
<td>0.002</td>
<td>0.001</td>
</tr>
</tbody>
</table>

the sampled test locations gave an incorrect location. This one percent is the false positive rate without implementing a rejection threshold. An equal error rate threshold was implemented for each output class where below the threshold the rate of true positives and false positives is equal. With the threshold the false positive rate was reduced to 0.2%. This created a false negative rate of 1.3% and a true negative rate of 0.6%. These results are summarized in Table 1.

4.3 Artificial Neural Network results

An ANN was trained with the same sample parameters as the logistic regression. The same data set from the logistic regression was used with all the samples being equal and the training, validation and test sets containing the exact same samples. The ANN had the same validation improvement threshold of 99.5%, patience of two validation updates, and minimum patience of 5000 updates. The ANN had one layer with 500 nodes. The values of the penalty scalars were $l_1 = 0.00$ and $l_2 = 0.0001$. This removed $L_1$ as a penalty and reduced the overall magnitude of $L_2$. The ANN achieved similar results compared to the logistic regression. The final testing performance had an error rate of 0.2%. An equal error rate threshold was introduced as in the logistic regression to further reduce false positives. With the threshold the false positive rate was reduced to 0.1% while introducing a false negative rate of 0.2% and a true negative rate of 0.1%.

5 Conclusion

This paper presents two algorithms for determining location based on the VLF environment. Both methods record samples using a loop antenna and then process the recorded data into windowed samples. The window samples are used to train a logistic regression or an ANN to determine locations. After training the logistic regression and ANN can produce likelihoods for new samples from unknown locations.

Based on the results, either algorithm has an error performance of less than 1% error rate for 17 different locations in one building.

5.1 Future Work

The logistic regression and ANN are both supervised methods of classifying the locations. They require the samples to have a known location in order to train the weights and biases. This makes the weights and biases not generalize well to unknown locations. Any new location without any training samples or very few would be overpowered by the other samples. An unsupervised method attempts to look at all the samples regardless of output and make general characterizations. These characterizations are used to train an ANN to derive meaning from signals that have not been seen by decomposing them into basic characteristics and then determining what characteristics this new signal has. Thus, determining the characteristics is computationally intensive but assigning characteristics to signals is not. Thus it is possible to bring in new locations with out determining new characteristics or training an entire ANN from the ground up.

Disclaimer

The views expressed in this paper are those of the authors, and do not reflect the official policy or position of the United States Air Force, Department of Defense, or U.S. Government. This document has been approved for public release; distribution unlimited.

References


