Localization Through Wireless Access Point Channel State Information

Alexander Dewing, Robert Sun, Xiaonan Tong
adewing@stanford.edu ; robsun@stanford.edu ; xiaonan@stanford.edu

Thursday, 10 Dec. 15

1. Abstract

We propose and implement a way of estimating distance from Wi-Fi access points as a method of indoor localization. In contrast to many other solutions, the proposed method is software-only and works with current widespread technology, and can be implemented on current smartphones without silicon modifications. Our method utilizes machine learning to produce a relatively low root mean squared error of 1.27 meters, training on data collected between 2m and 15m away from the wireless AP.

2. Introduction

GPS is widely accepted as a standard of effective and accurate positioning. However, GPS requires signal acquisition from multiple satellites, a complication anywhere inside structures or underground. As our need for navigation systems grow, accurate indoor localization becomes increasingly valuable and necessary.

We propose an infrastructure-free method of triangulating local position in 3D space through predicting direct-path distances to multiple Wi-Fi access points. We differentiate our system from other localization methodologies by its straightforwardness and generalizability, while still attaining a fairly accurate result. Our system uses metrics that are exposed by Wi-Fi chipsets as part of normal operation and do not require any hardware changes, nor changes in router-software. Our technique readily works with 2.4GHz 802.11n routers.

3. Background Information

3.1. Intricacies of Wi-Fi localization

Wi-Fi transmits using Orthogonal Frequency Division Multiplexing (OFDM), meaning that it broadcasts on several narrowly separated subcarriers at the same time to increase data rate. Each subcarrier is synchronized at the beginning of a frame, thus the phase angle between the subcarriers encodes time of flight data for this packet. Furthermore, each subcarrier is measured with an additional phase angle on each antenna due to the spatial layout of the antenna array. At the receiver, the client can recombine the various signal characteristics from multiple antennas and gain information about the channel. The end result is a complex number describing the signal received per antenna per subcarrier. Collectively, this data is known as Channel State Information (CSI). We will analyze further our data format in section 4.2.

Radio signals in the gigahertz range are prone to reflecting off of walls and other surfaces. This introduces complications to our estimation of the direct-path by creating a multi-path system. Due to the effect of multi-path interference, as well as normal signal attenuation, the received signal will be a convolution of the original signal with the wireless channel's properties. See Figure 1.

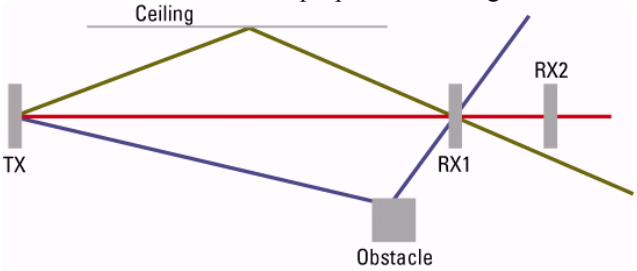

Figure 1 Multipath and Signal Propagation, shown with one transmitter TX and two antennas RX1,2. Sometimes a signal could even destructively interfere on RX1, but can be reconstructed through RX2.[1]

The 802.11n Wi-Fi specification transmits in the 2.4GHz ISM band, using 11 overlapping channels in the US and up to 14 elsewhere in the world. Each channel defines 64 subcarrier frequencies, each separated by 302.5kHz. Through complex signal processing on the CSI, it is possible to obtain a vector of possible AoAs (Angles of Attack) and isolate the direct path (the Euclidean line to the router)[2].

The signal processing can be summarized as this: the phase angle between subcarriers encodes Time of Flight (ToF), and the phase angle between antennas encodes AoA. By obtaining AoA and ToF of the direct path, one

[1] Image from *Multipath and Diversity* – Cisco
[2] SpotFi: Decimeter Level Localization using Wi-Fi. (Kotare et al. 2015)

can reverse calculate the exact distance to the receiver using only one router. See Figure 2.

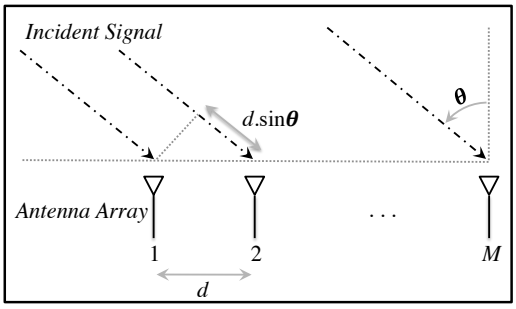

Figure 2. AoA (θ) is influenced due to the distance $d * sin(\theta)$ that the signal travels between antennas.

However, for our purposes, we wanted to see if machine learning techniques can predict direct path distance from the input dataset of CSI. We assume that given an accurate direct-path distance, a user could easily triangulate his/her position using a database of known-AP locations, possibly using SLAM techniques. Thereby, we simplify the problem to estimating the distance of the client from the router based on the Channel State Information data.

4. Data Collection

4.1. Hardware Setup

We are using an unsecured Wi-Fi Router running DD-WRT. The laptop collecting information is an HP Pavilion DV6, modified with an Intel Wireless Link 5300 Wi-Fi card. This laptop will run the Linux CSI Tool to extract channel measurements. As the IWL5300 is a multiple-input multiple-output (MIMO) chipset with 3 antennas (Figure 3), we obtain CSI for 3 antennas and 30 subcarriers each.

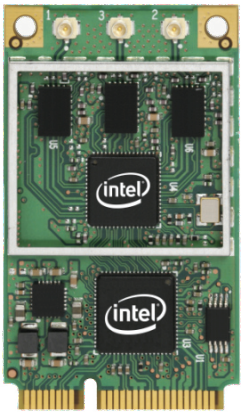

Figure 3 IWL5300

4.2. Data Collecting

As specified by the Linux CSI tool, each channel matrix entry is a complex number, with signed 8-bit resolution each for the real and imaginary parts. Each complex number encodes the gain and phase of the signal path between a single transmit-receive antenna pair[3].

We moved the laptop unit around the Wi-Fi router in concentric circles with radii of 2m to 15m in 1 meter intervals. Simultaneously, we downloaded a large file to ensure a steady stream of packets, and generated multipath interference by occluding the source and adding reflections. To do so, we placed reflective planes (aluminum foil makes for extremely radio-reflective surfaces) and moved them continuously. This is not an entirely rigorous data collection solution, but it does guarantee high levels of multi-path reflections.

For each of the approximately 1 million gathered packets, we separate complex data into gain and phase in a matrix, by combining information from 3 antennas on each of the 30 accessible subcarriers

$$csi_{packet} = \begin{bmatrix} \langle g_{1,1}, \theta_{1,1} \rangle & \cdots & \langle g_{1,30}, \theta_{1,30} \rangle \\ \vdots & \ddots & \vdots \\ \langle g_{3,1}, \theta_{3,1} \rangle & \cdots & \langle g_{3,30}, \theta_{3,30} \rangle \end{bmatrix}$$

and store it as an array s.t:

$$csi_p = \left[\langle g_{1,1}, \theta_{1,1} \rangle, \ldots, \langle g_{3,30}, \theta_{3,30} \rangle \right]$$

5. Modeling

For initial review and the first milestone. We decided to train our preliminary data using Gaussian Kernel Ridge Regression with Leave-One-Out Cross Validation. We wrote our tool in C++ program using the DLIB library (version 18.18).

We chose Kernel Ridge Regression for its speed advantage against Support Vector Regression when trained on large datasets. Also, we chose a linear-type regression since we believed linear combinations of the data (scaled differences between phases and magnitudes) created a meaningful result related to the direct-path time of flight since it seemed that delta phase created AoA and ToF info. We opted to use a Gaussian kernel in hope of finding Taylor series approximations of the trigonometric functions required in transforming the raw phase and magnitude data into our desired output.

As each adjacent packet is subject to very similar spatial constraints, we concatenated packet CSI data, hoping to improve performance. Initially, we used 10

[3] Linux 802.11 CSI Tool http://dhalperi.github.io/linux-80211n-csitool/

consecutively received packets per training/test pair since we thought that would give enough information for the multipath to be filtered out along with comparisons on a per channel basis for the direct path ToF estimation.

Example:

$$x = \langle csi_{p+0}, \dots, csi_{p+9} \rangle$$
$$y = 9(meters)$$

Our initial dataset comprised of packets at 2-4m. The KRR-LOOCV resulted in a mean squared error of approximately 0.3 meters^2 –indicating that a large portion of the samples will be within 0.5 meters of their actual distance from the router.

Given a proven methodology, we continued to experiment with more data, various feature spaces, and packet concatenations so as to best train the regression.

For our final results, all numbers are generated by Kernel Ridge Regression, using K-Folds Cross Validation where K=5. We computed the Gaussian (Radial Basis Function) Kernel by randomly subsampling 2000 feature vectors, and calculating inverse of their mean squared distance.

$$\gamma = \frac{1}{meanSquaredDistance(x_1, \dots, x_{2000})}$$

6. Results and Discussion

6.1. Expanded Dataset

Having proven that the distance estimation works acceptably for 2-4m, we sought to expand the data set to distances up to 15m. In this range we collected about 1 million packets, enough to generate at least 10^4 samples per distance class.

However, using the same methodology as the initial run, RMS increased to 2.78m, causing us to doubt the generalizability of the earlier experiment.

6.2. Real and Imaginary Components

To reduce training time in the next test, after verifying the limited effects, we reduced the training set size by randomly selecting 4000 samples per distance class. We also added real and imaginary components into the feature vector alongside the magnitude and phase angle interpretation. This gave us a total of 3600 features per sample ($3_{antennas}$*$30_{subcarriers}$*$10_{packets}$*$4_{features}$).

Unfortunately, adding real and imaginary components increased the distance estimation error to 3.1m.

Table 1. The method of generation, as well as the Root Mean Squared error of the KRR K-Folds Cross Validation. Subsequent uses of x refer to recursively operating on the feature vector.

#	SAMPLE GENERATION METHOD	DESC OF TRIAL	RMS ERROR (M)
1	$x = \langle csi_{p1}, \dots, csi_{p10} \rangle$, *2m-15m*	Expanded Dataset	2.78
2	$x = \langle csi_{p1}, csi_{p1}^{unprocessed}, \dots, csi_{p10}, csi_{p10}^{unprocessed} \rangle$ *2m-15m, limit 4000 samples per distance class.*	Using Real and Imaginary component as well as sampling	3.1
3	$x = \langle csi_{p1}, \dots, csi_{p10} \rangle$, *2m-15m, 4000 limit per distance class.*	Experimenting with Gamma value	See Figure 4.
4	$x = \langle csi_{p1}, csi_{p2}, xx^T \rangle$, *2m-15m, 4000 limit per distance class.*	Pairwise multiply 2 packets	1.27
5	$x = \langle csi_{p1}, \dots, csi_{p3}, xx^T \rangle$, *2m-15m, 4000 limit per distance class.*	Pairwise multiply 3 packets.	1.3
6	$x = \langle csi_{p1}, \dots, csi_{p4}, xx^T \rangle$, *2m-15m, 4000 limit per distance class.*	Pairwise multiply 4 packets.	1.66
7	$x = \langle csi_{p1}, (csi_{p1} csi_{p1}^T) \rangle$, *2m-15m, 4000 limit per distance class.*	Packet convolves with self.	2.1
8	$x = \langle csi_{p1}, \dots, csi_{p4}, (csi_{p1} csi_{p1}^T), \dots, (csi_{p4} csi_{p4}^T) \rangle$, *2m-15m, 4000 limit per distance class.*	Intra-packet pairwise multiply 4 packets.	2.13
9	$x = \langle csi_{p1}, \dots, csi_{p10}, (csi_{p1} csi_{p1}^T), \dots, (csi_{p10} csi_{p10}^T) \rangle$, *2m-15m, 4000 limit per distance class.*	Intra-packet pairwise multiply 10 packets.	2.56

6.3. Gamma experimentation

We observed the effect of Gamma value (for the RBF Kernel) on the cross validation error. Mechanically altering it to larger values and testing the same feature vector as defined in Section 6.1, we observed decreasing accuracy as we increased gamma. See Figure 4 for more details.

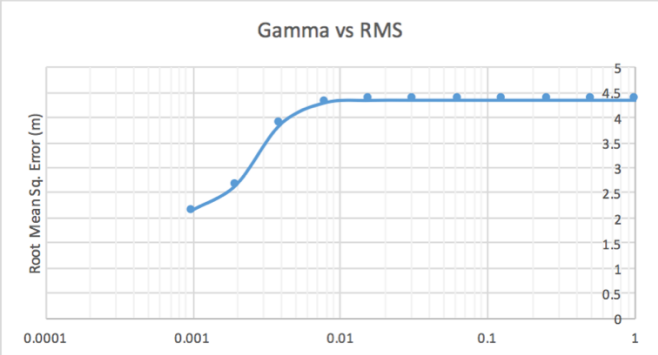

Figure 4. Gamma vs 5-folds root mean squared error.

Ultimately, we left the gamma value unchanged from the mathematical definition in Section 5, as that yielded the smallest RMS error (approximately 0.001 in Figure 4).

6.4. Pairwise Multiplications

From further analysis of the mathematics of calculating Time of Flight from received signals, we determined that some sort of convolution might be necessary. Intuitively, AoA and ToF are calculated using matrix multiplication from CSI.

Since AoA is calculated through combinations of phase and magnitude between antennas and ToF through combinations per channel, it seemed to be a reasonable choice for us to find all pair combinations of features and multiply the pair together. The algorithm should figure out which of these pairs contributed the most to the distance estimation.

Therefore, we focused on a smaller number of packets for runtime consideration, and generated our feature space using the outer product of the CSI vector. Using two concatenated packets, 65K features were generated, with up to 260K features for 4 packets. Essentially, we are learning on a single fully connected neural net layer.

This technique yielded substantially improved results. With a relatively small 1.27m of RMS error, the 2-packet vector gave the best generalization of actual distance. After achieving a substantially better result for the 2-packet vector, we attempted 3 and 4 packet groupings with a vastly higher number of features, but the test error increased. We attribute this effect to overfitting.

With infinite training time, we would have tried to further train with triplet-wise multiplications (multiplications of 3 numbers chosen from the feature vector), on feature vectors of 2 to 10 packets. However, performing triplet-wise multiplication on a two packet feature vector would generate 7 million features would not be feasible due to memory and processing constraints. However, it is highly possible to attain high accuracy with those features as the convolution style of data modeling makes perfect sense in this application.

6.5. Internal Pairwise Multiplications

Based on a theory that each packet might be linearly separable we sought to limit our feature vector by convolving each individual packet, instead of the entire multi-packet vector. This allowed us to train on larger concatenated packet counts with acceptable runtime.

Using 1-packet, 4-packets, and 10-packets, we could not improve RMS error; results were substantially worse than the current optimum of 1.27m. We obtained RMS errors of 2.1, 2.1, and 2.5m respectively.

7. Future Considerations

Using 2-packet convolutions and a Kernel Ridge Regression, we have achieved almost meter-level accuracy on a tough localization challenge. With no signal processing involved, we demonstrate errors that compare favorably against much more complex and advanced localization research. There is promise in applying machine learning to localization and Machine Learning could be coupled with more theoretical algorithms for improved results.

Due to our final algorithm's resemblance to neural net structures, future research should evaluate the use of deeper neural nets to approximate distance. One possible way to increase speed would be restricting the pair-wise multiplication to just phases or just magnitudes, since the phase and magnitude product makes little sense.

However, given current results and our initial constraints to commodity hardware, indoor localization appears increasingly ready for large-scale deployment.