MAINTAINING ACCURACY OF DEVICE-FREE LOCALIZATION SYSTEMS IN CHANGING ENVIRONMENTS

by

Brad Mager

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

The University of Utah

August 2014

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Brad Mager**

has been approved by the following supervisory committee members:

Neal Patwari	, Chair	**April 29, 2014** Date Approved
Sneha K. Kasera	, Member	**April 29, 2014** Date Approved
Tolga Tasdizen	, Member	**April 29, 2014** Date Approved

and by **Gianluca Lazzi** , Chair/Dean of

the Department of **Electrical and Computer Engineering**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Device-free localization (DFL) systems are used to locate a person in an environment by measuring the changes in received signal strength (RSS) on all the links in the network. A fingerprint-based DFL method, such as the type addressed in this thesis, collects a database of RSS fingerprints and uses a machine learning classifier to determine a person's location. However, as the environment changes over time due to furniture or other objects being moved, the RSS fingerprints diverge further and further from those stored in the database, causing the accuracy of the system to suffer. This thesis investigates the degradation over time of localization accuracy in radio frequency (RF) sensor networks using a fingerprint-based method with a machine learning classifier. We perform extensive experiments that allow quantification of how changes in an environment affect accuracy, through a process of moving specific items in a residential home one at a time and conducting separate localization experiments after each change. We find that the random forests classifier performs the best as changes are made, compared to three other classifiers tested. In addition, we present a correlation method for selecting the channel used with each link, which improves localization accuracy from an average of 95.2% to an overall 98.4% accuracy using random forests. We thus demonstrate that combining the random forests classifier with a correlation method of selecting a channel for each link offers a viable approach to developing a more robust system for device-free localization that is less susceptible to changes in the environment.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank Dr. Neal Patwari, who invited me to work in the SPAN (Sensing and Processing Across Networks) Lab and get involved in performing research that ultimately led to this thesis. His kind mentoring, guidance, and encouragement were a continual inspiration throughout this project.

Thanks are also due to the other members of my thesis committee, Dr. Tolga Tasdizen and Dr. Sneha Kasera, who provided insightful comments that led me to ask more questions about the data and refine my understanding of the algorithms used herein.

I must also acknowledge the following individuals: Dr. Maurizio Bocca, who worked as a post-doc in the SPAN Lab and from whom I learned a great deal about performing localization experiments using sensor networks; Peter Hillyard, a fellow SPAN Lab student, with whom I had many productive discussions about the algorithms and concepts I was exploring; and Philip Lundrigan, with whom I worked for six months at Xandem Technology on a NSF grant conducting localization and fall detection research, and who developed much of the code I used to process the data.

Without the help and support of all the individuals above, the research in this thesis would not have been possible.

INTRODUCTION

Radio frequency (RF) sensor networks have been employed to locate a person in an environment using radio tomographic imaging or fingerprint-based methods, which do not require the person to wear or carry a device. Such device-free localization (DFL) techniques typically measure the changes in received signal strength on the links in the network, then process that data to estimate a person's location. Over the last few years, researchers have demonstrated the ability of DFL systems to locate and track one or more people in homes or offices with an accuracy of less than one meter [1, 2, 3, 4, 5, 6, 7]. One of the main advantages of using radio waves is their ability to penetrate walls and other objects, thus allowing for more flexibility in placing the sensors in or around an environment of interest [8, 9, 10]. This becomes important when placing sensor nodes at two levels for vertical motion detection [11], where the nodes near the floor are more likely to be obstructed by furniture.

In fingerprint-based methods, training data in the form of received signal strength (RSS) measurements of each link are gathered as a person walks to several predetermined locations in the network. These measurements are then used by a machine learning algorithm to classify new data in terms of a person's estimated location [12, 13, 14, 15, 16, 17]. Although different supervised learning algorithms have been employed, the approaches are similar: the area inside a network is divided into cells, and the RSS fingerprints gathered during the training session when a person stands inside each cell are used to define each class. The classifier algorithm then processes the test data to determine which class a particular measurement belongs to, thus providing an estimate of a person's location.

Localization methods that rely on attenuation in mean RSS due to shadowing when a person is obstructing a link work well in environments with many line-of-sight paths. However, signals in cluttered environments will experience significant multipath, which can affect the quality of the links in unpredictable ways. A link that shows a marked decrease in RSS during a link line crossing may later show an increase in RSS when something in the environment has been moved. Thus, as the environment changes over time due to furniture or other objects being moved, the performance of the system may degrade because one method for selecting and processing the data was too finely tuned to a specific arrangement of the environment. Specifically, the training data gathered for fingerprint-based localization methods may no longer be viable if anything changes in the environment.

However, changes in RSS fingerprints or degradation in localization performance over time in a changing environment have only rarely been presented in the literature. For example, in [18] a DFL system was deployed in an apartment over several weeks. To allow the system to dynamically adjust to changes in the environment, the reference RSS value for each link was calculated using a moving average. With radio tomographic imaging, the system can tell where the person is in relation to the links and thus can recalibrate links individually when a person is not affecting that link, in essence recalibrating to empty room data. Fingerprint-based localization, however, requires not only empty room data, but also new training data of the person at each location. Therefore, dynamically adjusting reference RSS values is not a viable option for this localization method.

In [12], the authors performed tests in an apartment a month after the initial training data were gathered. The results degrade over time due to the change in environmental RSS, and the authors correct for this by removing links that experienced a large variation from the training data due to environmental instabilities. One concern with the link removal method is that, over time, fewer and fewer links are used for localization; eventually, there may be insufficient information for localization.

Neither of these papers notes the specific changes in the environment that caused any degradation in localization performance. Therefore, a more rigorous investigation of localization accuracy over time is needed, so that we can see how accuracy degrades when specific changes are made to the environment. Keeping track of changes allows for conducting repeatable experiments, which can help in creating a localization method that can handle the kinds of changes commonly made in certain environments, such as residences. In addition, we want to investigate link properties and methods of selecting and processing the data, so as to develop a more robust system that is less susceptible to such changes in the environment over time.

In this thesis we examine the degradation in localization accuracy over time, and test various methods to mitigate the decrease in accuracy. We perform extensive measurements in a residential home in which a network of RF sensor nodes has been deployed, gathering data on multiple channels in the form of received signal strength on all the links in the network. Before conducting the localization experiments, we first gather training data in which the subject stands in each of 32 designated test locations in the house. The passage of time is simulated by moving various items within the network. After each item is moved, a localization experiment is performed in which the subject stands at several test locations while RSS data are collected. We then employ a fingerprint-based method using a machine learning classifier to estimate the subject's location for each measurement sample.

The accuracy for each tested location and each channel is calculated by comparing the estimated locations to the known locations of the subject. However, as more and more objects are moved, localization accuracy degrades due to the environment's RSS signature varying from the signature observed during training, which the machine learning classifier relies on to produce correct location estimates.

The first goal is to find the most accurate classifier for this kind of data. Four machine learning classifiers are tested, to determine which one performs the best over the course of the experiments. Next we wish to find a combination of link features that provides consistent results despite changes in the environment, so that retraining the system need only be done infrequently. Here, a link feature is defined as some aspect of the link, such as channel or mean calibration RSS, that allows us to select a subset of links to use for processing the data.

We find that localization performance varies significantly by classifier and fingerprint link features used. For example, after 18 changes in the environment have been made for one experiment set, classification accuracy ranges from 52.1% to 87.2% for the four different classifiers. From this we determine that the random forests classifier consistently outperforms others tested. We then introduce a novel correlation metric for selecting the channel to be used as a link’s feature, and show that it achieves consistently high classification accuracy as the environment changes. This addition to the classifier further improves localization in the above experiment from a 96.5% average accuracy to 98.6%.

METHODOLOGY

System Framework

To set up the localization system, N sensor nodes are deployed in an environment such that the links between the nodes will provide maximum coverage of the area of interest. This generally involves setting up most of the nodes along the perimeter walls inside a room or building. The nodes constantly transmit and receive, and the transmission between each pair of nodes constitutes a link; since the RSS values in each direction of transmission are nearly the same, only one-way transmissions need be considered. Thus, the number of links $L = N(N–1)/2$.

The nodes employ a protocol in which each node takes a turn transmitting a packet of data consisting of the RSS values of the received transmissions from the other nodes. After all the nodes have transmitted in one cycle, they switch to the next channel in a predefined set of C channels. An additional sensor node listens to the traffic on the network and collects the data, which are stored in a file for later processing in the case of experiments, though a typical system would process the data in real time.

For fingerprint-based localization, training data are first gathered by having a person stand at several predetermined locations throughout the environment, so as to cover as much of the area as possible. For the experiments or real-time localization, RSS data are then collected at each time i, while a subject moves around in the environment. The data are processed to compute a feature vector, with the result being passed to a machine learning classifier. The classifier also accesses the training data, which have been processed using the same features as the experiment data; each sample of training data is labeled with the actual location where the person was standing at that time. The classifier then returns probabilities of the person being in each of the locations at every measurement sample (Figure 1). To determine the estimated location of a given sample, we find the location corresponding to the maximum value in the probabilities vector.

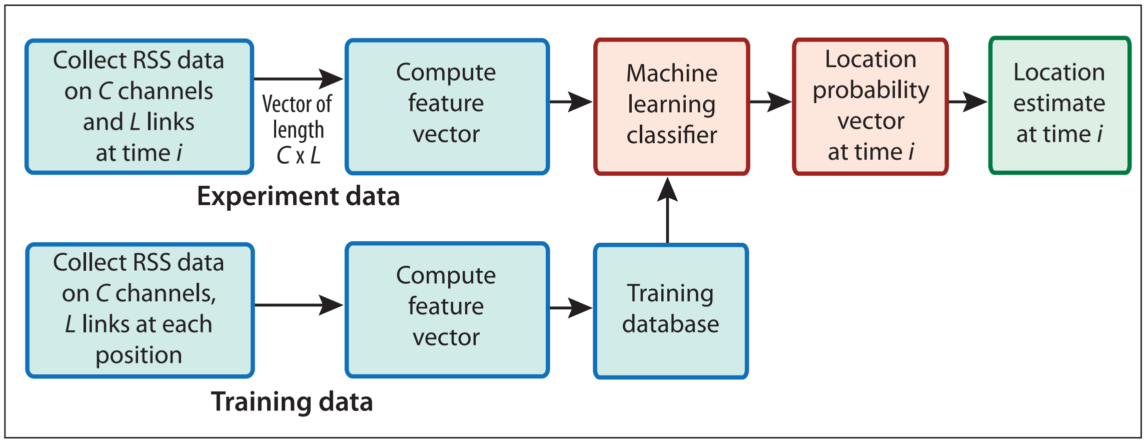
Collect RSS data on C channels and L links at time i
Vector of length C x L
Compute feature vector
Machine learning classifier
Location probability vector at time i
Location estimate at time i
Experiment data
Collect RSS data on C channels, L links at each position
Compute feature vector
Training database
Training data

Fig. 1. Algorithm framework for localizing a subject.

Experimental Approach

The goal of the experiments in this research was to create a repeatable and quantitative evaluation of how well a fingerprint-based localization system works in a changing environment. Thus, in addition to the typical localization experiments, we also identified several objects in the environment that would move or change in some way. Specifically, we wanted to make the kinds of changes that might happen in a residential home, while ensuring that locations and movements of the objects were documented.

The experiments were conducted on the main floor of a brick house, in an area approximately 84 m^2. A network of 30 transceivers was deployed in vertical pairs at 15 locations, with the lower nodes approximately 28 cm from the floor and the upper nodes 132 cm above the floor. These nodes are Texas Instruments CC2531 dongles, which transmit and receive in the 2.4 GHz range. A multichannel system with a time division multiple access (TDMA) protocol is employed, where each node takes a turn transmitting and all the nodes switch to the next channel of a predefined set of channels after each transmission cycle. For this research, data were gathered on the eight even-numbered channels (i.e., 12, 14, 16, 18, 20, 22, 24, and 26). Thirty-two locations were identified and marked by numbers on the floor, so as to cover a majority of the area, though furniture and walls prevented a completely uniform coverage. See Figure 2.

For fingerprint-based localization, we first gather training data to create the RSS

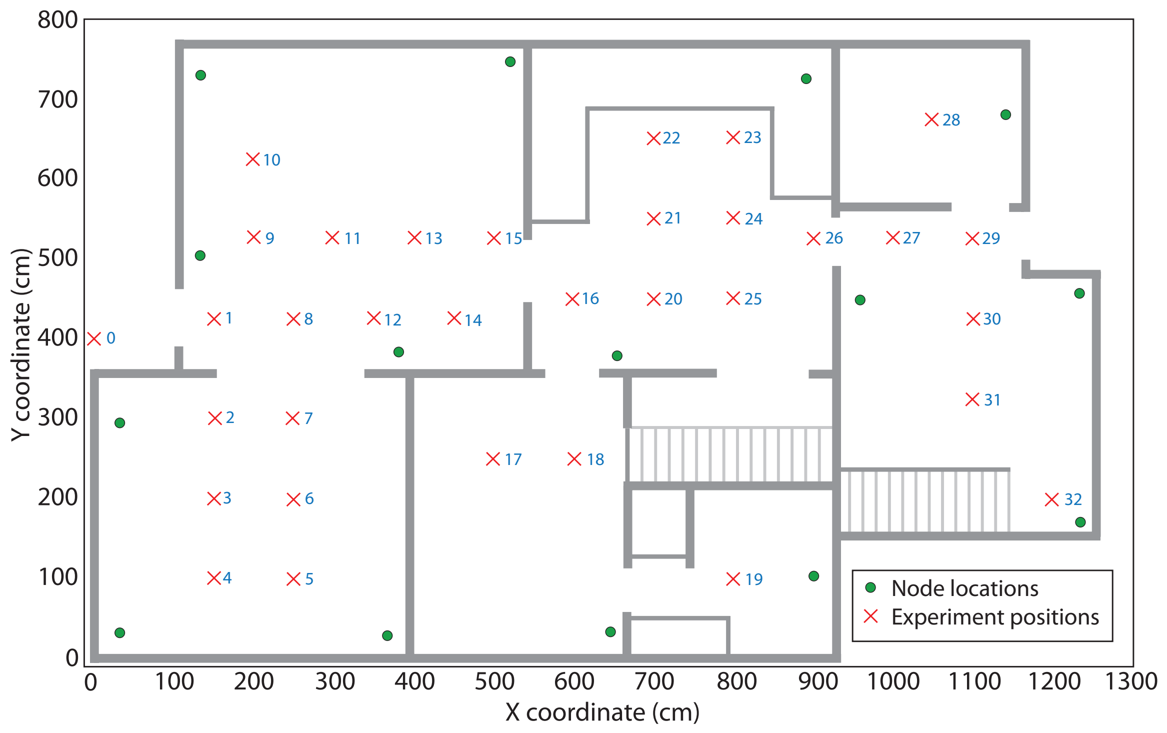

Fig. 2. Layout of house with sensor nodes and test locations. The nodes are deployed in vertical pairs.

signature that the machine learning classifier uses to identify where a person is. The subject stands at each of the 32 testing locations in the house for 60 s while RSS data are collected. To ensure the system is not dependent on one orientation of the person, the subject slowly spins around in place, making one rotation over the course of the 60 s.

The next phase consists of several experiments in which the subject moves from one location to the next every 30 s, standing still for a minimum of 20 s. The order and timing of the movements are predetermined, and the subject uses a stopwatch to know when to move to each spot. The location accuracy is determined only during the time a person is standing still at a designated location, since we are not concerned at this point about tracking movement.

There are two things to note here. First, because it takes time to move from one location to the next, the system ignores the first and last 5 s of data for each location, meaning we gathered 50 s of training data and 20 s of experiment data for the locations. Given that we collect one sample on all eight channels every 0.817 s, we have about 61 samples of training data and 24 samples of experiment data. Second, at the beginning of each experiment and the training, 60 s of calibration data are gathered with the subject standing outside the experiment area at location 0, providing a baseline measurement of RSS on all the links with no one in the network.

Thus, the term *training* refers to the procedure of gathering the RSS signature with a subject standing in every location; *experiment* refers to a localization test, where the system attempts to determine where a person is; and *calibration* refers to the collection of data when the network is empty.

For these particular experiments, we used attenuation as a feature for each link, where attenuation is calculated as the difference between the current sample's RSS and the mean of the calibration RSS. An additional part of the feature set was the link feature, defined earlier as an aspect of the link that allows us to select a subset of links to use for processing the data. For example, we can use channels as a feature, or select links based on a subset of sensor nodes, such as the links formed by only using the upper level of nodes in a two-level network.

Determining Accuracy

The accuracy for each experiment is calculated by comparing the estimated state to the actual state for each sample, where the actual state is known based on the timing of the experiment and when the subject was standing still at each measured location. For all samples for a particular known location, we find the number of samples for which the estimated state matches the known location. Overall accuracy for a channel for one experiment is calculated by taking the sum of all correct estimates for all the locations divided by the sum of the number of samples gathered at each location. In cases where multiple channels are used, the final accuracy is achieved by taking the mean of the accuracy values for all the channels.

Simulating the Passage of Time

We used a procedure to randomly change the environment in a way that might be done by residents of a home, but in a manner that allowed repeatability of the experiments. We first identified several moveable items and manner of movements (e.g., direction and distance), then, to reduce the chance of experimenter bias, the order and manner of movement were randomized by a computer program.

Objects that could be moved around the house, such as a stack of boxes or a houseplant, were assigned up to four locations (one in each of the four main rooms). One of these locations was then selected randomly so that the experimenter could not determine where these objects would end up. A bag of groceries was assigned four locations on the kitchen counters, then one of these was selected randomly. Each item was also assigned a default position, in which it was placed during training and where it remained for the first experiment of each set, before being moved during a later experiment. The Appendix contains details for each experiment set, listing the items moved for each experiment, and the timing of subject movement for all the experiments.

Five sets of experiments were conducted, with each set consisting of several separate data-gathering experiments. For each experiment, an object was moved or changed, so that the changes were cumulative over the set of experiments. Sets 1 and 2 included minor movements, such as slightly rotating a chair; however, later experiment sets included only more significant movements that would represent a greater change in the environment and thus potentially lead more quickly to a measurable degradation in localization performance. For example, in Set 3 each experiment involved moving two items at a time, while in Sets 4 and 5 the dining table and chairs formed a group that was either rotated or moved together, rather than as individual pieces. See Figures 3 and 4.

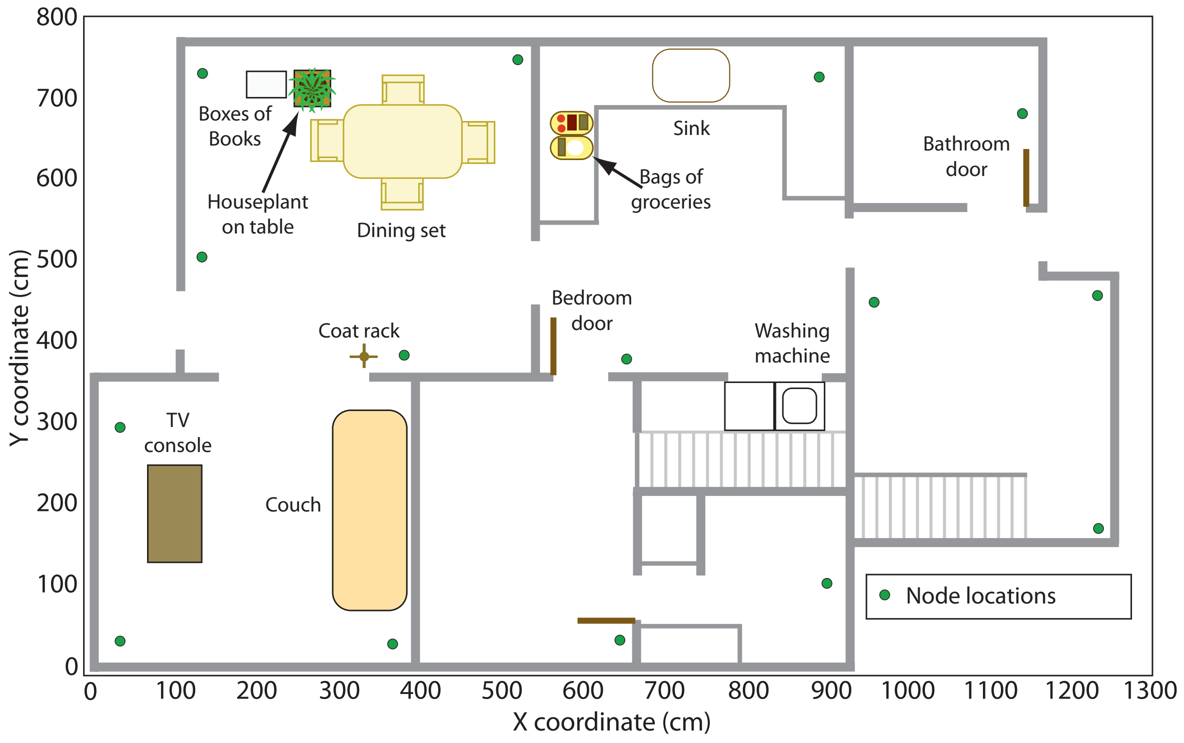

Fig. 3. House layout with node locations and default placement for objects.

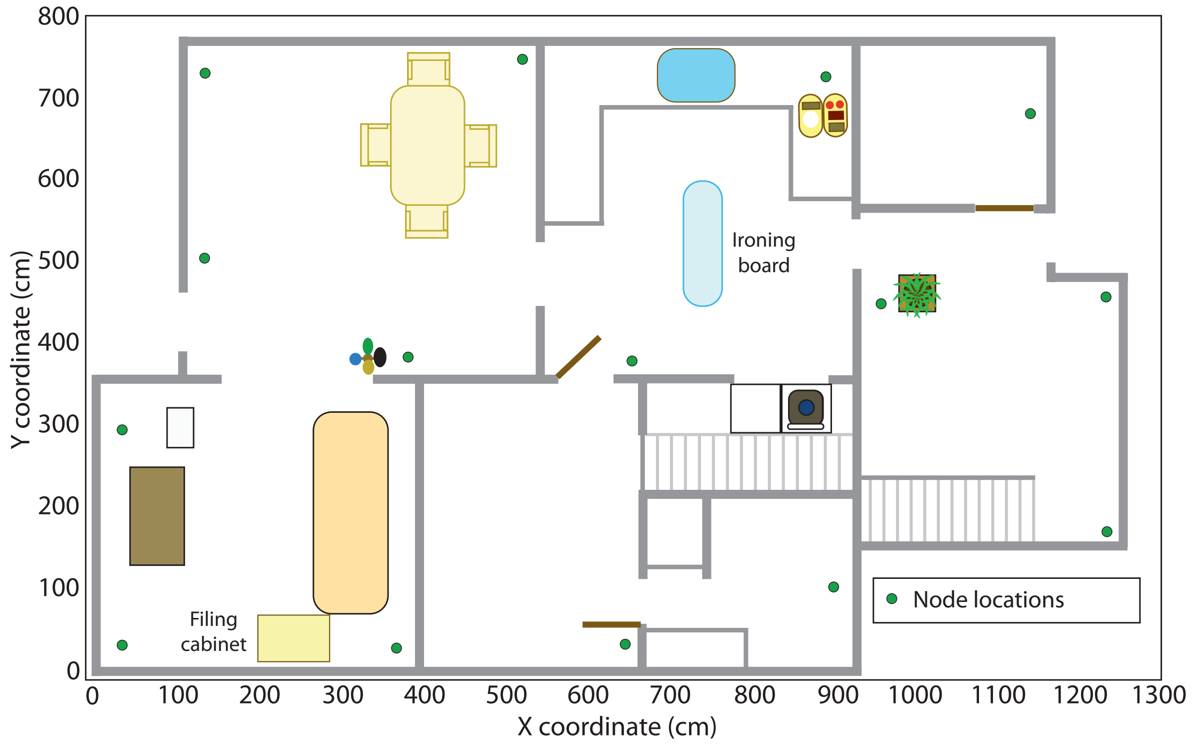

Fig. 4. House layout with node locations and final placement of objects for Sets 4 and 5.

RESULTS

Selecting a Machine Learning Classifier

Before a comparison can be made between approaches to processing the data, a control set of results has to be generated. At this stage, we are interested in which machine learning classifier performs the best. We use the data from experiment Set 2 for now, since it is the most representative of the data sets. Later we test the accuracy of the other experiment sets once a classifier and feature set have been selected.

Four machine learning classifiers were tested: random forests with 100 estimators, k-nearest neighbors (KNN) using three neighbors, support vector machine (SVM), and linear discriminant analysis (LDA). The first three classifiers were selected based on previous research using fingerprint-based methods for room-level localization, while LDA was tested due to its use in [12]. A brief description of each classifier follows.

Support Vector Machine

A support vector machine is a supervised learning algorithm that uses labeled data to classify new data. The SVM uses the labeled training data to create a model that can then predict which classes the new test data belong to, given a set of features from the test data [19, 20]. SVMs treat the data to be classified as points in a multidimensional space, in which the different classes can be separated by a hyperplane. In general, the SVM attempts to achieve the largest margin, defined as the distance between the hyperplane and the nearest training data point of any class. The greater this margin, the more likely the SVM will be able to correctly predict which class a new data sample belongs to. SVMs can also employ kernel functions to project data from a lower-dimensional space to a higher-dimensional space, as the data may become more separable in a higher dimension. The kernel used in this data set is the radial basis function (RBF), which maps samples into a higher dimensional space. The RBF kernel on two samples x and x' is defined as

$$K(x, x') = \exp(-\gamma |x - x'|^2) \quad (1)$$

Finally, SVMs can allow for a few anomalous data points falling on the incorrect side of a hyperplane by the inclusion of a user-defined penalty parameter, *C*, which determines how much the outliers are taken into account.

SVM Grid Search

We wish to determine the optimal values of *C* and γ for the SVM when using the radial basis function. We achieve this through a grid search by testing several pairs of each parameter and choosing the pair that results in the highest accuracy. In this case, we tested the results of Set 2, experiment 19. We performed a few iterations of the grid search, narrowing down the range of values with each iteration. From this we conclude that a value of $C = 0.95$ and $\gamma = 0.0005$ provide the best results, and these values are used for all implementations of SVM in this research.

k-Nearest Neighbors

The k-nearest neighbors algorithm is an instance-based learning method, where learning simply involves storing the training data for later retrieval, and each instance (i.e., new test sample) is assumed to correspond to a point in n-dimensional space [21]. When a new test sample arrives, the algorithm examines its relationship to the stored samples and assigns a target function value for this sample. In the case of k-nearest neighbors, the algorithm compares the test sample to a user-defined number of training samples, *k*, that are closest in distance to the test sample. The value of the test sample is then assigned the most common value of its *k* neighbors. Typically, as in our application of KNN here, the neighbors are defined using a standard Euclidean distance, though weighting can be applied so that closer neighbors have more influence than those further away.

Random Forests

The random forests classifier consists of a collection of single classification trees, in which each tree is grown by randomly drawing samples, with replacement, from the training set [22]. During tree construction, a small number of features is selected at random out of the input features, and the best split on these is used to split each node. In the original version of random forests, a new data sample is put down each tree, which provides a classification and votes for that class. The algorithm then chooses the classification with the most votes. However, the version of the algorithm used for this research combines classifiers by averaging their probabilistic prediction, rather than letting each classifier vote for a single class [20].

An advantage of random forests is that it is easy to choose the default parameters. One such user-defined parameter is the number of trees in the forest. In general, the more trees used, the better the results will be, though computation time increases. The results do not improve significantly after some point, due to diminishing returns. For these experiments, the number of trees, or estimators, was set to 100. Another parameter is the number of random subsets of features to consider when looking for the best split on each node. We used the default value of the square root of the number of features, though in [22] the author suggests that choosing one or two features is enough to provide near optimum results.

Linear Discriminant Analysis

LDA is a type of classifier that finds linear decision boundaries between classification regions that divide the input space [23]. Each class density, $P(X|y)$, is modeled as a multivariate Gaussian distribution, where the classes are assumed to share a common covariance matrix. Given $P(X|y = k)$ for each class k, LDA finds the probability that a particular data sample X belongs to a class by applying Bayes' rule:

$$P(Y = k|X = x) = P(x|k) \bullet P(k) \,/\, \mathrm{P}(x), \tag{2}$$

where Y is the class label and X is the vector of attenuation values on each link. One advantage of LDA is that it has no parameters to tune, making it simple to use.

Comparing Classifiers

Attenuation data from all the links in the network were processed using the various classifiers. Skewing of the class distribution is avoided by ensuring we gather the same number of samples per class. Figure 5 shows the accuracy rates for each method over time, i.e., over the course of the experiments, which are labeled from 1 to 19 in the graph. Table 1 summarizes the results.

Even before any items have been moved in the house, at experiment 1, k-nearest neighbors produces unacceptable localization results. This may be due to the large number of features (values from 435 links) and the possibility for significant overlap of the data in the feature space, which leads to KNN having a more difficult time in classifying new data points. LDA performs well at first, but its accuracy degrades more quickly than RF as the environment changes, perhaps because LDA assumes multivariate normal data within each class, where all classes have the same covariance matrix. As the environment changes, test data vectors may fall far from the training data vectors, resulting in the LDA assumption of Gaussian tails being inaccurate.

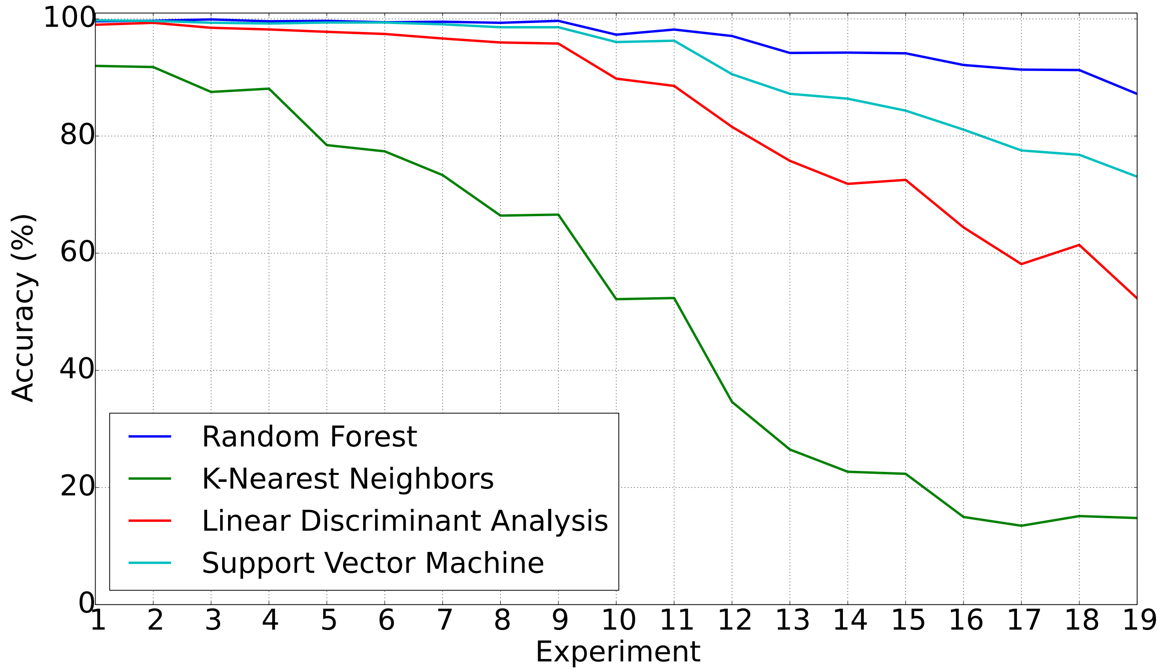

Fig. 5. Comparison of classifier methods for Set 2 experiments.

Table 1
Accuracy results from four classifiers

Classifier	Overall average
Random Forest	96.5%
K-Nearest Neighbors	52.1%
Linear Discriminant Analysis	83.9%
Support Vector Machine	91.7%

One advantage of random forests is that using a random split selection when building the tree classifiers makes the algorithm more robust to noise, since it does not put weight on any particular subset of the instances [24]. If we were to retrain the system after a change in the environment, we might find the feature vectors associated with some locations to have some altered values. We could thus think of the original training data as containing noise, i.e., some of the data will be misclassified since some pairs of feature vectors and labels no longer correspond to what the new training data would show.

Since the random forests classifier provides the best overall performance, we will use that for the remainder of the discussions. (Note: random forests with 500 estimators was also tested for this experiment set, producing an overall average accuracy of 97.3%, compared to 96.5% accuracy with 100 estimators. However, it is too computationally expensive for the small amount of gain in performance, so we decided that 100 estimators is sufficient.)

How Results Vary by Location and Channel

Up to this point, the results presented have used the average accuracies over the course of an experiment, i.e., by averaging the accuracies for all the locations tested in the experiment, then averaging over all the channels. This can hide some pertinent information about the results, so it is instructive to look at the results per location for a single channel. Figure 6 shows accuracy results for Set 3, experiment 10, on channel 12. This set tested just 10 of the locations. Accuracy figures are shown next to the corresponding locations.

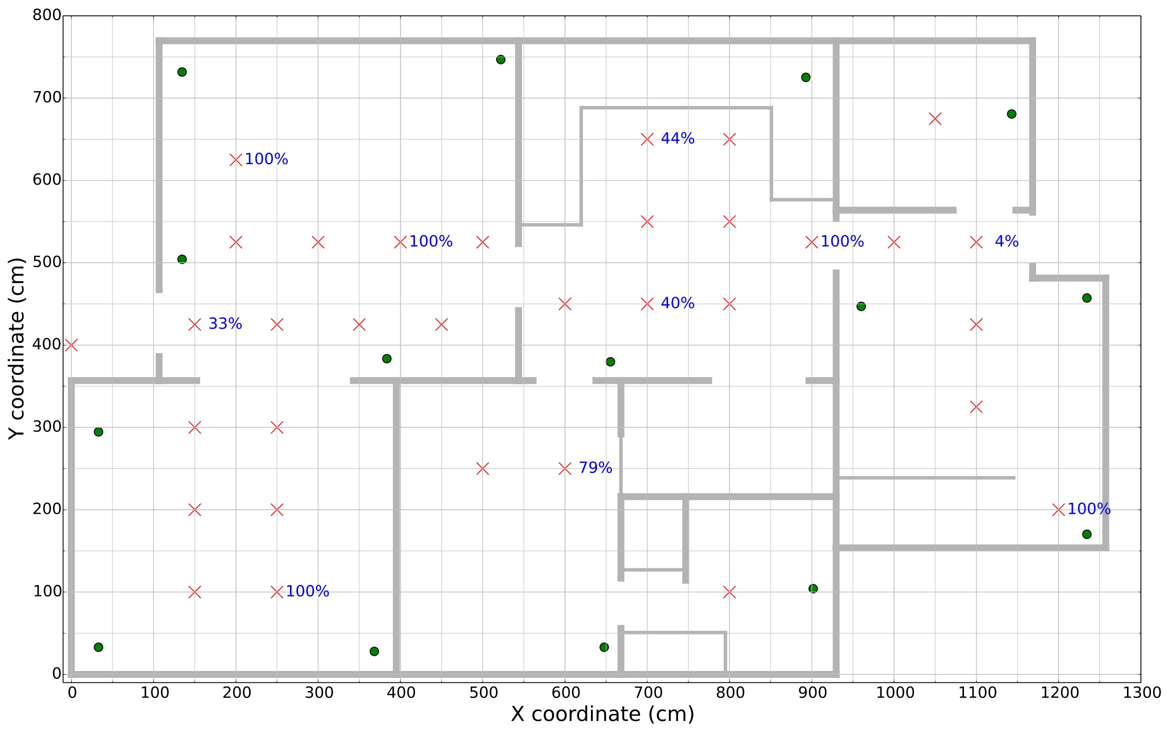

Fig. 6. Layout of house showing accuracy values for the 10 locations tested in Set 3. Values are for experiment 10, channel 12.

While many of the locations had 100% accuracy, one of the locations came in at just 4%, with others showing low results of 33% and 40%. Taking the average over all the channels can boost the accuracy rate if most of the channels have good results. However, the overall accuracy can be unduly diminished if it includes results from channels we would be better off ignoring. For example, channel 12 provides an accuracy of just 70.0% averaged over all the locations, while channel 22 for the same experiment provides an accuracy of 95.5%. This is one of the motivations for developing a method to find the best channels to use when processing the data, which is addressed in a later section.

Determining the Best Feature Set

At this point we can investigate whether it is possible to find a feature set that will help maintain acceptable localization performance even though the environment changes. That is, we wish to know if we can ignore certain data that may be adversely affecting the performance results, and only select the data that contribute to performance. Making a change in the environment will lead to different multipath effects on some of the channels on some of the links. Including links that have been too greatly affected could lead to a misclassification by the machine learning algorithm. Thus, we want to only include data from links and channels that remain viable throughout the course of any changes in the environment.

Link Sets Based on Network Topology

We begin by looking at the results of different link sets based on topology, selecting from the upper links (created by nodes on the upper level of the network), the lower links, the diagonal cross links (all the links traversing diagonally from the lower level to the upper level), or all links minus the lower links. Figure 7 provides a visualization of the link sets for a sample network.

Because most of the changes in the environment involve objects and furniture that sit on the floor, we expect the results generated by the lower links to degrade the most. Indeed, this is what we find. However, we also want to determine if using one of the other link sets will provide localization performance that equals or surpasses that of using all the links. Figure 8 shows the accuracy results for each of the link sets.

It is clear that just selecting a link set based on topology will not improve performance, though note that removing just the lower links provides results similar to using all the links.

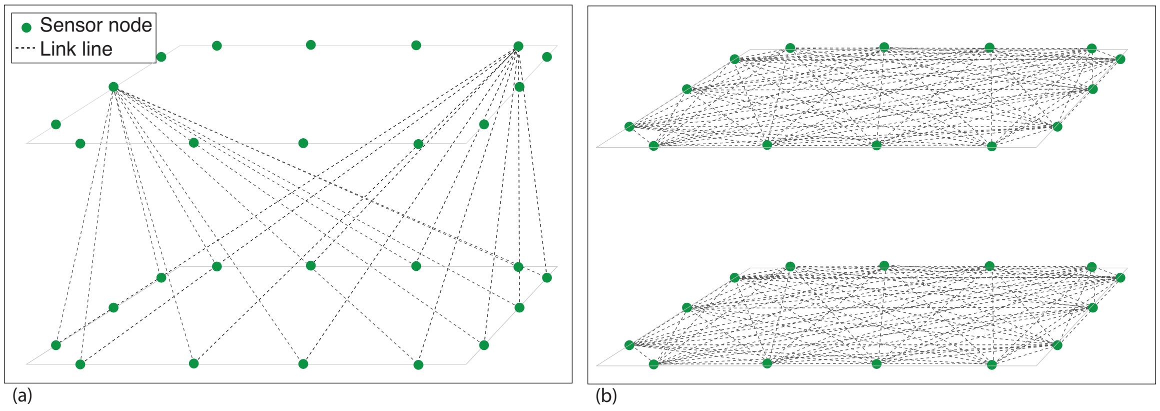

Fig. 7. Examples of link sets in a sample two-level network. (a) A few of the diagonal cross links. (b) The upper links and lower links.

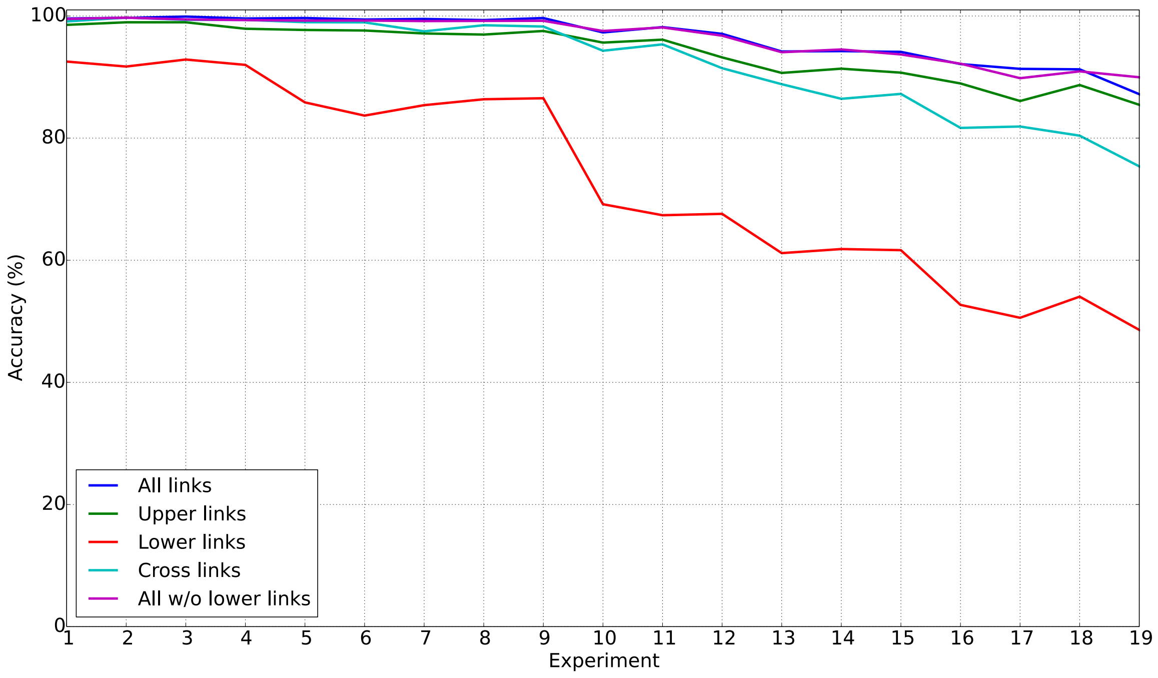

Fig. 8. Accuracy values for Set 2, for different link sets. Values are the results averaged over all the channels.

Best Channels — Highest Average RSS Method

Figure 9 shows the Set 2 accuracy results over time for each of the eight channels, compared with the average over all the channels. We can see that some of the channels provide better accuracy than others, which suggests it may be possible to identify a channel on each link that will offer the best performance overall. This is due to the fact that, as the multipath propagation of signals changes over time, some links experience destructive interference (said to be in "deep fade"), while others experience constructive interference (called "antifade" links in [25]). Links in antifade generally exhibit a higher average RSS than links in deep fade. Further, a link in antifade on one channel may be in deep fade on another. For this reason, we can look at the highest average RSS among all channels on a link to help us find one channel for each link that will provide better results than the others.

There are two initial approaches to using highest average RSS to find the best channels: 1) From the *training data*, select the channel on each link that has the highest average RSS during the calibration period, then use that channel for *all* of the experiments; 2) For *each experiment*,

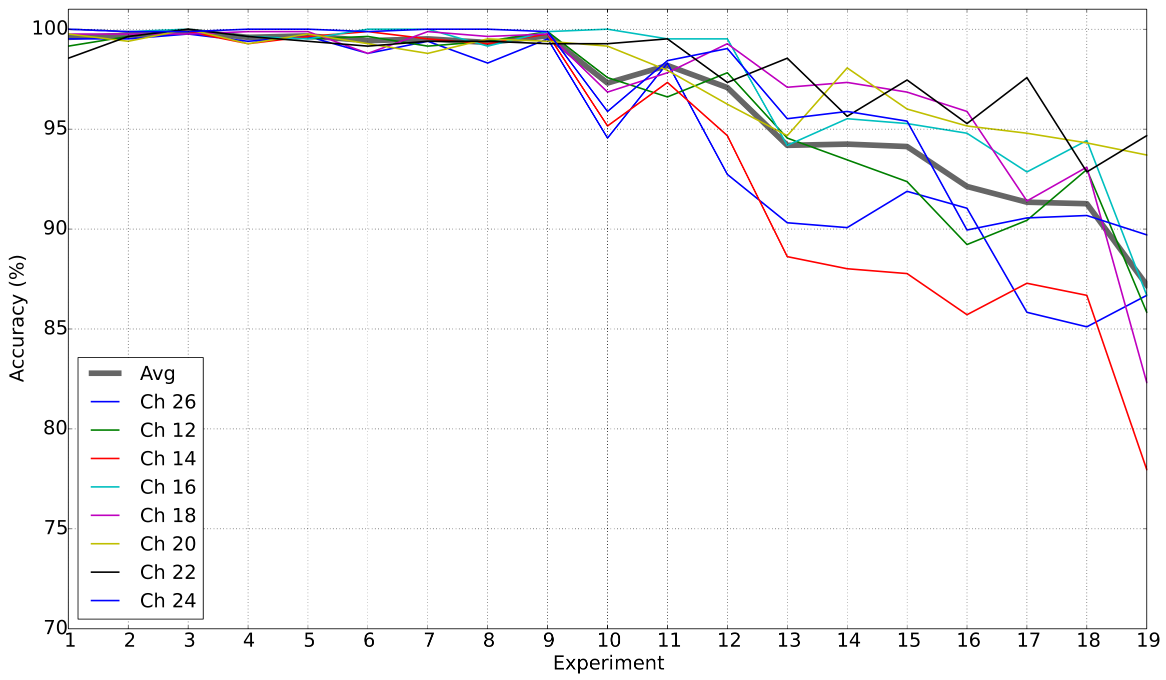

Fig. 9. Overall accuracy results for Set 2. This compares individual channels (thin lines) with the average of all the channels (thicker grey line).

select the channel on each link that has the highest average RSS during the calibration period, then use those link-channel pairs only for *that* particular experiment. One expects the second approach to provide better results, since this is akin to recalibrating the network. However, the first approach may be more desirable if we do not want to recalibrate when the network is empty.

Note that we are not choosing one channel to use overall for the experiments, but rather a single channel for each link. Thus, we may still use all the channels, but we now obtain a set of location estimates by providing the machine learning algorithm with the link data only for its best channel. Figure 10 compares the results for the best channels methods outlined above.

We can see that selecting a best channel per link from the calibration period before each experiment offers better performance than using all the channels. The rationale for this is that links with channels with the highest average RSS when the network is empty are less likely to be in deep fade, and thus more likely to be affected only when someone is standing on the link. Using this method improves the performance from an overall accuracy of 96.5% to 98.8%, an increase of 2.3% over the course of the experiments. While this may not seem like a significant improvement, consider that we have reduced the error rate by more than 65%.

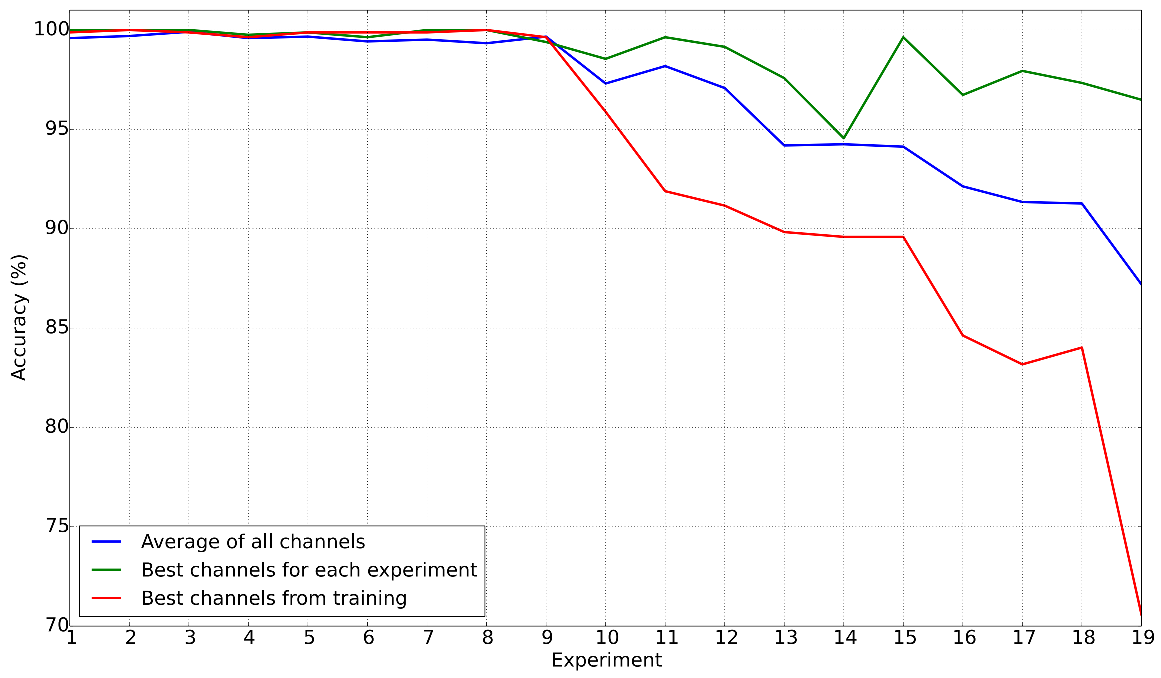

Fig. 10. Accuracy values for Set 2, comparing all channels with two best channels approaches.

Unfortunately, using the best channels from the training data is significantly worse than using all the channels, with a 3.38% decrease in overall average accuracy. Since the goal of this research is to find link features that will offer consistent results for the long term, it would have been preferable to find a feature set determined from the training data, rather than having to recalibrate the system from time to time. However, a recalibration method that does not require retraining must still be considered as a possible candidate for implementation in practical systems. In these cases, the system can be designed to recalibrate itself when the network is empty, i.e., whenever the resident leaves the premises.

Best Channels — Correlation Method

We can now try selecting a best channel by noting that a link in antifade exhibits a sharp decrease in RSS when a person crosses the link, and by further hypothesizing that two or more channels on a link that are most in antifade will likely show a greater similarity in their behavior. Thus, to find the channel that will most likely remain in antifade for a particular link, we find the two channels whose values are the most correlated over the course of the training experiment, then select one of those. We use the training data because they contain data for the subject standing in all possible locations, that is, all possible link crossings are represented. Plus, as stated earlier, it is desirable to find a feature set that does not require recalibrating the system.

We employ a function that finds the pairwise correlation of column vectors from the training data for one link, where each column represents one channel's data for the entire training time. This function returns the Pearson correlation coefficient, which is the covariance of two variables divided by the product of their standard deviations. We then simply select the pair of channels with the highest correlation coefficient.

Once a pair of channels with the highest correlation has been identified, we want to find the one that might offer the better performance over the other. One method involves selecting the channel with the highest average RSS over the course of the training experiment, while another approach involves selecting the channel that exhibits the greatest difference between the minimum and maximum RSS values during the experiment. We can further validate that one of these methods is better than the other by randomly selecting from one of the two most correlated channels. Figure 11, shows the accuracy for each experiment in Set 2 for each of the two correlation methods explained here, compared to using all the channels.

We can see that using correlation to identify two channels, then choosing the one that exhibits the greatest difference between the minimum and maximum values over the course of the training experiments provides slightly better results than the other methods — a 2.31% average improvement over using all channels, compared with 2.11% for using the highest average RSS to select from the two channels. (The average performance improvement increases to 4.07% and 3.70%, respectively, when considering just the last half of the experiments.)

Interestingly, the results from just choosing one of the two channels randomly are almost as good overall, offering a 1.65% average improvement. This is a reasonable result. Since there are only two channels to choose from at this point, and each may perform well based on having the highest correlation, we might expect that choosing one of them randomly will still produce similar results to the other methods.

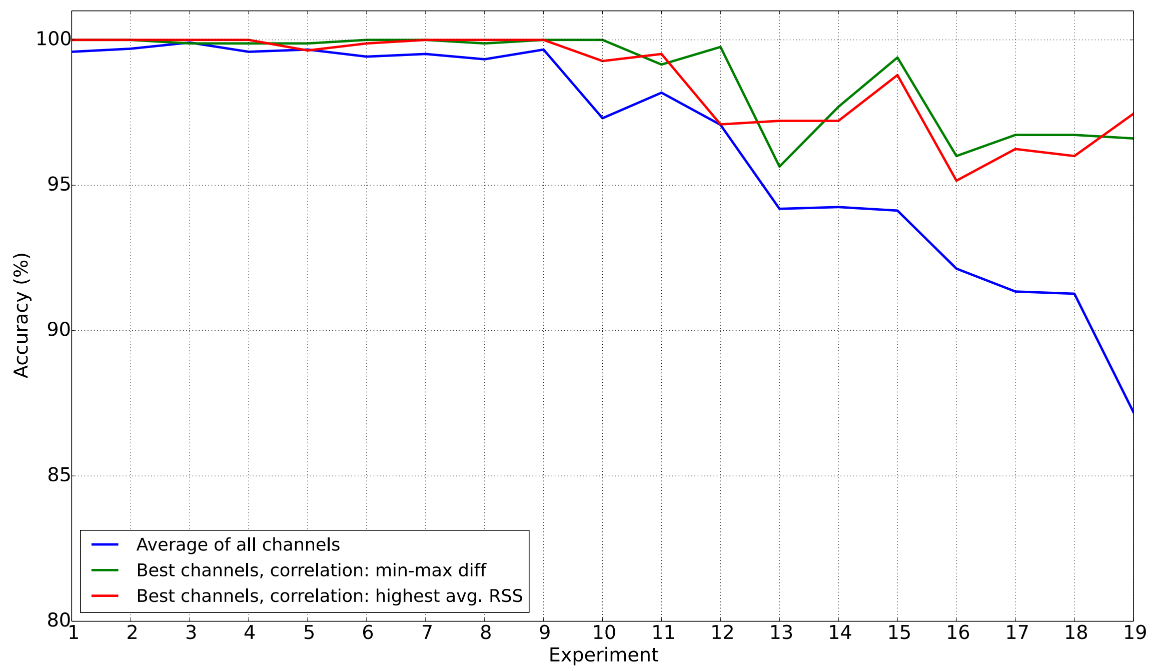

Fig. 11. Accuracy values for Set 2, comparing all channels with two correlation methods of selecting the best channels.

In addition, analysis of the two most correlated channels over all the experiment sets finds that 96.0% of these pairs consist of adjacent channels (e.g., channel pairs such as 14-12, 22-20, 26-24, since we are only using the even-numbered channels). One might expect to find that channels near each other would behave similarly. However, the specific channels chosen are evenly distributed among the eight possible channels, so it is unlikely that we could predict which channels will be chosen.

The results for the “min-max” and “highest RSS” correlation methods are too close to decidedly conclude that one is better than the other, so we will reserve that judgment until we look at the results of applying these methods to all the experiment sets. However, at this point it is reasonable to ask why we cannot just try finding the best channel that shows the greatest difference between the minimum and maximum values over the training data, and skip the step involving correlation. The answer lies in the behavior of links in deep fade, which can experience a high amount of variance when someone is in the network, and thus may at times exhibit a large difference in RSS values. The concept behind using correlation is to find the channels on the links that show spikes in the RSS values at the right times, i.e., when someone is crossing the link, which helps us identify links and channels in antifade.

Testing the Repeatability of the Results

Having identified, for one set of the experiments, a machine learning algorithm that offers the best localization performance for the system under investigation, as well as a feature set that improves performance over time, we now need to determine if the results are repeatable with other experiment sets. Five sets of experiments were conducted in the same environment over the course of two weeks. Each set consisted of several experiments in which one item was moved before each experiment, and all experiments in a set were conducted on the same day. The Appendix provides more information about each set, including what was changed before each experiment. Table 2 summarizes the number of experiments performed for each set and which locations in the house were tested.

Set 3 is anomalous in that the training data were not gathered immediately before the experiments. Rather, training data from the day before (Set 2) were used in an attempt to determine how the system would perform without fresh retraining. Instead, all items were returned to their initial locations and we tried to make the overall environment as close to what it was when the training data for Set 2 were gathered. The default experiment shows some degradation in performance, rather than the almost 100% accuracy achieved with the default

Table 2
Summary of experiment sets

Set	Number of experiments	Locations tested
1	16	1 – 32
2	19	1 – 32
3	10	1, 5, 13, 18, 22, 26, 32, 29, 20, 10
4	15	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31
5	15	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32

experiments in the other sets. This demonstrates that even conscientious attempts to make the environment the same as before can fall short. Inevitably, something is different enough to affect some of the links.

In Set 4 and Set 5, we tested the performance of the system using the same set of changes in the environment, though with the subject standing at different locations: Set 4 tested just the odd numbered locations, while Set 5 tested the even numbered locations. Set 5 was conducted on a different day, but every attempt was made to set up the environment as it was for Set 4. However, based on the experience with Set 3, we decided to gather new training data for Set 5. Each experiment then involved moving an item or changing the environment in the same way as was done for Set 4, which is why these sets have the same number of experiments.

Figure 12 shows the results of experiment Sets 1, 3, 4 and 5 using just the random forests classifier (with all links, and averaging the results over all the channels), compared with the results using the two correlation methods of choosing the best channels, and selecting the channel on each link that has the highest average RSS during the calibration period for each experiment.

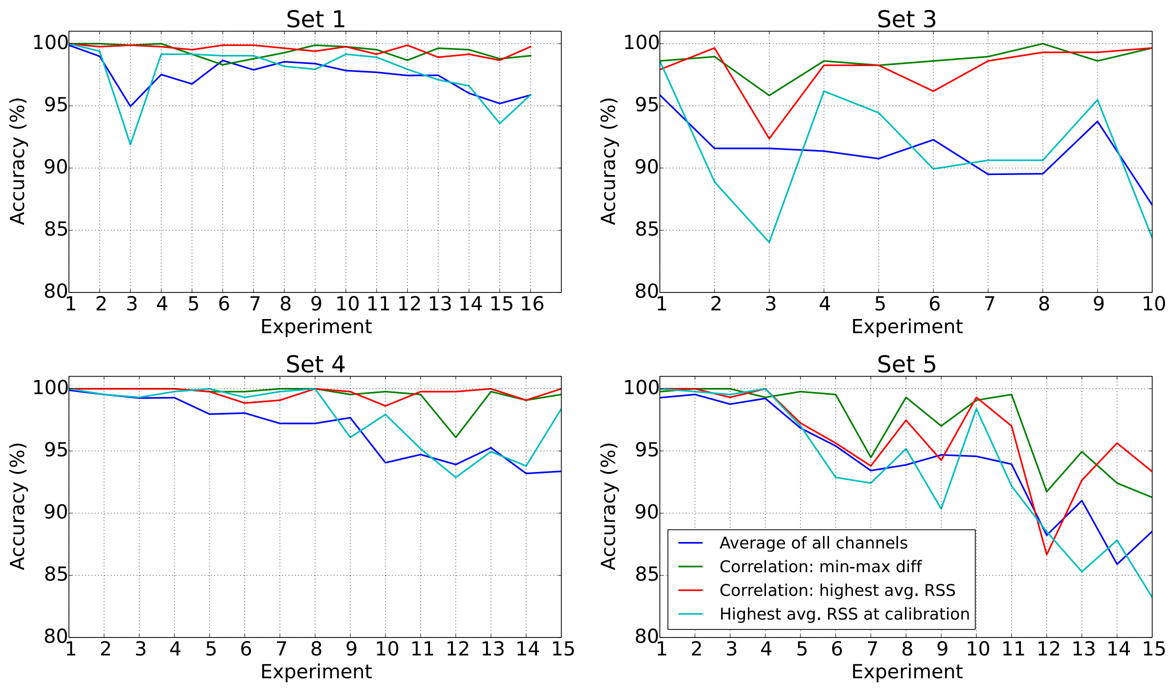

Fig. 12. Accuracy values for Sets 1, 3, 4, & 5, comparing all channels with various best channels methods.

Note that though Set 3's baseline results are relatively poor, as explained above, the addition of the best channels correlation feature significantly improves the results. In Set 1, experiment 3 shows a dip in accuracy, which may have been due to lost packets. Using the best channels correlation method, however, significantly improves the performance for that experiment.

The results for the two correlation methods are much better than using the channel with the highest average RSS value during calibration for each experiment. This suggests that we can use data from the training session after a network is set up, and not have to recalibrate periodically to maintain performance over time. However, we still see some unacceptable degradation in performance for some of the experiment sets, particularly Set 5. In addition, it is still not clear which of the two correlation methods offers the best choice; more experiments would need to be conducted to determine any statistically significant difference between the two methods.

Revisiting the Machine Learning Classifier Methods

Now that a particular method for improving performance over time has been identified, it is prudent to try it with the other three machine learning classifier methods that were earlier discarded in favor of random forests. Here, we choose the best channels correlation with highest average RSS method and use it with k-nearest neighbors (KNN), support vector machine (SVM), and linear discriminant analysis (LDA), so we can see how they compare to using random forest. As we can see by comparing Figures 13 and 14, random forests still provides the highest overall average accuracy for all the experiment sets, with SVM coming in a close second, though LDA and KNN show marked improvement.

Summary of Results

Table 3 summarizes the results obtained using different classifiers and feature sets. The overall performance figure is calculated by taking the average of the accuracy over all the experiments in each set, then averaging those values for the five sets. For feature sets using multiple channels, the accuracy is averaged over all the channels for each experiment.

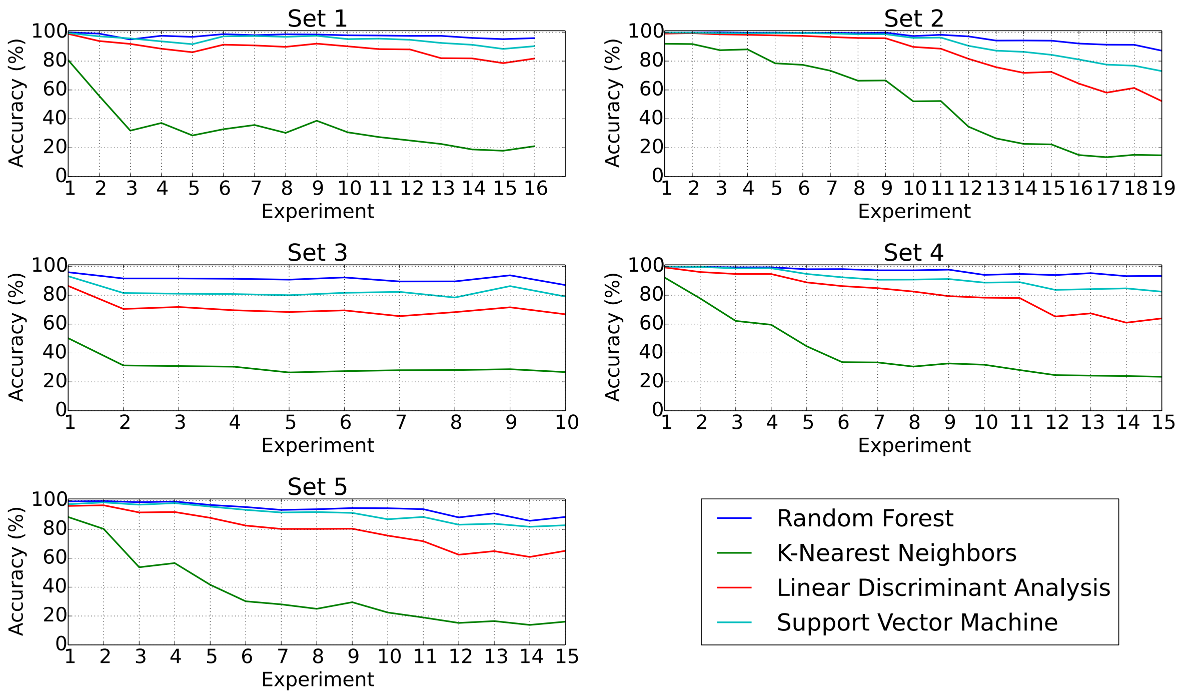

Fig. 13. Accuracy values for all sets, comparing classifier methods using all channels.

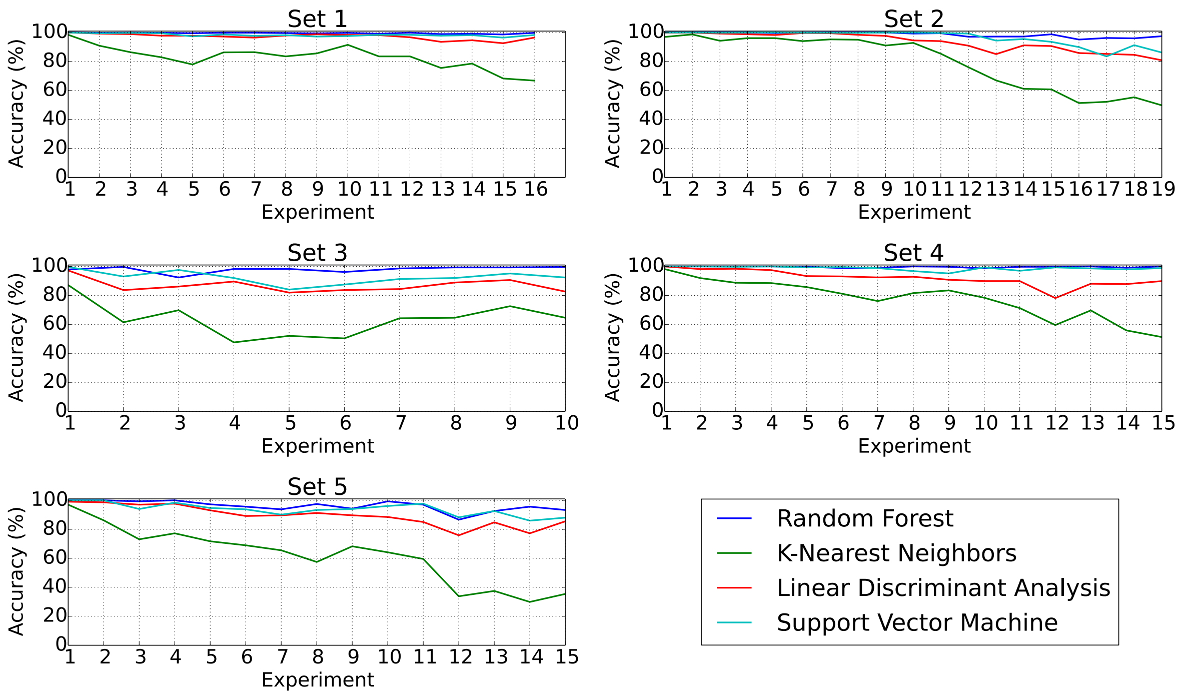

Fig. 14. Accuracy values for all sets, comparing classifier methods using the best channels correlation method.

Table 3
Summary of performance results

Classifier	Feature set	Performance
Linear Discriminant Analysis	All links, all channels	80.7%
Support Vector Machine	All links, all channels	90.2%
K-Nearest Neighbors	All links, all channels	38.8%
Random Forests	All links, all channels	95.2%
Random Forests	Highest avg. RSS	95.8%
Linear Discriminant Analysis	Correlation method	91.8%
Support Vector Machine	Correlation method	96.0%
K-Nearest Neighbors	Correlation method	73.0%
Random Forests	Correlation method	98.4%

For the sake of brevity in the table, “All links, all channels” refers to determining the performance using data from all the links in the network averaged over all eight channels. “Correlation method” refers to the process of finding the two channels for each link with the highest correlation coefficient from the training data, then taking the one with the highest average RSS during calibration. “Highest avg. RSS” refers to finding the channel for each link that has the highest average RSS value during the calibration period for each experiment.

CONCLUSION

In this thesis we have investigated the degradation over time of localization accuracy in RF sensor networks using fingerprint-based methods, and have tested various approaches to maintaining accuracy despite changes to the environment. We deployed a network of sensor nodes on the main floor of a residential home, and gathered data on multiple channels in the form of received signal strength on all the links in the network. We performed a rigorous analysis of how changes in an environment affect accuracy through a process of moving specific items in the house one at a time and conducting separate localization experiments after each change. We then employed a fingerprint-based method using a machine learning classifier to estimate the subject's location.

We conducted five sets of change-over-time experiments, where each set consisted of 10 to 19 experiments in which we simulated the passage of time by moving objects in the environment. As more and more objects were moved, the localization accuracy decreased more significantly for some classifiers than for others. From this we determined that the random forests classifier offers the best performance.

We next tested different links feature sets, including a correlation method for selecting the channel to be used as the link's feature. This method, in combination with the random forests classifier, helps achieve a much higher localization accuracy even as the environment changes. Using random forest, our correlation method improved localization accuracy from an average of 95.2% over all the experiment sets to an overall 98.4% accuracy. This demonstrates that we can develop a more robust localization system that requires less frequent retraining.

APPENDIX: EXPERIMENT DETAILS

This section provides details for each experiment set, listing the items moved for each experiment, and the timing for all the experiments, i.e., where the subject stood at each time. In Experiment 1 for each set, items were left in their default positions, where they were when the training data were gathered.

Set 1
March 7, 2014

Experiment	Item to Move
1	Default positions
2	Chair 1: rotate 45 counter-clockwise
3	Bedroom door: Closed
4	Chair 3: rotate 45 clockwise
5	Chair 3: move North 20 cm
6	Chair 2: move North 20 cm
7	Bathroom 2 door: Half-open
8	Chair 2: rotate 45 clockwise
9	Kitchen Sink: fill
10	Chair 4: rotate 45 clockwise
11	Chair 4: move North 20 cm
12	Couch: move forward 10 cm
13	Space heater: move to Dining room
14	Table: move East 15 cm
15	Box of books: move to Family room
16	Bag of groceries: move to 1

Time	Location	Time	Location
0:00	0	9:00	17
1:00	1	9:30	18
1:30	2	10:00	19
2:00	3	10:30	20
2:30	4	11:00	21
3:00	5	11:30	22
3:30	6	12:00	23
4:00	7	12:30	24
4:30	8	13:00	25
5:00	9	13:30	26
5:30	10	14:00	27
6:00	11	14:30	28
6:30	12	15:00	29
7:00	13	15:30	30
7:30	14	16:00	31
8:00	15	16:30	32
8:30	16	17:00	End

Set 2
March 11, 2014

Experiment	Item to Move
1	Default positions
2	Chair 3: move South 20 cm
3	Bathroom 2 door: Half open
4	Kitchen Sink: fill with water
5	House plant: move to Back room
6	Washing machine: open lid
7	Chair 1: move North 20 cm
8	Bags of groceries: move to 1
9	Chair 3: rotate 45 clockwise
10	Chair 2: rotate 45 counter-clockwise
11	Chair 2: move South 20 cm
12	Bedroom door: Closed
13	Ironing board: Set up in kitchen
14	Couch: move forward 15 cm
15	Coat rack: add coats
16	TV console: move South 20 cm
17	Table: move West 15 cm
18	Chair 1: rotate 45 counter-clockwise
19	Boxes of books: move to Dining room

Time	Location	Time	Location
0:00	0	9:00	17
1:00	1	9:30	18
1:30	2	10:00	19
2:00	3	10:30	20
2:30	4	11:00	21
3:00	5	11:30	22
3:30	6	12:00	23
4:00	7	12:30	24
4:30	8	13:00	25
5:00	9	13:30	26
5:30	10	14:00	27
6:00	11	14:30	28
6:30	12	15:00	29
7:00	13	15:30	30
7:30	14	16:00	31
8:00	15	16:30	32
8:30	16	17:00	End

Set 3
March 12, 2014

Experiment	Item to Move
1	Default positions
2	House plant: move to Family room
	Bedroom door: Half-open
3	Bags of groceries: move to 3
	Chair 1: move South 20 cm
4	Chair 3: rotate 45 clockwise
	Boxes of books: move to Kitchen
5	Couch: move forward 15 cm
	Chair 3: move North 20 cm
6	TV console: move South 20 cm
	Chair 1: rotate 45 counter-clockwise
7	Chair 2: move South 20 cm
	Table: move West 15 cm
8	Washing machine: open lid
	Bathroom 2 door: Closed
9	Coat rack: add coats
	Chair 2: rotate 45 counter-clockwise
10	Ironing board: Set up in kitchen
	Kitchen Sink: fill with water

Time	Location
0:00	0
1:00	1
1:30	5
2:00	13
2:30	18
3:00	22
3:30	26
4:00	32
4:30	29
5:00	20
5:30	10
6:00	End

Set 4
March 14, 2014

Experiment	Item to Move
1	Default positions
2	Couch: move forward 15 cm
3	Ironing board: Set up in kitchen
4	Bags of groceries: move to 4
5	TV console: move North 20 cm
6	Bedroom door: Half-open
7	Filing cabinet: move to Family room
8	Boxes of books: move to Family room
9	Coat rack: add coats
10	Kitchen Sink: fill with water
11	House plant: move to Back room
12	Dining set: Rotate 90 deg.
13	Washing machine: open lid
14	Dining set: move South 15 cm
15	Bathroom 2 door: Closed

Time	Location
0:00	0
1:00	1
1:30	3
2:00	5
2:30	7
3:00	9
3:30	11
4:00	13
4:30	15
5:00	17
5:30	19
6:00	21
6:30	23
7:00	25
7:30	27
8:00	29
8:30	31
9:00	End

Set 5
March 20, 2014

Experiment	Item to Move
1	Default positions
2	Couch: move forward 15 cm
3	Ironing board: Set up in kitchen
4	Bags of groceries: move to 4
5	TV console: move North 20 cm
6	Bedroom door: Half-open
7	Filing cabinet: move to Family room
8	Boxes of books: move to Family room
9	Coat rack: add coats
10	Kitchen Sink: fill with water
11	House plant: move to Back room
12	Dining set: Rotate 90 deg.
13	Washing machine: open lid
14	Dining set: move South 15 cm
15	Bathroom 2 door: Closed

Time	Location
0:00	0
1:00	2
1:30	4
2:00	6
2:30	8
3:00	10
3:30	12
4:00	14
4:30	16
5:00	18
5:30	20
6:00	22
6:30	24
7:00	26
7:30	28
8:00	30
8:30	32
9:00	End

REFERENCES

[1] D. Zhang, J. Ma, Q. Chen, and L. M. Ni, “An RF-based system for tracking transceiver-free objects,” in *IEEE PerCom’07*, 2007, pp. 135–144.

[2] X. Chen, A. Edelstein, Y. Li, M. Coates, M. Rabbat, and A. Men, “Sequential Monte Carlo for simultaneous passive device-free tracking and sensor localization using received signal strength measurements,” in *ACM/IEEE Information Processing in Sensor Networks (IPSN)*, April 2011.

[3] O. Kaltiokallio and M. Bocca, “Real-time intrusion detection and tracking in indoor environment through distributed RSSI processing,” in *2011 IEEE 17th Intl. Conf. Embedded and Real-Time Computing Systems and Applications (RTCSA)*, vol. 1, Aug. 2011, pp. 61–70.

[4] Y. Zheng and A. Men, “Through-wall tracking with radio tomography networks using foreground detection,” in *Proc. Wireless Communications and Networking Conference (WCNC 2012)*, April 2012, pp. 3278–3283.

[5] O. Kaltiokallio, M. Bocca, and N. Patwari, “Enhancing the accuracy of radio tomographic imaging using channel diversity,” in *9th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS 2012)*, October 2012.

[6] M. Moussa and M. Youssef, “Smart services for smart environments: Device-free passive detection in real environments,” in *IEEE PerCom-09*, 2009, pp. 1–6.

[7] C. Xu, B. Firner, R. S. Moore, Y. Zhang, W. Trappe, R. Howard, F. Zhang, and N. An, “Scpl: indoor device-free multi-subject counting and localization using radio signal strength,” in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, April 2013, pp. 79–90.

[8] J. Wilson and N. Patwari, “See through walls: motion tracking using variance-based radio tomography networks,” *IEEE Trans. Mobile Computing*, vol. 10, no. 5, pp. 612–621, May 2011, appeared online 23 September 2010.

[9] F. Adib and D. Katabi, “See through walls with Wi-Fi!” in *ACM SIGCOMM’13*, Aug. 2013.

[10] D. Maas, J. Wilson, and N. Patwari, “Toward a rapidly deployable rti system for tactical operations,” in *8th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2013)*, Sydney, Oct. 2013.

[11] B. Mager, N. Patwari, and M. Bocca, “Fall detection using RF sensor networks,” in *IEEE Personal, Indoor and Mobile Radio Communications Conference (PIMRC 2013)*, London, Sept. 2013.

[12] C. Xu, B. Firner, Y. Zhang, R. Howard, J. Li, and X. Lin, “Improving RF-based device-free passive localization in cluttered indoor environments through probabilistic classification methods,” in *Proc. Information Processing in Sensor Networks (IPSN-2012)*, April 2012, pp. 209–220.

[13] F. Viani, L. Lizzi, P. Rocca, M. Benedetti, M. Donelli, and A. Massa, “Object tracking through RSSI measurements in wireless sensor networks,” *IEEE Electronics Letters*, vol. 44, no. 10, pp. 653–654, 2008.

[14] M. Seifeldin and M. Youssef, “Nuzzer: A large-scale device-free passive localization system for wireless environments,” Arxiv.org, Tech. Rep. arXiv:0908.0893, Aug. 2009.

[15] F. Viani, P. Rocca, M. Benedetti, G. Oliveri, and A. Massa, “Electromagnetic passive localization and tracking of moving targets in a WSN-infrastructured environment,” *Inverse Problems*, vol. 26, pp. 1–15, March 2010.

[16] C. Xu, B. Firner, Y. Zhang, R. Howard, and J. Li, “Trajectory-based indoor device-free passive tracking,” *2nd Intl. Workshop on Mobile Sensing (IWMS)*, vol. 12, 2012.

[17] C. Xu, M. Gao, B. Firner, Y. Zhang, R. Howard, and J. Li, “Towards robust device-free passive localization through automatic camera-assisted recalibration,” in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 339–340.

[18] O. Kaltiokallio, M. Bocca, and N. Patwari, “Follow @grandma: long-term device-free localization for residential monitoring,” in *7th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2012)*, October 2012.

[19] W. Noble, “What is a support vector machine?” *Nature Biotechnology*, 24(12):1564–1567, 2006.

[20] F. Pedregosa, “Scikit-learn: Machine learning in Python,” *JMLR* 12, pp. 2825–2830, 2011.

[21] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[22] L. Breiman, “Random Forests,” *Machine Learning*, 45 (1): 5–32, 2001.

[23] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[24] T. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization,” *Machine Learning*, 1-22, 1998.

[25] J. Wilson and N. Patwari, “A fade-level skew-Laplace signal strength model for device-free localization with wireless networks,” *IEEE Transactions on Mobile Computing*, 12 May 2011, vol. 11, no. 6, June 2012, pp. 947–958.