LIPS: Learning Based Indoor Positioning System Using Mobile Phone-Based Sensors

David Mascharka and Eric Manley
Department of Mathematics and Computer Science
Drake University
Des Moines, Iowa 50311
Email: {david.mascharka, eric.manley}@drake.edu

***Abstract*—In this paper we investigate the problem of localizing a mobile device based on readings from its sensors utilizing machine learning methodologies. We consider a real-world environment, collect a dense set of 3110 datapoints, and examine the performance of a substantial number of machine learning algorithms. We found algorithms that have a mean error as accurate as 0.76 meters, outperforming other indoor localization systems. We also propose a hybrid instance-based approach that results in a speed increase by a factor of ten with no loss of accuracy in a live deployment over standard instance-based methods. Further, we determine how less dense datasets affect accuracy, important for use in real-world environments. Finally, we demonstrate that these approaches are appropriate for real-world deployment by evaluating their performance in an online, in-motion experiment. The Learning Based Indoor Positioning System (LIPS) Android application source has been made available on the web.**

I. Introduction

As smartphones and other mobile devices become ubiquitous, applications that are able to harness contextual information such as position become increasingly powerful. Uses for indoor localization systems include context-based advertising [1], emergency response and assisted living [8], robotics applications [12], and indoor navigation.

Machine learning is a field of artificial intelligence dealing with algorithms that improve performance over time with experience. Supervised learning algorithms for regression are trained on data with the correct value given along with each variable. This allows the learner to build a model based on the attributes that best fit the correct value. By giving more data to the algorithm the model is able to improve. Learning can be described in this way as improving performance. The measure of performance is how well the algorithm predicts the regression value given a set of variables or attributes. Machine learning algorithms provide excellent solutions for building models that generalize well given large amounts of data with many attributes by discovering patterns and trends in the data; a task that is often difficult or impossible by other means.

With an increasing number of sensors being made available in the majority of mobile devices, large amounts of data can be collected and used to aid in the localization process. Machine learning algorithms are a natural solution for sifting through these large datasets and determining the important pieces of information for localization, building accurate models to predict an indoor position. Machine learning algorithms may also provide a fast, efficient method for indoor tracking, which will often be more useful to applications than static localization.

In this study, we perform a large-scale analysis of a wide range of machine learning algorithms using real-world data for localization. We also present a hybrid approach for certain learning algorithms to lower prediction times, making them suitable for real-world use. Finally, we conduct an online, in-motion evaluation of the best-performing models to show their usefulness when fully deployed in a live, dynamic environment.

The paper is organized as follows. In Section II, we examine other indoor localization systems and their limitations. In Section III, we describe our application and data collection process, the testing environment, and how our analysis was conducted. Section IV presents the results of our analysis on the full dataset and smaller partial sets of data in an offline environment. This section also presents our hybrid approach to localization and results from our online, in-motion analysis. Finally, the paper concludes with a discussion of future directions for research in Section V.

II. Related Work

Indoor localization research has garnered a good deal of interest from both academia and industry, with numerous systems being proposed using a variety of technologies. A major disadvantage of many of these systems (such as infrared [12], ultrasound [11], and rfid [10]) is that they require substantial infrastructure changes and incur a significant cost to deploy. Effort has been made to devise localization systems that require little to no infrastructure change using Bluetooth [3] and WiFi signal strengths [6], [9] with some success. The systems developed using WiFi signal strengths show promise but have yet to receive widespread adoption. These systems can be divided into two categories: those using a fingerprinting aproach using algorithms for "nearest neighbor in signal space" and those using more complex signal ranging algorithms.

Fingerprinting-type approaches can achieve accuracies up to 2 meters on average [9], but current research is limited in that only one or a few algorithms are considered and many of the sensors available in most modern mobile devices are not taken into account. Further, a very large dataset will require a substantial amount of time to predict a position, hindering real-world deployment.

Signal ranging-based algorithms can achieve similar accuracy, but require a substantial amount of network configuration information and have difficulty modeling signal propagation

through obstacles. A few attempts have been made to remove the requirement of knowledge about access points in range [6] by using sniffer devices and a centralized localization server. The downside is that localization cannot be performed on the device itself. Additionally, these models are only able to take into account signal strengths, missing other variables that fingerprinting can handle.

In this paper, we expand on the fingerprinting approach by exploring a variety of machine learning algorithms using WiFi signal strengths and other embedded sensors. The model built for an algorithm can be easily implemented as part of an application and installed for localization on the device itself.

III. Methodology

The localization process consists of two distinct phases: data collection and analysis. Before collecting our dataset, some preprocessing was necessary to determine which WiFi base station IDs to store. In an initial scan of the various signals received throughout the building, we detected a few portable hotspots likely from people in the building tethering their devices, which were excluded from data collection.

A. Android Application

We developed an Android application called LIPS (Learning Based Indoor Positioning System) which can be used for both data collection and live deployment, and we released it under an open-source license available at `github.com/davidmascharka/LIPS`. The app allows a data collector to select which building they are in, allowing for collection of multiple datasets in multiple buildings. Each building has the WiFi access points that will be used listed in the application, and any access points received not in the chosen building's list will be ignored. The data collector can also select a room or building size and a grid is drawn of the proper size (optionally overlaid onto a map of the space) to allow the data collector to more easily indicate their position.

B. Data Collection

The data collection phase consists of moving about the building taking readings of the WiFi signal strengths and pulling data from the other sensors in the device. The data is associated with a user-provided location and written to a text file on the device, which can be pulled from the device later for analysis.

A single datapoint, in our study, consists of 172 attributes corresponding to values from each sensor on the device. We took into account a value from the light sensor, GPS/Network location data, signal strengths to 156 WiFi radios from 21 access points, and x, y, and z values for the device accelerometer, magnetometer, rotation sensor, and orientation sensor. The number of WiFi signals to account for will vary depending on where the localization system will be deployed. While we recorded GPS/Network location data at each point, the accuracy was generally extremely low in our experiments. However, for areas near windows or doors, this may be useful to account for in predicting a position, which motivated us to record it.

In total, we collected 3110 datapoints in the Cowles Library at Drake University in a space about 62 meters wide by 39 meters long using a Motorola XT875. Collection of data involved initiating a scan of WiFi networks in range of the device to record up-to-date signal strengths. This and data from all the other sensors in the device were written to a text file with the data collector's position in the room, indicated by the data collector in the application itself. This process took an average of five seconds to complete on our device.

Measurements were taken in a grid based on 2 foot by 2 foot ceiling tiles in the building and all measurements are relative to the building. This is an arbitrary measure chosen for ease of use in our case and can be modified to fit any desired building layout. Latitude and longitude coordinates can be interpolated. The map of collected datapoints can be seen in Figure 2. The missing points are caused by obstacles. The left portion of the building contained stacks of books while the rest of the area consisted mostly of open space with tables and desks throughout. We chose to include the area with the stacks of books specifically because it represents a particularly challenging environment.

C. Analysis

In the preprocessing and offline analysis phase, various machine learning algorithms are trained and their errors measured. Some preprocessing of the data may be beneficial, depending on the algorithm.

The implementations of the algorithms used was provided by the Waikato Environment for Knowledge Analysis (WEKA) [7], developed at the University of Waikato. WEKA allows for easy selection of various algorithms and parameter options, which allowed us to test different parameters for each algorithm.

The final phase is an online analysis, in which the algorithms are tested in real-world conditions including with the user in motion, receiving new, unclassified data to process live. Live, in-motion testing is difficult to conduct with accuracy and to our knowledge has not been done to this extent for other similar indoor localization systems. To test our algorithms, we designed a route to walk through the building that would give a representative time series for someone actually traversing the building. While walking, new readings would be taken from the sensors whenever available and given to the algorithms to predict a position. The prediction was output along with a timestamp and written to a text file on the device. A timestamp was also recorded when the researcher reached each vertex of the path and changed directions. The vertex timestamps were used to calculate the researcher's actual position at each time of prediction by interpolating position between vertex timestamps.

IV. Results

Because we were interested in performance not only in a static, offline analysis but in a real-world environment in motion, we split our analysis into two distinct phases: offline and online. The offline analysis section presents the performance of several algorithms in predicting a position. The online analysis section introduces our hybrid approach and shows its performance in live, in-motion testing.

A. Offline

1) Full Dataset: We examined the performance of 20 well-known machine learning algorithms and 87 different parameter

TABLE I. MEAN ERRORS OF EACH ALGORITHM IN THE x AND y PREDICTIONS IN THE OFFLINE ANALYSIS.

Algorithm	Best Mean Error (m)		Algorithm	Best Mean Error (m)	
	x	y		x	y
K* (12)	1.134	0.762	SMOReg (6)	2.043	2.010
MultiScheme	1.135	0.762	RandomTree	2.150	1.950
Voting	1.058	1.015	MultilayerPerceptron (9)	1.963	2.530
k-Nearest Neighbor (22)	1.417	1.227	DecisionTable	2.595	1.964
RBFRegressor (11)	1.374	1.333	RBFNetwork (3)	2.798	3.434
RandomForest	1.454	1.470	LinearRegression	3.478	3.362
M5P	1.671	1.429	LWL (6)	5.199	4.159
M5Rules	1.756	1.604	DecisionStump	6.220	4.851
REPTree	1.790	1.665	SimpleLinearRegression	6.655	4.822
MLPReg (6)	1.821	2.143	ZeroR	20.100	7.249

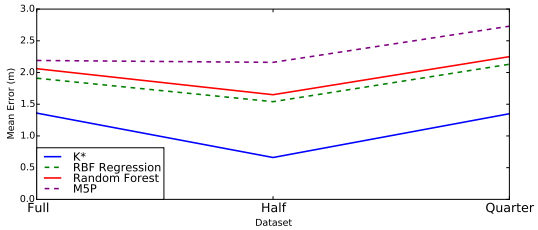

Fig. 1. Performance of the best algorithms on each dataset in offline analysis.

settings on our dataset. All results were obtained using a tenfold cross-validation and verified with ten repetitions. Table I shows the table of results, giving the best mean error for each algorithm. The number in parentheses indicates how many parameter variations of the algorithm were trained.

The best-performing algorithm was the K* algorithm [4], an instance-based approach that uses an entropy-based distance function, with mean errors of 1.13 and 0.76 meters for x and y position, respectively, for an absolute mean error of 1.36 meters. Another algorithm that performed well for both x and y classification is the RBFRegressor implementation [5], a radial basis function network trained in a fully supervised manner, with mean errors of 1.37 and 1.33 meters, respectively, resulting in an absolute mean error of 1.91 meters. When combining these algorithms using a voting scheme taking the average of both predictions, the x error is reduced to 1.05 meters, while the y error increases slightly from only using K* to 1.01 meters.

2) Partial Datasets: We were also interested in how performance may deteriorate with less training data, which will be useful for faster and easier real-world deployment. We examined algorithm accuracy using one-half the full dataset and one-quarter the full dataset, with both sets of data in a grid pattern like the full dataset. Table II shows the results of each analysis. A comparison of the four most accurate algorithms over each dataset can be seen in Figure 1.

The error using half the original dataset results in slightly increased accuracy for most of the tested algorithms. The K* error is reduced to 0.563 meters in x prediction and 0.395 meters in y prediction, resulting in an absolute error of 0.763 meters. The RBF algorithm's error is also slightly reduced to 1.113 meters and 1.053 meters for x and y prediction respectively, which gives an absolute error of 1.532 meters. This slight reduction leads us to believe that the full dataset may have a density that results in overfitting and too little variation between sets of readings. It may also be the case that the RBF model performs better using half the data because the network built on the full set was too small to completely capture the complexity of the data.

The errors when using only one quarter of the original

TABLE II. MEAN ERRORS IN x AND y POSITION FOR PARTIAL DATA WITH DIFFERENCES FROM THE FULL SET. BETTER PERFORMANCE COMPARED WITH THE FULL DATASET IS INDICATED BY GREEN WITH WORSE PERFORMANCE IN RED.

Algorithm	Mean x Error (m)		Mean y Error (m)		Difference (m) x/y	
	Half	Quarter	Half	Quarter	Half	Quarter
K*	0.563	1.077	0.395	0.831	-0.572/-0.368	-0.058/+0.068
k-Nearest Neighbor	0.695	1.396	0.841	1.475	-0.723/-0.387	-0.022/+0.247
RBFRegressor	1.113	1.561	1.053	1.511	-0.26/-0.280	+0.187/+0.178
RandomForest	1.146	1.567	1.194	1.622	-0.308/-0.277	+0.113/+0.151
M5P	1.450	1.823	1.603	2.030	-0.221/+0.173	+0.152/+0.600
M5Rules	1.597	2.067	1.847	2.118	-0.159/+0.242	+0.311/+0.513
REPTree	1.707	2.249	1.768	2.048	-0.084/+0.102	+0.458/+0.382
SMOReg	1.250	2.073	1.268	2.067	-0.794/-0.742	+0.029/+0.057
RandomTree	1.292	2.085	1.195	2.115	-0.858/-0.755	-0.065/+0.165
MultilayerPerceptron	1.743	2.524	2.176	3.097	-0.221/-0.355	+0.560/+0.566
DecisionTable	2.707	2.743	2.560	2.975	+0.112/+0.596	+0.148/+1.011
RBFNetwork	2.859	2.932	2.877	3.158	+0.615/-0.558	+0.133/-0.277
LinearRegression	3.414	3.456	3.426	3.463	-0.064/+0.063	-0.022/+0.100
LWL	5.157	5.395	5.456	5.139	+0.317/1.297	+0.195/+0.980
DecisionStump	6.242	6.248	6.267	6.279	+0.021/+1.415	+0.027/+1.427
SimpleLinearRegression	6.651	6.626	6.651	6.632	+0.094/+1.829	-0.030/+1.810
ZeroR	20.092	20.092	20.086	20.086	-0.0085/+12.837	-0.015/+12.841

dataset increased slightly on average, up to almost half a meter. This indicates the optimal density of data for our algorithms is closer to one reading every 1.5 meters.

B. Online

In the online phase we saved the models for the two best-performing algorithms on the full dataset to test. In total we collected 27 sets of test results for K* and 20 for the RBF model. The bulk of these results were collected several months after our initial data collection, indicating some stability of the original data and algorithms.

In a live setting, the difference in how these two algorithms work is very important. As K* is instance-based, it compares each new datapoint to every classified point in the dataset. In contrast, the RBF algorithm learns weights for each of the 172 attributes in the data and must only multiply these weights by the attribute values of a new datapoint, then add these together to predict a position. This is a much faster operation than the entropy calculation for each of the 3110 points in the dataset and makes a substantial difference in a live environment. Initially, K* took 30 to 45 seconds to calculate a position; much too long for real-world applications. In contrast, the RBF model predicts a location almost instantly.

1) Proposed Hybrid Approach: To solve the time problem for K* we decided to break our full dataset into smaller partitions and trained K* classifiers on each partition. Our partitioning can be seen in Figure 2. To determine which partition of the building the user was in, and thus which K* classifier to use to calculate the user's position, we again looked at machine learning methods, settling on a random forest model [2] which achieved over 96% accuracy in a tenfold cross-validation. This reduced the number of comparisons from 3110 to about 400 to 500 and substantially increased the speed from 30-45 seconds to 3 seconds: fast enough to be useful in a real-world setting. A hybrid approach such as this may also speed up other instance-based localization systems.

2) Online, Static Results: We were interested first in how the algorithms would perform with a user standing still at a point in the building. K* achieved accuracies within three meters at every point, with most predictions within one meter of the user's actual position. The RBF regression algorithm performed similarly in a static online test.

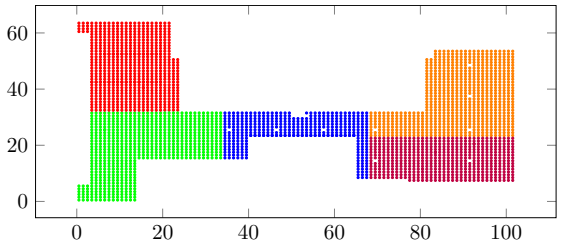

Fig. 2. Partitioned dataset, where each color corresponds to a partition.

TABLE III. ONLINE, IN-MOTION RESULTS SHOWING EACH ALGORITHM'S AVERAGE ERROR.

	K*		RBF Regression	
	Constant Orientation	Changing Orientation	Constant Orientation	Changing Orientation
Slow	6.03m	5.08m	7.85m	7.39m
Normal	6.39m	6.23m	8.86m	9.21m
Fast	9.19m	8.53m	10.15m	9.85m

3) Online, In-motion Results: After the static testing, we were interested in the performance of both algorithms in an online, in-motion testbed. We collected datasets with the user walking at various paces along the planned route to determine whether speed of movement affected accuracy. We looked at a normal pace of about 1.15 meters per second, a slow pace of 0.75 meters per second, and a quick pace of 1.69 meters per second. We also wanted to determine whether changing the device orientation would affect accuracy. Since data was collected with the device held facing south, we walked some routes constantly holding the device facing south. We also walked routes allowing the device to change orientation as we walked, mimicking a real-world user. The results can be seen in Table III. A single walked route using K* at a slow pace can be seen in Figure 3. The starting point is at x coordinate 4.5, y coordinate 1.5 in the bottom left and end point x coordinate 32.5, y coordinate 28.5 near the middle in red. The black line indicates the path walked with square points as vertices. The circle points in the plot indicate predicted positions.

Allowing the device to change orientation as the user walks does not appear to significantly affect accuracy. Removing this attribute may help increase accuracy as algorithms would have fewer attributes to build a model on.

It is significant that the testing was done at the library at Drake University. The left area of the live testing was amidst densely-packed bookshelves. Given that wireless signal strengths contributed most to the calculated position and the bookshelves likely affected signal propagation a great deal this area tended to have the highest error. Removing the positions in this area from our calculations improved accuracy an average of one meter, which may indicative of a typical environment.

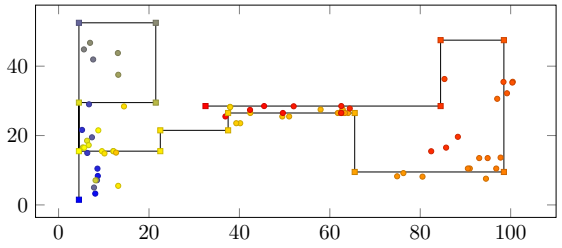

Fig. 3. Live Test: K* algorithm predicting position as the researcher walks the path in black allowing the device to change orientation. Color indicates time starting at blue, ending at red with smooth interpolation between.

V. CONCLUSIONS AND FUTURE WORK

In this work, we examined a large number of machine learning algorithms for indoor localization based on the sensors readily available in smartphones. We found algorithms as accurate as 0.76 meters on average in a real-world environment without the need for dedicated hardware or changes to infrastructure, outperforming algorithms considered in previous studies [9]. Our online, in-motion experiments show that our proposed hybrid approach achieved accuracy on par with the best offline instance-based methods with the speed of non-instance-based methods.

In the future, we will explore using multiple devices to collect and evaluate models on. Some small-scale experiments have shown promise in collecting datasets from multiple devices and aggregating them, then allowing the algorithms to build a model from this combined dataset. Other areas to investigate include taking multiple readings at different times, which may improve accuracy and reliability since signal strengths may fluctuate throughout the day. We will also investigate the impact of receiving delayed sensor readings for in-motion localization.

REFERENCES

[1] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala. Bluetooth and wap push based location-aware mobile advertising system. In *Proc. of the 2nd Int. Conf. on Mobile Systems, Appl., and Services*, MobiSys '04, pages 49–58, New York, NY, USA, 2004. ACM.

[2] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[3] R. Bruno and F. Delmastro. Design and analysis of a bluetooth-based indoor localization system. In M. Conti, S. Giordano, E. Gregori, and S. Olariu, editors, *Personal Wireless Commun.*, volume 2775 of *Lecture Notes in Computer Science*, pages 711–725. Springer Berlin Heidelberg, 2003.

[4] J. G. Cleary and L. E. Trigg. K*: An instance-based learner using an entropic distance measure. In *12th Int. Conf. on Mach. Learn.*, pages 108–114, 1995.

[5] E. Frank. Fully supervised training of Gaussian radial basis function networks in WEKA. Technical Report 04/14, Department of Computer Science, University of Waikato, 2014.

[6] A. Goswami, L. E. Ortiz, and S. R. Das. Wigem: A learning-based approach for indoor localization. In *Proc. of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 3:1–3:12, New York, NY, USA, 2011. ACM.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, nov 2009.

[8] E. D. Manley and J. Deogun. Location learning for smart homes. In *Adv. Information Netw. and Appl. Workshops, 2007, AINAW '07. 21st Int. Conf. on*, volume 2, pages 787–792, May 2007.

[9] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy. Precise indoor localization using smart phones. In *Proc. of the International Conference on Multimedia*, MM '10, pages 787–790, New York, NY, USA, 2010. ACM.

[10] L. Ni, Y. Liu, Y. C. Lau, and A. Patil. Landmarc: indoor location sensing using active rfid. In *Pervasive Comput. and Commun., 2003. (PerCom 2003). Proc. of the First IEEE Int. Conf. on*, pages 407–415, March 2003.

[11] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proc. of the 6th Annu. Int. Conf. on Mobile Comput. and Netw.*, MobiCom '00, pages 32–43, New York, NY, USA, 2000. ACM.

[12] B. Sohn, J. Lee, H. Chae, and W. Yu. Localization system for mobile robot using wireless communication with ir landmark. In *Proc. of the 1st Int. Conf. on Robot Comm. and Coordination*, RoboComm '07, pages 6:1–6:6, Piscataway, NJ, USA, 2007. IEEE Press.